Hidden Field Equations (HFE) and Isomorphisms of Polynomials (IP): two new Families of Asymmetric Algorithms

- Extended Version -

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

Abstract

In [11] T. Matsumoto and H. Imai described a new asymmetric algorithm based on multivariate polynomials of degree two over a finite field. Then in [14] this algorithm was broken. The aim of this paper is to show that despite this result it is probably possible to use multivariate polynomials of degree two in carefully designed algorithms for asymmetric cryptography.
In this paper we will give some examples of such schemes. All the examples that we will give, belong to two large family of schemes: HFE and IP. With HFE we will be able to do encryption, signatures or authentication in an asymmetric way. Moreover HFE (with properly chosen parameters) resist to all known attacks and can be used in order to give very short asymmetric signatures or very short encrypted messages (of length 128 bits or 64 bits for example). IP can be used for asymmetric authentications or signatures. IP authentications are zero knowledge.

Note 1 : Another title for this paper could be **"How to repair Matsumoto-Imai algorithm with the same kind of public polynomials"**.

Note 2 : This paper is the extended version of the paper with the same title published at Eurocrypt'96.

1 Introduction

Currently the security of most algorithms that we know in Asymmetric Cryptography for encryption or signatures rely on the (not proved) intractability of the factorisation or discrete log problem.
So today one of the problems of Asymmetric Cryptography is to find new and efficient algorithms for encryption or signatures that do not depend on these two closely related problems. (For authentication, thanks to new algorithms for example [19] or [20], the situation is much better now).
Another problem of Asymmetric Cryptography is to find a way to have very short asymmetric signatures (the shorter outputs have actually about 320 bits, with the DSS algorithm for example, or 220 bits with C. Schnorr's algorithm, but we can imagine that a much shorter asymmetric signature may exist).
Similarly another problem is to find a way to perform asymmetric encryption with short length outputs when the inputs are short (the shorter outputs have at the present about 512 bits).
The main subject of this paper is to describe a new class of asymmetric algorithms, called HFE which stands for "Hidden Field Equations". HFE is a candidate for these two problems.
The security of HFE is not proved but "apparently" it seems to be related to the problem of solving a system of multivariate quadratic equations over a finite field (for example $GF(2)$).
The general problem of solving a randomly selected system of multivariate quadratic equations over $GF(2)$ is NP-hard (cf [9] p. 251), and it is a completely different problem from the factorisation problem or the discret log problem. (However we will see that to recover a cleartext from an encrypted HFE text is not an NP complete problem, although this problem is expected to be exponentially difficult).

Moreover HFE with some well chosen parameters will give us a candidate algorithm for asymmetric signatures of 128 bits, or even 64 bits !
Similarly when HFE is used for encryption, parameters can be chosen in order to encrypt messages by blocks of 128 bits, or even by blocks of 64 bits.
However it should be noticed that HFE is not the first try to use multivariate quadratic equations over $\mathbf{F}_2 = GF(2)$ for an asymmetric cryptosystem: in [11] Matsumoto and Imai have designed such an algorithm, called C^*, and this algorithm was broken in [14].
Despite a lot of common points between HFE and the algorithm of [11], HFE has been especially designed to resist all the ideas of the attacks of [14] and we have made careful simulations about this.
The second family of algorithms that we will present is called IP which stands for "Isomorphisms of Polynomials". IP, as HFE can use public multivariate polynomials of degree 2 (or more). However IP is very different from HFE. IP authentications can be proved to be zero knowledge.

This paper is divided in four parts:

1. In sections 2,3,4,5 we describe and comment the "basic" HFE, *i.e.* the version of HFE with the easiest description.

2. In part II we will give some comments about the security of this "basic" HFE, and we will describe the "affine multiple attack". This attack detects some weak keys but is not efficient against well chosen parameters.

3. In part III we will see that there are a lot of variations of the "basic" HFE.

4. In part IV we will see a very different algorithm: IP. IP is only for authentications or signatures.

Finally we will conclude in section 24.

What is new in this extended version
In this extended version, many more details are given compared to the short paper version published in Crypto'96 with the same title. Moreover, some new sections have been added. For example:

1. In section 4.3, we explain how to generate HFE signatures of length only 64 bits (in the short paper, we explained only for length 128 bits).

2. In section 5, the different ways to solve $f(x) = y$ are explained.

3. In section 7 (example 4), we study whether a new quadratic permutation, found by Hans Dobbertin in [7], could be used in HFE.

4. In section 9, we explain some of the (unsuccessful) tries we did to cryptanalyze the HFE schemes.

5. In part IV, more details are given about the IP problems. However, this extended version is mainly focused on HFE, since another paper ([6]) is devoted explicitly to the IP problems.

6. In section 24, we present a HFE challenge with a total prize of US \$1000, that we offer for breaking an explicit example of HFE asymmetric signature of 80 and 128 bits.

Part I: The "basic HFE"

2 Mathematical Background

The main Mathematical properties needed are given in this section.

2.1 Function f

Let K be a finite field of cardinal q and characteristic p (typically but not necessary $q = p = 2$).
Let L_N be an extension of degree N of K.
Let β_{ij}, α_i and μ_0 be elements of L_N.
Let $\theta_{ij}, \varphi_{ij}$ and ξ_i be integers.
Finally let f be the function:

$$f : \begin{array}{ll} L_N & \to L_N \\ x & \mapsto \sum_{i,j} \beta_{ij} x^{q^{\theta_{ij}} + q^{\varphi_{ij}}} + \sum_i \alpha_i x^{q^{\xi_i}} + \mu_0. \end{array}$$

Then f is a polynomial in x.
Moreover let B be a basic of L_N.
Then the expression of f in the basis B is:

$$f(x_1, \ldots, x_N) = (p_1(x_1, \ldots, x_N), \ldots, p_N(x_1, \ldots, x_N))$$

where $p_1, \ldots, p_N$ are N polynomials in N variables of **degree 2**.
The reason for this is that for any integer λ, $x \mapsto x^{q^\lambda}$ is a linear function of $L_N \to L_N$.
The polynomials $p_1, \ldots, p_N$ are found by choosing a "representation" of L_N.
Such a "representation" is typically given by the choice of an irreductible polynomial $i_N(X)$ over K, of degree N, so we can identify L_N with $K[X]/(i_N(X))$.
It is then easy to find the polynomials $p_1, \ldots, p_N$.

2.2 Inversion of f

f is not always a permutation of L_N.
However the heart of our new algorithm HFE will be this theorem:

Theorem 2.1 *Let L_n be a finite field, with $|L_n| = q^n$ with q and n "not too large" (for example $q \leq 64$ and $n \leq 1024$).
Let $f(x)$ be a given polynomial in x in a field L_n, with a degree d "not too large" (for example $d \leq 1024$).
Let a be an element of L_n.
Then it is always possible (on a computer) to find all the roots of the equation $f(x) = a$.*

Proof. This result is a classical result about the root finding of polynomials over finite fields. Proof of this result, *i.e.* the classical and efficient algorithms for root finding, can be found in [1] pp. 17-26, or in [10] chapter 4 for example. Improved algorithms for root finding can be found in [21] and in [22]. Moreover at the end of this Part I, in section 5 we give some results about the expected complexity of theses algorithms, and how efficient they are on real values.

Notes.

1. Let d be the degree of the polynomial f.
   Then for all a of L_N there are **at most** d solutions x of $f(x) = a$. Moreover in practice there will be often very few solutions x of $f(x) = a$ (for example there will be often only 0, 1 or 2 solutions).

2. Sometimes f can be a permutation (an "Hermite's criterion" is given in [10] chapter 7 for this). However it seems to be difficult to find how to choose f to be a permutation when f has more than one monomial in x (cf [10] chapter 7 or [13]) because only monomials in $x^{q^i} + x^{q^j}$ or x^{q^i} are allowed.

3 Description of the basic HFE in encryption

In this section we will describe the "basic" HFE algorithm for encryption, *i.e.* the version with the easiest description. The "basic" HFE of this section is the HFE that we have studied the most so far. (We will give the other versions in part III of this paper, in order to show that there is really a large number of ways to obtain multivariate quadratic equations and to hide the trapdoor. Moreover some of these versions have some advantage, for example some of them have easier secret computations).

Representation x of the message M

A field K, with $q = p^m$ elements is public.
Each message M is represented by a value x where x is a string of n elements of K. (So if $p = 2$, each message will be represented by nm bits).
Moreover we will sometimes assume that some redundancy has been put in the representation of the messages (details about how this redundancy can be put are given below).

Encryption of x

The secret items will be:

1. An extension L_n of K of degree n.
2. A function f, as described in section 2.1 from L_n to L_n, with a degree d "not too large". (For example $d \leq 1024$, and more precisely a typical d could be such that $17 \leq d \leq 64$). (We will give more details about this in section 4).
3. Two affine bijections s and t of $K^n \to K^n$. (These affine bijections can be represented in a basis as polynomials of total degree one and with coefficients in K).

Note. It is also possible to consider that L_n is public (because it is possible to prove that instead of changing L_n we will have the same result if we change s and t, so we can consider that L_n is fixed).

The ciphering is described in figure 1 (this figure should be read from the top to the bottom). The ciphertext y is given by: $y = t(f(s(x)))$.

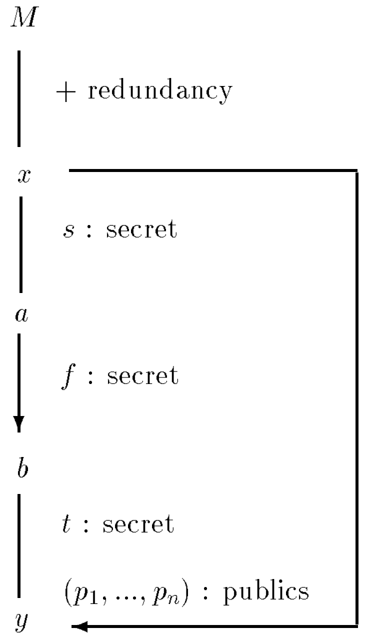

Figure 1: The "basic" HFE for encryption

An important point to notice is that since s and t are of degree one, and since f is of degree two in a basis, the composition of all these operations will still be a quadratic function in a basis.
So this function can be given by n polynomials with coefficients in K, $(p_1, \ldots, p_n)$. These polynomials give the components y_1, ..., y_n of the ciphertext y from the components x_1, ..., x_n of x:

$$\begin{cases} y_1 = p_1(x_1, ..., x_n) \\ y_2 = p_2(x_1, ..., x_n) \\ \quad \vdots \\ y_n = p_n(x_1, ..., x_n) \end{cases}$$

The public items are:

1. The field K of $q = p^m$ elements, and the length n.
2. The n polynomials $(p_1, \ldots, p_n)$ in n variables over K.
3. The way to put redundancy in the messages (*i.e.* the way to obtain x from M).

So no secret is needed to encrypt a message M.
Moreover decryption will be easy if all the secret items are known since all the operations given in figure 1 will be inverted. The inversion of the function f will be obtained by solving a polynomial equation with one variable in the field L_n, as it is explained in section 2.2 (and in section 5).
It is important to notice that since f is not necessarily a bijection, we may find more than one solution to this inversion (we will find at most d solutions because f is a polynomial of degree d in a (commutative) field). However, as we are to see, thanks to the redundancy given for M, the right solution M will be found.

Redundancy
We give two examples that show how some redundancy can be put in order to recover the right solution.

Example 1 ("Redundancy in the cleartext x"): Here, the length of the input x of the function is larger than the length of the message M, because in x we have M + some redundancy. For example if $K = \mathbf{F}_2$ and if we want to cipher a message M of k bits we will introduce for example 64 extra bits of redundancy so the message will be represented by $n = k + 64$ elements of K.
A nice way to put the redundancy is to make use of an error correcting code. We can also suggest that $x = M||h(M)$, where $||$ is the concatenation function and where $h(M)$ is the 64 first bits of a hash function such as MD5 or SHS. We can also use some simpler functions for $h(M)$ but it is important that each bits of $h(M)$ really depends on the bits of M in a non-linear way.

Example 2 ("Redundancy outside the cleartext x"): Here, $x = M$ is the message, where no special redundancy has been put. However, the ciphertext Y is not only y, but y + a one-way function of x.
For example, the ciphertext is $Y = y||\text{Hash}(x)$, where $||$ is the concatenation function, Hash is a public one-way hash function (for example Hash=MD5 or SHS), and $y = t(f(s(x)))$ as before.
It can be noticed that if Hash is a collision-free one-way hash function, then we will always be sure to find only one solution for the cleartext x.

Remarks :

1. Instead of using a one-way hash function, we can also have $Y = y||y_{n+1}$, ..., y_{n+k}, where $y_{n+1}, ..., y_{n+k}$ are given by k public polynomials p_{n+1}, ..., p_{n+k} of total degree two in x_1, ..., x_n. Unlike in p_1, ..., p_n, there is no trapdoor in p_{n+1}, ..., p_{n+k} (these extra polnomials are just here to eliminate the wrong values x). When k is small ($k \leq n$ for example), it is expected that these extra polynomials do not weaken the security of the system. We can also notice that these polynomials p_{n+1}, ..., p_{n+k} can also be mixed with p_1, ..., p_n in order to have public polynomials p'_1, ..., p'_{n+k}, where the p'_i have secret linear expressions in the p_1, ..., p_{n+k} values.
2. Of course, the redundancy is always done with public functions or equations (in order to have a public key encryption scheme).

Attack with related messages
Let y be the ciphertext of x, and let y' be the ciphertext of $x+d$, where d is a constant. If y, y' and d are known, then x will be easily found because $y - y'$ is of degree one in the x_i variables. A similar property exists in RSA when the public exponent is less than than 32 bits large (see [4]). For some applications, such a property must be avoided. For that purpose, one solution is to publish a public permutation g – for example g is a DES with a public key K – and to encrypt by $y = HFE(g(x))$ (instead of $y = HFE(x)$). Another solution might be to padd the plaintext with random bits.

This concludes the description of the "basic" HFE algorithm for encryption.

4 The basic HFE in signature and authentication

We will now see three examples of how we can use HFE for asymmetric signatures. In the first example the signatures will have 160 bits, in the second example the signatures will have about 128 bits, and in the third example, the signatures will have about 64 bits (or even about 32 bits if we allow very slow signature verification !).

4.1 Example 1

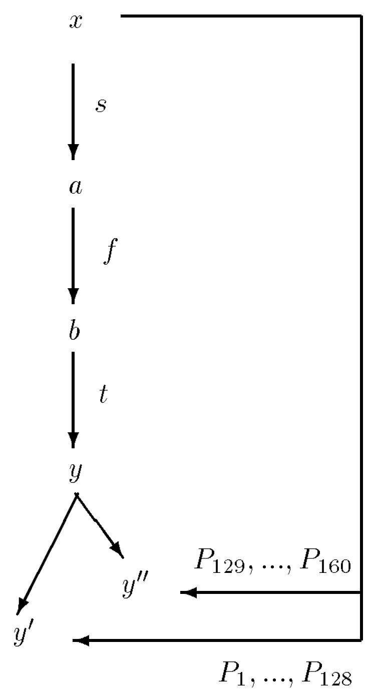

Figure 2: Example 1 of HFE in signature. x : the signature (160 bits). y' : the hash to sign (128 bits). $P_1, \ldots, P_{128}$ are public. $P_{129}, \ldots, P_{160}$ are secrets.

Let us consider an HFE algorithm, as described in the next sections, with x and y of about 128+32=160 bits.
Let $P_1, \ldots, P_n$ be the n public polynomials that give y from x, with $n = 160$ and $K = \mathbf{F}_2$ for example.
If only P_1 to P_{128} of these polynomials are public (the over are secret), then the polynomials $P_1, \ldots P_{128}$, give a value z of 128 bits from a value x of 160 bits.
In our algorithm here z is the hash of a message to sign and x will be the signature of z. When z is given, then with the secret polynomials P_{129} to P_{160} and the other secret values we will be able to find a value x so that from this x, the polynomials $P_1, \ldots, P_{128}$ will give exactly the value z.
For this we will padd z with 32 extra bits and try to find an x with a decryption of the HFE algorithm. If it fails we try with another padding until we succeed.
In figure 2 we illustrate such a use of HFE in signature.

4.2 Example 2

Computation of the signature

In this example 2 to sign a message M there will be three steps.

Step 1. We generate a small integer R with no block of numbers with 10000 in its expression in base 2 (for example $R = 0$ to start).
Step 2. We compute $h(R||10000||M)$ where h is a public collision free hash function with an output of 128 bits (for example h is the MD5 algorithm).
Step 3. We consider an HFE algorithm (as in section 3) with values x and y of 128 bits.
If we take $y = h(R||10000||M)$, then we can (with the secret key) try to find a cleartext x so that $HFE(x) = y$.
If we succeed, then $R||x$ will be the signature of M.
If we do not succeed (because since HFE is not a permutation some value y have no corresponding x) then we try again at Step 1 with another R (for example with the new R equal to the old $R+1$ if this new R has no block of 10000 in base 2).

Verification of a signature

The message M and a signature $R||x$ of M is given. First, we separate R and x (since x has a fix length of 128 bits this is easy). Then we compute $h(R||10000||M)$ and $HFE(x)$ and the signature is valid if $h(R||10000||M) = HFE(x)$.

Length of the signature

In this example 2 the length of the signature is not fixed. However in average R will be very small so that the signature $R||x$ will have in average just a few more than 128 bits.

Note. Of course the pattern 10000 is just an example and another pattern P can be chosen. More precisely the property that we want is that from $R||P||M$ we can recover R and M when we know that R do not have the pattern P. (So the pattern will have at least one 1 and one 0).

4.3 Example 3

Computation of the signature

Step 1, Step 2: These steps are as in example 2 above. We denote by h_1 the first 64 bits of the hash value, and by h_2 the last 64 bits of the hash value.

Step 3: We consider an HFE algorithm with values x and y of 64 bits. We will denote by F the public computation of this HFE function, and by F^{-1} one pre-image of the secret computation (so that $y = F(x)$).
We compute $S = F^{-1}(h_1 \oplus F^{-1}(h_2 \oplus F^{-1}(h_1)))$, *i.e.* we look (with the secrets) if there is at least a value S such that:

$$F\Big(F(F(S) \oplus h_1) \oplus h_2\Big) = h_1. \qquad (\#)$$

If we succeed, then $R||S$ will be the signature of M. If we do not succeed (because HFE is not a permutation, so that some values have no pre-image), then we go back to Step 1 with another R.

Verification of the signature

The message M and a signature $R||S$ of M are given. First, we separate R and S (since S has a fixed length of 64 bits, this is easy). Then we compute $h(R||10000||M) = h_1||h_2$, and the signature is valid if and only if the equation (#) above is satisfied.

Length of the signature
In this example 3 (as in example 2), the length of the signature is not fixed. However, in average, R will be very small, so that the signature $R||S$ will have in average just a little more than 64 bits.

Remark 1: If we allow slower signature verification, it is also possible to have even shorter signatures. For example, R will not be put in the signature, so that all the small values of R will be tried to verify a signature. Moreover, we can also decide that only the first 32 bits of S are given as the signature: the 32 other bits will have to be found by exhaustive search during a signature verification. In this case, the signature verification is very slow (it takes a few hours !), but the signatures have only 32 bits !

Remark 2: In this example 3, we assume that the cryptanalyst does not have access to $2^{48}.8 = 256$ Terabytes of memory. Because, if he had access to such a memory, he could store 2^{48} pairs $(y, F^{-1}(y))$, *i.e.* he could compute F^{-1} of a value with probability 2^{-16}. He would then generate successively 2^{48} messages and compute the 2^{48} values (h_1, h_2) of these messages. From these 2^{48} messages, about 2^{32} will be such that $F^{-1}(h_1)$ is in his table, and from them, about 2^{16} such that $F^{-1}(h_2 \oplus F^{-1}(h_1))$ is in his table, and finally about one such that $F^{-1}(h_1 \oplus F^{-1}(h_2 \oplus F^{-1}(h_1)))$ is in his table. Like this, he has computed a signature with complexity 2^{48} in time and memory.

Remark 3: Of course, by increasing the number of F^{-1} computations in the definition of S, we will make the attack given in remark 2 less and less efficient (*i.e.* its complexity will become closer and closer to 2^{64}), but this is at the cost of more computations in the generation and verification of a signature. Moreover, it also increases the average number of R values to try to get a signature.

4.4 The basic HFE in authentication

It is well known that each asymmetric algorithm that can be used for ciphering or for signature can also be used for authentication. So we can use the HFE algorithms for authentication.
For example we can encrypt a challenge with the public polynomials $P_1, \ldots, P_n$, and ask for the clear-text.

5 Appendix: Resolution of $f(x) = y$

5.1 The problem

Let f be a polynomial of degree d in $\mathbf{F}_{q^n}$, where $q = p^m$, p prime. Let $y \in \mathbf{F}_{q^n}$. We want to find all the solutions $x \in \mathbf{F}_{q^n}$ so that $f(x) = y$.
We will assume that p is small and that the computation of an operation in $\mathbf{F}_q$ is very easy (for example we have stored the tables of $\times$ and $+$ in $\mathbf{F}_q$). More precisely the integer m is defined such that the operations in $\mathbf{F}_{p^m}$ can be considered as one unit of operation. For example typically $m \leq 8$ in practice on a computer since it is then easy to store the 2^{2m} values of the tables of $\times$ and $+$.

5.2 The classical algorithms

There are three classical algorithms for this very general problem: the Berlekamp-Rabin algorithm, the Berlekamp trace algorithm, and a linearized polynomial algorithm. A description of these algorithms can be found in [1] pp.17-26 or in [10] chapter 4. We will just give here their expected complexity.

a. The Berlekamp-Rabin algorithm

This algorithm works when p is an odd prime. It is a non-deterministic algorithm. Its expected running time is $0(nd^2 \log d \log q)\mathbf{F}_{q^n}$ operations. Therefore since (with our choice of m) a multiplication in $\mathbf{F}_{q^n}$ is in $\mathcal{O}(n^2)$, it is in $\mathcal{O}(mn^3 d^2 \log d)$ from a practical point of view. (Faster algorithms exist asymptotically but this is the practical running time when operations in $\mathbf{F}_q$ are easy).
HFE can work with odd prime p but we have mainly studied the case of $p = 2$. So we have not actually used the Berlekamp-Rabin algorithm.

b. The linearized polynomial algorithm

As we will see in section 7 a careful analysis of the possible linearisation of the polynomial $f(x) - y$ is important to avoid weak keys.
However, we can always use a linearisation of the polynomial $f(x)-y$ to find the solutions x of $f(x) = y$ when we have the secret key, even if this linearised polynomial is useless for a cryptanalyst.

The algorithm proceeds in three parts.

Part 1. Find a linear multiple $A(x)$ of $f(x)$.
Part 2. Solve $A(x) = 0$.
Part 3. Test if the solutions found in Part 2 are indeed roots of f.
If the linearisation is made over $\mathbf{F}_2$ (*i.e.* with $1, x, x^2, x^4, \dots x^{2^{d-1}}$), then the expected running time for Part 1 is in $\mathcal{O}(d^3n^2)$, for Part 2 in $\mathcal{O}(m^3n^3 + dm^2n^3)$, and Part 3 is expected to took negligible time compared with Parts 1 and 2.

So the total expected time is in $\mathcal{O}(d^3n^2 + m^3n^3 + dm^2n^3)$.

c. The Berlekamp trace algorithm

This algorithm is quite efficient for large extension fields $\mathbf{F}_{q^n}$, where q is small.
We have only studied the algorithm when $q = 2^m$, but the algorithm is also efficient for small odd q.
With the same notations as before (*i.e.* when operations in $\mathbf{F}_q$ are the basic operations) the average expected complexity of the algorithm is in $\mathcal{O}(mn^3d^2 + n^2d^3)$. (Moreover the algorithm is deterministic and in the worst cases it will be in $\mathcal{O}(m^2n^4d^2 + mn^3d^3)$).

5.3 Improved algorithms

a. A modified linearized polynomial algorithm (nice when d is very small)

Following an idea of Van Oorschot and Vanstone (cf [11] or [20]) we can combine a generalization of the Berlekamp Trace algorithm with an affine multiple $A(x)$ of $f(x)$.
For example we can design an algorithm like this:

Step 1. Find an affine multiple $A(x)$ of $f(x)$ (as in the linearized polynomial algorithm).
Step 2. Compute $B(x) = x^{2^{mn}} + x$ modulo $A(x)$.
Step 3. Compute $C(x) = GCD(A(x), B(x))$.
Step 4. Compute $GCD(C(x), f(x)) = P(x)$.
Step 5. If the degree of P is small then solve $P(x) = 0$ (immediately if degree of $P = 1$, or with the Berlekamp Trace algorithm).
If the degree of P is not small then go back to step 2 with a trace function instead of $x^{2^{mn}} + x$.

The expected time for such an algorithm is in average in $\mathcal{O}(dmn^3 + d^3n^2)$. (So the term in n^3 is only dm instead of $m^3 + dm^2$). (The expected time are $\mathcal{O}(d^3n^2)$ for step 1, $\mathcal{O}(dmn^3)$ for step 2, $\mathcal{O}(d^2n^2)$ for steps 3 and 4, and steps 1,2,3,4 are expected to be dominant in time in average).

b. A modified Trace algorithm (nice when d is not too small)

The Trace functions have a transitivity property:
if $K \subset F \subset E$ then $Tr_{E|K}(\alpha) = Tr_{F/K}(Tr_{E|F}(\alpha))$.
For example if $E = \mathbf{F}_{2^N}$, with N even, and if $K = \mathbf{F}_2$ and $F = \mathbf{F}_4$ then if $Tr_{E/K}(x) = 0$ then $Tr_{E/F}(x) = \alpha$ or β, where $\mathbf{F}_4 = \{0, 1, \alpha, \beta\}$. So if $f(x)$ is a polynomial with all the roots x so that $Tr_{E/K}(x) = 1$, then we can try to obtain a factorisation of f with $GCD(f(x), Tr_{E/F}(x) - \lambda)$, where $\lambda = \alpha$ or β (we **do not** need to try $\lambda = 0$ or 1).
As another example if N is a multiple of 4 and if $f(x)$ is a polynomial will all its roots x with a constant Trace over $\mathbf{F}_4$ then we can try the Trace over $\mathbf{F}_{16}$ to factor $f(x)$, and only 4 values of this Trace are possible (instead of 16).

More generally we can use the values of the Trace E/F_i, with increasing subfields $F_i, F_1 \subset F_2 \dots$ to try to factor f.
So we can design an algorithm like this:

Step 1. Compute and store all the $u_k = x^{2^k} \bmod f(x), 1 \leq k \leq mn$. Let $P(x) = x^{2^{mn}} + x \bmod f(x)$.
Step 2. Compute $GCD(P(x), f(x)) = F(x)$.
Step 3. Compute and store all the $v_k = u_k \bmod F(x) = x^{2^k} \bmod F(x)$, $1 \leq k \leq mn$.
Step 4. Try to factor $F(x)$ with some Trace E/F_i with k increasing subfields $F_i, F_1 \subset \dots F_k$, as we have seen above. (For example $k = 4$ and $F_1 = \mathbf{F}_2, F_2 = \mathbf{F}_4, F_3 = \mathbf{F}_{16}$ and $F_4 = \mathbf{F}_{256}$. There are 256 possible values for the trace over $\mathbf{F}_{256}$ but we will only use the values that have a given Trace over $\mathbf{F}_{16}$ corresponding to roots of f with these trace value over $\mathbf{F}_{16}$).
Step 5. If all the roots of $F(x)$ have not been found after Step 4 then we will use the classical Berlekamp Trace algorithm to achieve the factorisation (*i.e.* we will use some $Tr(\alpha^j x)$).

The expected average complexity of this algorithm is in $\mathcal{O}(md^2n^3)$.
(Step 1 is expected to be in $\mathcal{O}(md^2n^3)$, Step 2 in $\mathcal{O}(n^2d^2)$ and these two steps are expected to be dominant in average).

c. von zur Gathen and Shoup algorithm (asymptotically fastest known algorithm)

In [22], von zur Gathen and Shoup present non-deterministic algorithm to solve $f(x) = y$ with a complexity $(d^2 + d\log(q^n))(\log d)^{\mathcal{O}(1)}\mathbf{F}_{q^n}$ operations.
This is about $d^2(\log d)^{\mathcal{O}(1)}n^2 + d(\log d)^{\mathcal{O}(1)}mn^3$ operations.

5.4 Real simulations

In [12] some running times of some root finding algorithms, applied to find all roots of polynomials $f(x)$ of degrees $d = 10, 20$ and 40 have been tabulated for fields $GF(2^n)$ for particular values n, as large as $n = 1013$.
For example for $n = 130$ and $d = 20$ it took them with a C program 106 seconds on a SUN 3/160. 106 seconds is quiet a long time but it shows that this is feasible. Moreover it seems that no storage of $\times$ and $+$ in a work field $\mathbf{F}_{2^m}$ have been done. If such a storage were done, for example with $m = 4$ or 8, then the computation time should be sensibly smaller. von zur Gathen and Shoup algorithm [22] may also be faster.

5.5 Example of smart card secret key computations

Usually, with HFE, the smartcard performs the public key computations and a computer performs the secret key computations. However, we can also evaluate the RAM needed and the time to perform a secret key computation F^{-1} in a smartcard (in some cases, as in example 3 of signature, three such F^{-1} computations are required).

Family 1 of secret key computation
In this family, computations like $(x^{2m} + x \bmod f(x))$ (Step 1), and then GCD$(A(x), B(x))$ (Step 2) are required, where $A(x)$, $B(x)$ and $f(x)$ are polynomials of degree d.
Step 1 requires about $\mathcal{O}(dmn^3)$ operations and Step 2 about $\mathcal{O}(d^2n^2)$ operations. If $d = 17$, $m = 2$ and $n = 32$, for example, and if $T = 64^2.2 = 8192$ is an evaluation of the number of 8 bits operations to perform a 512 bits modular multiplication, then Step 1 requires about $136T$ and Step 2 about $36T$. (To compare with about $768T$ for a 512 bits RSA secret key computation). In characteristic $p = 2$, the RAM required for Step 1 is about $(d - 1)$ times the length of the messages, and for Step 2 about twice this RAM. For messages of 64 bits and $d = 17$, this gives about 128 bytes of RAM for Step 1 and 256 bytes of RAM for Step 2.

Family 2 of secret key computation
We can also imagine that a linearized polynomial $A(x)$ has been precomputed so we have to solve $A(x) = 0$ by Gaussian reductions. In this case, about n^3 computations are required for the Gaussian reductions ($\simeq 4T$ with $n = 32$), but before the terms in $A(x)$ will have to be evaluated, and this sometimes requires a lot of time. The RAM needed is about $\frac{n^2}{2}$ elements of $\mathbf{F}_q$ (we divide by two because there is a Gaussian reduction and the equations can be computed one by one). This is about $\frac{32.32.2}{8.2} = 128$ bytes of RAM in our example.

5.6 Conclusion

When n and d are not too big it is always possible to find all the roots of f.
One algorithm is expected to be in average in $\mathcal{O}(mdn^3 + d^3n^2)$, another to be in average in $\mathcal{O}(md^2n^3)$ and a von zur Gathen and Shoup algorithm is expected to be in $d^2(\log d)^{\mathcal{O}(1)}n^2 + d(\log d)^{\mathcal{O}(1)}mn^3$. (Asymptotically these algorithms are faster but these complexities are the complexity for practical values of m, d, n).
Moreover they are algorithms for very general polynomials f. For special polynomials f (for example when f has just a few monomials with only one large exponent) much faster algorithms may exist. (However some of these special polynomials, are weak choices for the basic HFE, as we will see in section 7).

Part II: About the security of the basic HFE

6 Theoretical considerations about the security of the basic HFE

6.1 How to choose the parameters

We have mainly studied HFE in characteristic 2. However, in opposition to the Matsumoto-Imai scheme of [11] which needed $p = 2$, with HFE we can have any small prime value for p.
For security of the basic HFE it is necessary that:

1. The message M has at least 64 bits. (If not it is easy to find M by exhaustive search). So when some redundancy is put in x, then x and y may have at least about 128 bits. When the redundancy is put outside x, then x and y will have at least 64 bits.

2. The number n of variables in the public key is such that $d^{2.7n} \geq$ or $\simeq 2^{64}$, where d ($= 2$ in the basic HFE) is the degree of the public equations. So $n \geq 23$. This comes from the fact that there are some general Gröbner-bases algorithms able to compute the solutions of any set of equations of degree d with n variables with complexity about $\mathcal{O}(d^{3n})$ in theory and about $\mathcal{O}(d^{2.7n})$ empirically (cf [8]). (Remark : in [11], it was recommended to have $n \geq 32$, also to roughly avoid the general Gröbner-bases algorithms. However, $n \geq 23$ seems to be a more precise evaluation.)

3. The polynomial f must have at least two monomials in x. (If not the basic HFE will be in fact just a Matsumoto-Imai algorithm and will be attacked as shown in [2]). More generally, the polynomial f must be chosen in order that the "affine multiple attack" that we will describe in section 7, and the "quadratic attack" that we will study in section 8, will require too much computations to be performed.

However, it is not always easy to test when these attacks on a polynomial f can be dangerous or not. So instead of doing that, and in order to avoid all the weak keys, we suggest (in characteristic $p = 2$) to choose for f a random "quadratic over $\mathbf{F}_2$" polynomial of degree ≥ 33 (*i.e.* for the degree 33 :

$$f(x) = x^{33} + \alpha_{32}x^{32} + \alpha_{24}x^{24} + \alpha_{20}x^{20} + \alpha_{18}x^{18} + \alpha_{17}x^{17} + \alpha_{16}x^{16} + \alpha_{12}x^{12}$$

$$+\alpha_{10}x^{10} + \alpha_9 x^9 + \alpha_8 x^8 + \alpha_6 x^6 + \alpha_5 x^5 + \alpha_4 x^4 + \alpha_3 x^3 + \alpha_2 x^2 + \alpha_1 x$$

where all the α_i are random elements of $\mathbf{F}_{2^n}$), or we suggest the parameters given in the two challenges of section 24.
Moreover in order to make things even more difficult for a cryptanalysis we will see in section 11 that we can eliminate some public polynomials, and add some others.

Length of the public key If $f(x) = x^{17} + \alpha x^{16} + \beta x^5 + \gamma x^4 + \delta x^2 + \varepsilon x$, where α, β, γ, δ, ε are values of $\mathbf{F}_{q^n}$, where $q = 4$ and $n = 32$, then the length of the public key is about $n \cdot \frac{n(n-1)}{2} \cdot \frac{1}{4} \simeq 4$ Kbytes.

6.2 Comparison between the basic HFE and Matsumoto-Imai algorithm

The Matsumoto-Imai algorithm of [11] can be seen as a weak key version of the basic HFE algorithms. The main new ideas that we have introduced in the design of HFE are:

1. Not to have necessary a bijection, since we will be able to encrypt and to sign without bijections.

2. To use the fact that a lot of practical algorithms are known to find the roots of a lot polynomials in a finite field. For example it is always possible to find the roots when the polynomial is a univariate polynomial of not too large degree.

For security HFE may be much stronger than the Matsumoto-Imai algorithm because:

1. The equations found in [13] to attack the Matsumoto-Imai algorithm do not exist in general HFE schemes. (Moreover we have made simulations to check this point).

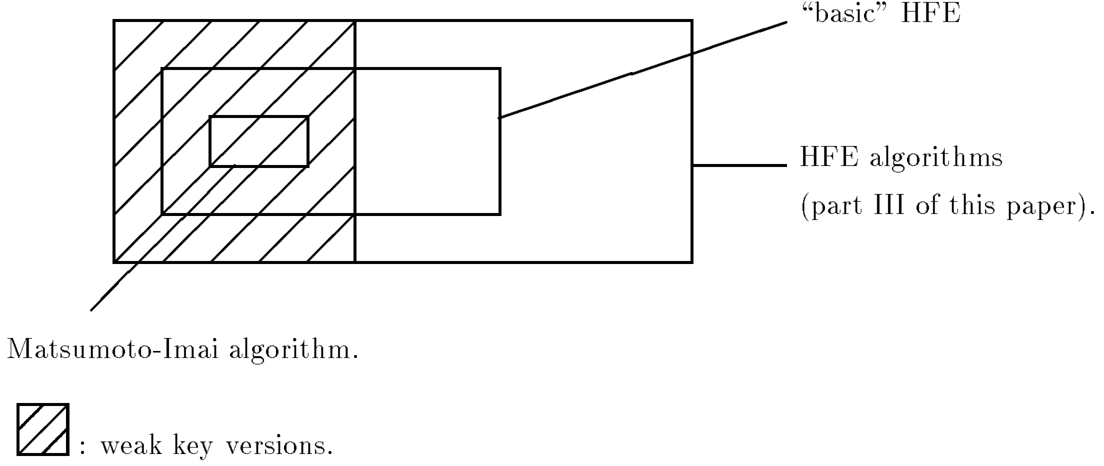

Figure 3: The Matsumoto-Imai can be seen as a weak key version of the basic HFE

2. In Matsumoto-Imai algorithm very few possibilities existed for f, so we could assume that f was public. However in HFE we have much more possible functions f so the cryptanalyst can no longer assume that f is public.

   HFE may still be secure if f is public, since its security may essentially be in the secrets affine functions s and t (this point leads to the IP authentication and signature scheme that we present in Part IV). The function f is "hidden", by the affine functions s and t, as the name HFE shows, but the function f is not necessary secret. However to keep f secret can only increase the security, so we actually prefer to recommend to keep f secret.

   Note. It is possible to prove that if the two exponents in x of higer degree are λ_1 and λ_2, with $\lambda_2 - \lambda_1$ coprime with $q^n - 1$, then we can assume, without loosing generality, that the coefficients in x^{λ_1} and x^{λ_2} are 1.

3. The fact that some of the P_i polynomials could not be made public could also increase the security of HFE since the trapdoor is then even more "hidden" (this point will be seen in section 9).

Public key computations are easy in HFE (for example they can easilly be done in low cost smart cards).
However, from a practical point of view, the basic HFE is slower in secret key computations than Matsumoto-Imai algorithm since the computation of f^{-1} is more complex.

6.3 Brassard Theorem

Some authentication algorithms (such as [19] or [20]) are proved to be as secure as a NP-hard problem. (This is a very nice result of security but of course this is not a proof of absolute security: a problem can be NP-hard but easy in average, or easy with bad parameters or difficult only with very large parameters). Can we also hope to prove that HFE is as secure as a NP-hard problem? No: from a generalisation of a theorem given by G. Brassard in [2], we can prove that recovering a cleartext from an encrypted HFE text is never an NP-hard problem (if NP $\neq$ co NP). However this is not really a flaw of HFE, but a property of almost all asymmetric encryption algorithms.

Idea of the proof. Let F be an asymmetric encryption algorithm with a secret key K and a public key k such that, when the secret key K is given and when a value y is given, it is always very easy to see if there is or not a cleartext x such that $y = F_k(x)$, *i.e.* such that y is the encryption of x by the algorithm F with the public key k. HFE, as all efficient encryption algorithms (such as RSA) has of course this property. Now let us consider the problem: "Is there an x such that $y = F_k(x)$?", where y is a given value. Then if the answer is "yes", x is a certificate that indeed the answer is "yes", *i.e.* it is easy to verify that the answer is "yes" if such an x is given (K is also another certificate). Moreover

if the answer is "no", K is a certificate that indeed the answer is "no". So this problem is in NP $\cap co$ NP. But (if NP $\neq$ *co* NP) there is no NP-hard problem in NP $\cap co$ NP. Similarly if from the secret key K we can compute easilly all the x such that $y = F_k(x)$, then the problem: "Is there an x such that $y = F_k(x)$ and $a \leq x \leq b$?", where a and b are two integers, is also in NP $\cap co$ NP. So recovering a cleartext x from its corresponding ciphertext y can not be a NP-hard problem. This shows that there is little hope to design any practical asymmetric encryption algorithm with a security proved to be based on a NP-hard problem.
It is also instructive to see that RSA may (or may not) be as secure as the factorisation problem because the factorisation problem is in $NP \cap co\,NP$ (so is not a NP-hard problem).

Comments about Brassard Theorem:
One can think that this result could suggest that when we have introduced a trapdoor in HFE, in order to have a cryptosystem useful for encryption, we may have weakened the problem. This result may also suggest that the problem on which the security of HFE relies is not clearly shown (it may not be the general NP-hard problem of solving randomly selected system of multivariate quadratic equations over $GF(2)$). (The same sort of suggestions occured for asymmetric encryption algorithms based on the knapsack problem, because the knapsack problem is also NP-hard. For example, in [3] p. 506, it is written: "[...] then it seems likely that there is an attack on the Merckle-Hellmann knapsack cryptosystem that runs faster than algorithms that solve the general knapsack problem").
However, some people in the cryptographic community do not share this idea that Brassard theorem "suggest" that some attack may exist that runs faster than algorithms to solve a more general NP-hard problem. And in fact, it is important to notice that the theorem does not give any explicit attack neither does it prove that some more efficient attack exist.
Let us consider again the following problem:

Input: A ciphertext message (of a public key cryptosystem) and an integer n.

Question: Is the n-th bit of the corresponding cleartext equal to 1 ?

As seen above, this problem is in NP $\cap co$ NP, hence it cannot be NP-hard (unless NP $=$ *co* NP). However, this remark gives no practical attack and moreover for some asymmetric encryption algorithm, the best algorithm might be the exhaustive search of the cleartext. So, despite the fact that this problem is not NP-hard, it might happen that the best algorithm is exhaustive search...
Similarly, in an HFE algorithm, recovering a cleartext from its HFE ciphertext is expected to be exponentially difficult when the HFE parameters are properly chosen. (However, despite the fact that it gives no attack, we believe that it is very relevant to take Brassard theorem into account, because from this theorem we know that we must not spend time in trying to prove that HFE is as secure as a NP-hard problem).

6.4 Security Simulations

In order to test the security of the Basic HFE we have made two sets of simulations:

1. Computation of some affine (in x) multiple of $f(x) - y$. This is described in section 7.

2. For $n = 17$ and $n = 22$, we have also made Toy simulations in order to see if the attacks given in [14] could also occur in the Basic HFE. This is described in section 9.

These simulations are useful to detect some weak keys, but they did not give us a way to break the HFE algorithm for well chosen parameters.

7 The affine multiple attack

7.1 Introduction

The "affine multiple attack" of the basic HFE that we will consider in this section 7 is a generalization of the main attack of [14] of the Matsumoto-Imai algorithm.

The "quadratic attack" that we will see in section 8, and the "affine multiple attack" are the only attacks that we know against the basic HFE that can sometimes be much better than exhaustive search on the cleartext.

7.2 Principle of the attack

Let f be a polynomial used in the basic HFE algorithm. Let d be the degree of f. By using a general algorithm (see for example [1] p. 25) we know that there are always some affine (in x) multiple $A(x,y)$ of the polynomial $f(x) - y$. (This means that $x \mapsto A(x,y)$ is an affine function and that each solution x of $f(x) = y$ is also a solution of $A(x,y) = 0$). $A(x,y)$ can be found by computing 1, x, x^q, x^{q^2}, ..., $x^{q^{d-1}}$ mod $f(x)$, and by writing that these $d+1$ polynomials are linearly dependent (since they are in the vector space of dimension d of all polynomials mod$f(x)$ (see [1] p. 25 for further details). Moreover, this shows that the degree of $A(x,y)$ – seen as a univariate polynomial in x – is at most q^{d-1}. For example in characteristic 2 the polynomial $A(x,y)$ will have at most $1, x, x^2, x^4, x^8 \ldots, x^{2^{d-1}}$ as monomials in x.
From now on we will assume for simplicity that the characteristic is 2.
Moreover, sometimes for such an affine multiple $A(x,y)$ all the exponents in y have small Hamming Weight in base 2. If this occurs, then the polynomial f will be a weak key for HFE.
More precisely if all the exponents in y have a Hamming Weight $\leq k$, then there will be an attack with a Gaussian reduction on $\mathcal{O}(n^{1+k})$ terms (more precisely with about $\sum_{i=1}^{k} n^{1+i}/i!$ terms because we have about $n^{1+i}/i!$ terms of total degree i in the y_j variables, $1 \leq i \leq k$) where n is the number of bits of the message. This attack will work exactly as the attack of [14] for the Matsumoto-Imai algorithm. The Gaussian reduction needed may be easier than a general Gaussian reduction but the complexity will be at worst in $\mathcal{O}(n^{3k+3})$ and at least in $\mathcal{O}(n^{1+k})$. Gaussian reductions with N terms are asymptotically in N^ω with $\omega < 2.376$ (cf [5]). Moreover we can choose some x but not some y, so in the equations on which we need to do a Gaussian reduction we will have at least $\mathcal{O}(n^{1+2k})$ unpredictible values. So it seems that more precisely the Gaussian reduction needed will be at most in $\mathcal{O}(n^{(1+k)\omega})$ and at least in $\mathcal{O}(n^{1+2k})$.

7.3 Examples

The examples 1,2 and 3 that we will present are easy. The other examples have been obtained with the "nullSpace" instruction of AXIOM.

Example 1 (an example of C^* permutation): Let $f(x) = x^3$. So $x^3 = y$.
Then $x^4 = yx$.
So $A(x,y) = x^4 + yx$ is an affine (in x) multiple of $f(x) + y$, and here all the exponents in y have a Hamming Weight ≤ 1.
This leads to an attack with a Gaussian reduction on $\mathcal{O}(n^2)$ terms as described in [14].

Example 2 (the general C^* permutation): Let $f(x) = x^{1+2^\theta}$. So $x^{1+2^\theta} = y$.
Then $x^{2^{2\theta}}.y = x.y^{2^\theta}$.
So $A(x,y) = x^{2^{2\theta}}y + xy^{2^\theta}$ is an affine multiple of $f(x) + y$, and here also all the exponents in y have a Hamming Weight ≤ 1.
So this also leads to an attack with a Gaussian reduction on $\mathcal{O}(n^2)$ terms as described in [14].

Example 3 (an example of Dickson permutation): Let $f(x) = x^5 + x^3 + x = y$.
Then it is possible to prove that

$$y^3.x + (y^2+1)x^4 + x^{16} = 0$$

Here all the exponents in y have a Hamming Weight ≤ 2.
So this leads to an attack of the HFE algorithm with a Gaussian reduction on $\mathcal{O}(n^3)$ terms if this polynomial f is used.
So this polynomial should not be used (it's a weak polynomial).

Remarks **1.** If $q^n \equiv 2 \bmod 5$ or $q^n \equiv 3 \bmod 5$, then this polynomial $f(x)$ is a permutation: it is a Dickson polynomial (see [10], chapter 7).
2. This polynomial f will also be studied in section 8.1 for $n = 13$ and we will also conclude that this polynomial is weak.
3. We also have $y^4.x + (y^3 + y)x^4 + y.x^{16} = 0$, $y^5.x + (y^4 + y^2)x^4 + y^2x^{16} = 0$ etc. and the result that we will see in section 8.2 (*i.e.* that there are $78 = 6 \times 13$ independent equations) suggest that 6 such equations may exist, with a degree one in x and two in y.

Exemple 4 (the general Dickson permutation of degre 5):

Let $f(x) = x^5 + ax^3 + a^2x$. Then $xy^3 + x^4(a^6 + ay^2) + x^{16} = 0$.
Here again, all the exponents in y have a Hamming weight ≤ 2. So this also leads to an attack with a Gaussian reduction on $\mathcal{O}(n^3)$ terms.

Remark: It is possible to prove that (in characteristic 2) the Dickson permutations of degree ≥ 5 either are not quadratic multivariate polynomials, or are a linear transformation of a C^* or of a Dickson polynomial of degree ≤ 5.

Example 5 (Dobbertin permutation):

Let $f(x) = x^{2^{m+1}+1} + x^3 + x$, in the field $GF(2^n)$, where $n = 2m + 1$ is odd. Then:

$$f(x) = y \Rightarrow x^9 + yx^6 + x^5 + yx^4 + (b + y^2)x^3 + y^2x + y^3 = 0,$$

where $b = y^{2^{m+1}}$.
Moreover, with MAPLE, we found that:

$$f(x) = y \Rightarrow xby^3 + y^4x^2 + y^2x^2b + x^4y^2 + x^4b^2 + x^4by^2 + x^4y^4 + x^8y^2 + x^8 + x^{16} = 0.$$

(Here, as above, $b = y^{2^{m+1}}$.)
In this affine multiple $A(x, y) = 0$ of $f(x)$, the largest Hamming weight of the exponents in y is 3. As a result, this leads to an attack with a Gaussian reduction on $\mathcal{O}(n^4)$ terms if this polynomial $f(x)$ is used. Therefore, we do not recommend to use this polynomial.

Remarks: **1.** Our main motivation to study this polynomial comes from the fact that Hans Dobbertin proved that $f(x)$ is a permutation (see [7]).
2. This polynomial f will also be studied in section 8.1 for $n = 13$, and we will show there that no relation with Hamming weight ≤ 2 in y and affine in x exist. In fact $A(x, y)$ here has a Hamming weight $= 3$ in y (so it is indeed ≥ 3).

Example 6:

Let $f(x) = x^9 + x^6 + x^5 + x^3 + x = y$.
Then the affine multiple $A(x, y)$ of $f(x)$ of degree 2^8 in x found by AXIOM is:

$$\begin{array}{ll}
 & (y^{27} + y^{24} + y^{23} + y^{20} + y^{19} + y^{11} + y^8 + y^7 + y^4 + y^3)x \\
+ & (y^{27} + y^{25} + y^{21} + y^{20} + y^{15} + y^9 + y^7 + y^5 + y^4 + y^3)x^2 \\
+ & (y^{28} + y^{26} + y^{25} + y^{20} + y^{18} + y^{16} + y^{14} + y^9 + y^8 + y^6 + y^4 + 1)x^4 \\
+ & (y^{22} + y^{21} + y^{18} + y^{16} + y^{15} + y^{14} + y^{13} + y^{10} + y^8 + y^7)x^8 \\
+ & (y^{25} + y^{22} + y^{21} + y^{19} + y^{17} + y^{12} + y^{11} + y^6 + y^5)x^{16} \\
+ & (y^{23} + y^{20} + y^{19} + y^{18} + y^{17} + y^{16} + y^{15} + y^{14} + y^{13} + y^{11} + y^{10} + y^9 + y^8 + y^6 + y^5)x^{32} \\
+ & (y^{18} + y^{17} + y^{14} + y^{11} + y^{10} + y^9 + y^6 + y^3)x^{64} \\
+ & (y^{13} + y^{11} + y^5 + y^4 + y^3)x^{128} \\
+ & x^{256}.
\end{array}$$

In $A(x, y)$ the largest Hamming Weight of the exponents in y is 4.
So this leads to an attack with a Gaussian reduction on $\mathcal{O}(n^5)$ terms if this polynomial is used.

This attack will need a lot of power but may be feasible. (For example if $n = 64$ it will need Gaussian reduction on 2^{25} variables ($\simeq n^5/4!$) and if $n = 128$ it will need Gaussian reduction on 2^{30} variables...). So we do not recommend to use this function f. (We have just presented this function in order to show the increasing complexity of the affine multiple attack when different functions are chosen).

Remark. This polynomial f will also be studied in section 8.2 for $n = 13$ and we will show there that no relation with Hamming Weight ≤ 2 in y and affine in x exist.
In fact $A(x, y)$ here has a Hamming Weight $= 4$ in y, and this is indeed ≥ 3 as expected.

Example 7:

Let $f(x) = x^{12} + x^8 + x^4 + x^3 + x^2 + x = y$.
Then one of the affine multiple of $f(x)$ found by AXIOM is:

$$\begin{array}{l} x^{256} + (y^{16} + y)x^{64} + (y^8 + y^5 + y^2)x^{16} + (y^3 + 1)x^4 \\ +yx + y^{16} + y^8 + y^5 + y^3 + y^2 = 0. \end{array}$$

Here all the exponents in y have a Hamming Weight ≤ 2.
So this polynomial f should not be used for HFE.
Since the degree of f was not so small (it was 12), and since f had a lot of monomials (6), this example shows that the affine multiple attack has to be taken seriously: it is not always obvious whether it works or not.

Example 8:

Let $f(x) = x^{17} + x^{16} + x = y$.
Then with AXIOM we have very easilly found 9 indepents affine multiple of $f(x) + y$, and one of this affine multiple is:

$$\begin{array}{lll} x^{256} & + & (1 + y + y^2 + y^3 + \ldots y^{15})x \\ & + & (y + y^2 + y^3 + \ldots + y^{15}) = 0. \end{array} \qquad (1).$$

In this equation the exponent in y with the largest Hamming Weight is 15 and it as a Hamming Weight of four.
However from (1) is obvious that:

$$(y - 1)x^{256} + (y^{16} - 1)x + y^{16} + y = 0. \qquad (2).$$

And in this equation (2) all the exponents in y have a hamming weight ≤ 1.
So this function f is very easy to attack.
Moreover this function f gives a good example that from a affine multiple like (1) it may be possible to found an even more devastating affine multiple like (2)...

Example 9:

Let $f(x) = x^{17} + x^9 + x^4 + x^3 + x^2 + x = y$.
With AXIOM, we have computed the least affine multiple $A(x, y)$ of $f(x) + y$ (it took us two days on a workstation).
In $A(x, y)$ all the exponents in y are ≤ 3840, and the exponent with the largest Hamming Weight as a Hamming Weight of 11.
So this affine multiple leads to an attack of HFE with this polynomial f with a Gaussian reduction on $\mathcal{O}(n^{12})$ terms, where $n \geq 64$. (For $n = 128$ it will need Gaussian reduction on 2^{58} terms because $2^{58} \simeq n^{12}/11!$ and for $n = 64$ it will need Gaussian reductions on 2^{47} terms).
Since this attack is completely impracticable, this polynomial f resists to the "affine multiple attack" and may be a strong polynomial for HFE.

Example 10:

Let $f(x) = x^{17} + x^{16} + x^5 + x = y$.
With AXIOM (also after two days of computations) we have computed the least affine multiple $A(x, y)$ of $f(x) + y$. In $A(x, y)$ the exponents with the largest Hamming Weight have a Hamming Weight also of 11.
So this function may also be a strong polynomial for HFE.

Note: What is nice with this function is that this function is not only quadratic over $\mathbf{F}_2$ but also quadratic over $\mathbf{F}_4$. (So the public computations will be easier with this function).

Example 11:

Let $f(x) = x^{17} + x^{16} + x^2 + x$.
Then the affine multiple $A(x, y)$ of $f(x)$ of degree 2^{16} in x found by AXIOM is:

$$\begin{aligned}
& (y^{3840} + y^{3634} + y^{3592} + y^{3510} + y^{3480} + y^{3456} + y^{3382} + y^{3352} \\
& +y^{3304} + y^{3300} + y^{3274} + y^{3262} + y^{3232} + y^{3216}) \\
+ \; & (y^{3630} + y^{3300} + y^{3270})x \\
+ \; & (y^{3292})x^2 \\
+ \; & (y^{3576} + y^{3246} + y^{3216})x^4 \\
+ \; & (y^{3634} + y^{3304} + y^{3274} + y^{3184})x^8 \\
+ \; & (y^{3630} + y^{3615} + y^{3600} + y^{3285} + y^{3255} + y^{3240})x^{16} \\
+ \; & (y^{3592} + y^{3292})x^{32} \\
+ \; & (y^{3576} + y^{3516} + y^{3456} + y^{3246} + y^{3186} + y^{3156} + y^{3126} + y^{3096})x^{64} \\
+ \; & (y^{3514} + y^{3154} + y^{3064} + y^{2944})x^{128} \\
+ \; & (y^{3840} + y^{3615} + y^{3600} + y^{3510} + y^{3480} + y^{3390} + y^{3360} \\
& +y^{3285} + y^{3270} + y^{3255} + y^{3240} + y^{3000})x^{256} \\
+ \; & (y^{3382} + y^{3352} + y^{3262} + y^{3232} + y^{3142} + y^{3112})x^{512} \\
+ \; & (y^{3516} + y^{3186} + y^{3156} + y^{3126} + y^{3096} + y^{2646} + y^{2616} + y^{2496} + y^{2166} + y^{2136})x^{1024} \\
+ \; & (y^{3514} + y^{3184} + y^{3154} + y^{3064} + y^{2944} + y^{1654} + y^{1624} + y^{1024})x^{2048} \\
+ \; & (y^{3390} + y^{3360} + y^{3000} + 1)x^{4096} \\
+ \; & (y^{3142} + y^{3112})x^{8192} \\
+ \; & (y^{2646} + y^{2616} + y^{2496} + y^{2166} + y^{2136})x^{16384} \\
+ \; & (y^{1654} + y^{1624} + y^{1024})x^{32768} \\
+ \; & x^{65536}.
\end{aligned}$$

In $A(x, y)$ the largest Hamming Weight of the exponents in y is 8.
So this leads to an attack with a Gaussian reduction on $\mathcal{O}(n^9)$ terms if this polynomial is used.
This attack is unpractical if $n = 128$ for example (because it will need Gaussian reduction on 2^{47} variables).
So this polynomial f may be a good choice for HFE.
Moreover this f is not only quadratic over $\mathbf{F}_2$ but also over $\mathbf{F}_{16}$, and this will give easier public key computations.
However, even if $A(x, y)$ is not useful, it has a particulary short expression and this may means that $f(x)$ is very special, so it's perhaps more risky to choose this $f(x)$ than a random polynomial, quadratic over $\mathbf{F}_2$, and of degree 17.

7.4 Asymptotic complexity

For large d, and for most of the polynomials f of degree d, the degree (in x) of the affine multiple $A(x, y)$ will be q^{d-1}, the largest Hamming weight of the exponents in y is expected to be in $\mathcal{O}(d)$, so that the complexity of the affine multiple attack of the basic HFE with this polynomial f is expected to be in $\mathcal{O}(n^{\mathcal{O}(d)})$.
So, if $d = \mathcal{O}(n)$ the complexity of the attack is expected to be exponential in n. (Of course public and secret computations for the legitimate users are polynomials in n, but the attack is expected to be exponential in n).

Moreover, $d = \mathcal{O}(\ln n)$ is expected to be sufficient to avoid all polynomial attacks.

7.5 Conclusion

The affine multiple attack is very efficient for some very special polynomials. However when the degree of f is ≥ 17 and when f has enough monomials of Hamming Weight two in x, this attack is expected to fail completely. Moreover, for some polynomials of practical interest it is even possible to compute with AXIOM the least affine (in x) multiple of these polynomials and to see if the attack will work or fail with this affine multiple.

Note For easier computations, we have chosen in all the examples the constant terms in the monomials of f equal to 0 or 1.
Of course this is not an obligation and any elements of the extension field can be chosen. Moreover if we want to keep f secret, some secret values for these terms should be chosen. (For example $f(x) = x^{17} + \alpha_9 x^9 + \ldots + \alpha_0$ with the numbers $\alpha_i \neq 0$ and 1).
We can also notice that, when all the constant terms are 0 or 1, then f commutes with x^2, x^4, x^8, ...
Therefore, there will be some linear functions u and v such that $v(y) = P(u(x))$, where $y = P(x)$ is the public key. It is not clear whether this may be dangerous, but this can suggest to choose very general terms and not only 0 and 1.

8 The "quadratic" attack

Idea of the attack

In section 7, we generate some affine relations on the bits of the cleartext (when an explicit value for the ciphertext y is given).
In this section 8, we now study how some quadratic relations on the bits of the cleartext might be useful in order to recover the cleartext. The motivation is that – as we will see in the tables computed in section 9 – for many polynomials f, we can indeed obtain more such quadratic relations than we have in the public key.
Let λ be the number of independent quadratic equations obtained on the bits of the cleartext. If λ is larger than, or is approximately $\frac{n(n+1)}{2}$, then by Gaussian reductions on $X_{ij} = x_i \cdot x_j$, we will probably obtain the value x. Moreover, even if λ is smaller than $\frac{n(n+1)}{2}$, these equations will give an attack more efficient than the exhaustive search on x. We will be able to find (by Gaussian reductions on $X_{ij} = x_i \cdot x_j$) about $\sqrt{2\lambda}$ variables, so that we will have to perform an exhaustive search on only $n - \sqrt{2\lambda}$ variables.

The "cubic" attack, or higher degree attacks

From a theoretical point of view, we can also imagine to collect some cubic equations in the bits of the cleartext (or even of higher degree). However, the detection of such equations requires a lot of computing power, so that we did no simulations on this. Moreover, if we assume that (after maybe a lot of computations) λ cubic equations in x_i have been found, then we will have to perform exhaustive search on $n - \sqrt[3]{6\lambda}$ variables (instead of n).

9 Toy simulations with small n

9.1 Classification of the equations

Despite the fact that it is a bit boring, we will describe in this section the different families of equations of total degree two or three that are stable by affine transformation on x_i and y_j variables.

Equations of degree total two and of degree one in x:

- $[Y^2]$ are the equations where the only quadratic monomials are in $y_i y_j$, i.e.:

$$\sum \mu_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0. \qquad [Y^2]$$

- $[XY]$ are the equations where the only quadratic monomials are in $x_i y_j$, i.e.:

$$\sum \mu_{ij} x_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0. \qquad [XY]$$

  Note: These equations $[XY]$ were the key idea for the cryptanalysis of C^*.

- $[XY + Y^2]$ are the equations where the only quadratic monomials are in $x_i y_j$ or $y_i y_j$, i.e.:

$$\sum \mu_{ij} x_i y_j + \sum \gamma_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0. \qquad [XY + Y^2]$$

Equations of degree total two and of degree two in x:

- $[X^2]$ are the equations where the only quadratic monomials are in $x_i x_j$, i.e.:

$$\sum \mu_{ij} x_i x_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0. \qquad [X^2]$$

  Remark: The vector space of these equations $[X^2]$ is of dimension exactly n, whatever the quadratic functions y_i are (because if $P(x_1, ..., x_n) = 0$ for all $(x_1, ..., x_n)$, then $P = 0$). Therefore, these equations do not give informations for any attack.

- $[X^2 + Y^2]$ are the equations where the only quadratic monomials are in $x_i x_j$ or $y_i y_j$, i.e.:

$$\sum \mu_{ij} x_i x_j + \sum \gamma_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0. \qquad [X^2 + Y^2]$$

- $[X^2 + XY]$ are the equations where the only quadratic monomials are in $x_i x_j$ or $x_i y_j$, i.e.:

$$\sum \mu_{ij} x_i x_j + \sum \gamma_{ij} x_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0. \qquad [X^2 + XY]$$

- Finally $[X^2 + XY + Y^2]$ are the general equations of total degree two, i.e.:

$$\sum \mu_{ij} x_i x_j + \sum \gamma_{ij} x_i y_j + \sum \nu_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0. \qquad [X^2 + XY + Y^2]$$

Equations of total degree three and of degree one in x:

- $[Y^3]$ are the following equations:

$$\sum \mu_{ijk} y_i y_j y_k + \sum \lambda_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [Y^3]$$

- $[XY + Y^3]$ are the following equations:

$$\sum \mu_{ijk} y_i y_j y_k + \sum \lambda_{ij} y_i y_j + \sum \nu_{ij} x_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [XY + Y^3]$$

- $[XY^2]$ are the following equations:

$$\sum \mu_{ijk} x_i y_j y_k + \sum \lambda_{ij} y_i y_j + \sum \nu_{ij} x_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [XY^2]$$

Remark: We will make some simulations with those equations $[XY^2]$. For any quadratic functions y_i, in the x_j variables, we will always have $n(n+1)$ "trivial" equations $[XY^2]$ if $K = \mathbf{F}_2$: they come from $y_i^2 = y_i$ and $x_i y_j^2 = x_i y_j$.

- $[XY^2 + Y^3]$ are the following equations:

$$\sum \mu_{ijk} y_i y_j y_k + \sum \nu_{ijk} x_i y_j y_k + \sum \lambda_{ij} y_i y_j + \sum \gamma_{ij} x_i y_j$$
$$+ \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [XY^2 + Y^3]$$

Equations of total degree three and of degree two in x:

- $[X^2 + Y^3]$ are the following equations:

$$\sum \mu_{ijk} y_i y_j y_k + \sum \lambda_{ij} y_i y_j + \sum \nu_{ij} x_i x_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^2 + Y^3]$$

- $[X^2 + XY + Y^3]$ are the following equations:

$$\sum \mu_{ijk} y_i y_j y_k + \sum \lambda_{ij} y_i y_j + \sum \nu_{ij} x_i y_j + \sum \xi_{ij} x_i x_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^2 + XY + Y^3]$$

- $[X^2 + XY^2]$ are the following equations:

$$\sum \mu_{ijk} x_i y_j y_k + \sum \lambda_{ij} y_i y_j + \sum \nu_{ij} x_i y_j + \sum \xi_{ij} x_i x_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^2 + XY^2]$$

- $[X^2 + XY^2 + Y^3]$ are the following equations:

$$\sum \mu_{ijk} y_i y_j y_k + \sum \nu_{ijk} x_i y_j y_k + \sum \lambda_{ij} y_i y_j + \sum \gamma_{ij} x_i y_j$$
$$+ \sum \varphi_{ij} x_i x_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^2 + XY^2 + Y^3]$$

- $[X^2 Y]$ are the following equations:

$$\sum \mu_{ijk} x_i x_j y_k + \sum \lambda_{ij} x_i x_j + \sum \nu_{ij} x_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^2 Y]$$

Remark: We will make some simulations with those equations $[X^2Y]$. For any quadratic functions y_i in the x_j variables, we will always have $n + \frac{n(n+1)}{2}$ "trivial" equations $[X^2Y]$ if $K = \mathbf{F}_2$. They come from the n public equations (*i.e.* $y_i =$"y_i"), from $y_i \cdot$"y_i"$= y_i$ and from $y_i \cdot$"y_j"$=$"y_i"$\cdot y_j$, where "y_i" and "y_j" are written in x. Moreover, if we compute with a representation where x_i^2 and x_i are not the same formal variable, the we will also have the $n + n^2$ trivial equations $x_i^2 = x_i$ and $x_i^2 \cdot y_j = x_i \cdot y_j$ (then it gives $n + \frac{3n(n+1)}{2}$ equations).

- $[X^2 Y + Y^3]$ are the following equations:

$$\sum \mu_{ijk} y_i y_j y_k + \sum \nu_{ijk} x_i x_j y_k + \sum \lambda_{ij} y_i y_j + \sum \gamma_{ij} x_i y_j$$
$$+ \sum \varphi_{ij} x_i x_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^2 Y + Y^3]$$

- $[X^2 Y + XY^2]$ are the following equations:

$$\sum \mu_{ijk} x_i y_j y_k + \sum \nu_{ijk} x_i x_j y_k + \sum \lambda_{ij} x_i x_j + \sum \gamma_{ij} x_i y_j$$
$$+ \sum \varphi_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^2 Y + XY^2]$$

Remark We will make some simulations with those equations $[X^2Y + XY^2]$. If $K = \mathbf{F}_2$, we will always have $3n(n+1)$ "trivial" equations of this type (and n^2 if $K = \mathbf{F}_{p^m}$ with $p \neq 2$).

- $[X^2Y + XY^2 + Y^3]$ are the following equations:

$$\sum \mu_{ijk} y_i y_j y_k + \sum \nu_{ijk} x_i y_j y_k + \sum \xi_{ijk} x_i x_j y_k + \sum \lambda_{ij} y_i y_j$$
$$+ \sum \gamma_{ij} x_i y_j + \sum \varphi_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^2Y + XY^2 + Y^3]$$

Equations of total degree three and of degree three in x:

- $[X^3]$ are the following equations:

$$\sum \mu_{ijk} x_i x_j x_k + \sum \lambda_{ij} x_i x_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^3]$$

Remark: For all quadratic functions y_j, the vector space of those equations has always dimension exactly n (because if $P(x_1, ..., x_n) = 0$ for all $(x_1, .., x_n)$, then $P = 0$). As a result, those equations give no informations for any attack.

- $[X^3 + XY]$ are the following equations:

$$\sum \mu_{ijk} x_i x_j x_k + \sum \lambda_{ij} x_i x_j + \sum \nu_{ij} x_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^3 + XY]$$

Remark: For all quadratic functions y_j, the vector space of those equations has always dimension exactly $n^2 + n$ (because if $P(x_1, ..., x_n) = 0$ for all $(x_1, .., x_n)$, then $P = 0$). As a result, those equations give no informations for any attack.

- $[X^3 + Y^2]$ are the following equations:

$$\sum \mu_{ijk} x_i x_j x_k + \sum \lambda_{ij} x_i x_j + \sum \nu_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^3 + Y^2]$$

- $[X^3 + XY + Y^2]$ are the following equations:

$$\sum \mu_{ijk} x_i x_j x_k + \sum \lambda_{ij} x_i x_j + \sum \nu_{ij} x_i y_j + \sum \xi_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^3 + XY + Y^2]$$

- $[X^3 + Y^3]$ are the following equations:

$$\sum \mu_{ijk} y_i y_j y_k + \sum \nu_{ijk} x_i x_j x_k + \sum \lambda_{ij} y_i y_j + \sum \varphi_{ij} x_i x_j$$
$$+ \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^3 + Y^3]$$

- $[X^3 + Y^3 + XY]$ are the following equations:

$$\sum \mu_{ijk} y_i y_j y_k + \sum \nu_{ijk} x_i x_j x_k + \sum \lambda_{ij} y_i y_j + \sum \gamma_{ij} x_i y_j$$
$$+ \sum \varphi_{ij} x_i x_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^3 + Y^3 + XY]$$

- $[X^3 + XY^2]$ are the following equations:

$$\sum \mu_{ijk} x_i y_j y_k + \sum \nu_{ijk} x_i x_j x_k + \sum \lambda_{ij} x_i x_j + \sum \gamma_{ij} x_i y_j$$
$$+ \sum \varphi_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^3 + XY^2]$$

- $[X^3 + X^2Y]$ are the following equations:

$$\sum \mu_{ijk} x_i x_j y_k + \sum \nu_{ijk} x_i x_j x_k + \sum \lambda_{ij} x_i x_j + \sum \gamma_{ij} x_i y_j$$
$$+ \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^3 + X^2Y]$$

- $[X^3 + X^2Y + Y^2]$ are the following equations:

$$\sum \mu_{ijk} x_i x_j y_k + \sum \nu_{ijk} x_i x_j x_k + \sum \lambda_{ij} x_i x_j + \sum \gamma_{ij} x_i y_j$$
$$+ \sum \varphi_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^3 + X^2Y + Y^2]$$

- $[X^3 + XY^2 + Y^3]$ are the following equations:

$$\sum \mu_{ijk} y_i y_j y_k + \sum \nu_{ijk} x_i y_j y_k + \sum \xi_{ijk} x_i x_j x_k + \sum \lambda_{ij} y_i y_j$$
$$+ \sum \gamma_{ij} x_i y_j + \sum \varphi_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^3 + XY^2 + Y^3]$$

- $[X^3 + X^2Y + Y^3]$ are the following equations:

$$\sum \mu_{ijk} y_i y_j y_k + \sum \nu_{ijk} x_i x_j y_k + \sum \xi_{ijk} x_i x_j x_k + \sum \lambda_{ij} y_i y_j$$
$$+ \sum \gamma_{ij} x_i y_j + \sum \varphi_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^3 + X^2Y + Y^3]$$

- $[X^3 + X^2Y + XY^2]$ are the following equations:

$$\sum \mu_{ijk} x_i y_j y_k + \sum \nu_{ijk} x_i x_j y_k + \sum \xi_{ijk} x_i x_j x_k + \sum \lambda_{ij} y_i y_j$$
$$+ \sum \gamma_{ij} x_i y_j + \sum \varphi_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^3 + X^2Y + XY^2]$$

- Finally, $[X^3 + X^2Y + XY^2 + Y^3]$ are the following equations:

$$\sum \mu_{ijk} x_i x_j x_k + \sum \nu_{ijk} x_i y_j y_k + \sum \xi_{ijk} x_i x_j y_k + \sum \varphi_{ijk} y_i y_j y_k + \sum \lambda_{ij} x_i x_j$$
$$+ \sum \gamma_{ij} x_i y_j + \sum \zeta_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^3 + X^2Y + XY^2 + Y^3]$$

9.2 Simulations with $n = 17$

For $n = 17$ and $K = \mathbf{F}_2$ we have made some “Toy simulations” in order to see if the attacks given in [14] could also occur in the basic HFE and to test if the attacks of sections 7 and 8 are efficient or not on some explicit polynomials f. More precisely we have computed the exact number of independent equations $[XY]$, $[XY^2]$, $[X^2Y]$ and $[X^2Y + XY^2]$, in order to compare the values obtained for various polynomials f from the values given for random quadratic functions (with no trapdoor).
$[XY]$ was chosen because the cryptanalysis of the Matsumoto-Imai C^* algorithm is mainly based on these equations $[XY]$.
$[XY^2]$ was chosen because these equations are affine in x, so that each (non-trivial) equation $[XY^2]$ gives some information on x.
$[X^2Y]$ was chosen because equations $[X^2Y]$ are often critical in some variations of the C^* algorithm (as in the two rounds of C^*, or as in the C^{*-+} algorithm studied in [17]).
$[X^2Y + XY^2]$ was chosen in order to see if we obtain more equations or not that $[XY^2]$ with these equations.
The results are given in table 1 below (this table was computed by Nicolas Courtois). The value $[\alpha]$ means that we obtain a vector space of dimension α when a random explicit value is given for y.

$f(x)$	$[XY]$	$[XY^2]$ (-306)	$[X^2Y]$ (-476)	$[X^2Y+XY^2]$ (-918)	Family
x^3	34 [16]	612 [16]	578 [153]	1428 [153]	C^*
x^5	17 [16]	340 [16]	442 [153]	1275 [153]	C^*
x^5+x^3+x	0 [0]	102 [16]	289 [135]	918 [135]	Dickson
$x^9+x^6+x^5+x^3+x$	0 [0]	0 [0]	187 [117]	561 [117]	Non bijective
$x^{12}+x^5+x^3$	1 [1]	18 [1]	170 [131]	527 [148]	Non bijective
$x^{12}+x^9+x^6+x^5+x$	0 [0]	0 [0]	170 [117]	527 [117]	Non bijective
$x^{17}+x^9+x^4+x^3+x^2+x$	0 [0]	0 [0]	153 [136]	697 [136]	Non bijective
$x^{17}+x^{16}+x^2+x$	0 [0]	0 [0]	357 [153]	1156 [153]	Non bijective
$x^{17}+x^{16}+x^5+x$	0 [0]	0 [0]	187 [153]	731 [153]	Non bijective
$x^{24}+x^{10}+x^5$	0 [0]	0 [0]	119 [134]	442 [153]	Non bijective
$x^{66}+x^{34}+x^{24}+x^6+x$	0 [0]	0 [0]	1 [18]	1 [18]	Non bijective
$x^{129}+x^3+x$	0 [0]	0 [0]	205 [137]	749 [137]	Dobbertin
f_1 (degree 17)	0 [0]	0 [0]	68 [85]	136 [114]	Non bijective
f_2 (degree 24)	0 [0]	0 [0]	68 [85]	119 [132]	Non bijective
f_3 (degree 33)	0 [0]	0 [0]	0 [17]	0 [17]	Non bijective
g_1 (degree 17)	0 [0]	0 [0]	170 [151]	714 [151]	Non bijective
g_2 (degree 32)	0 [0]	0 [0]	153 [136]	697 [136]	Non bijective
g_3 (degree 128)	0 [0]	0 [0]	17 [34]	306 [34]	Non bijective
g_4 (degree 257)	0 [0]	0 [0]	0 [17]	0 [17]	Non bijective
h_1 (degree 17)	0 [0]	0 [0]	323 [150]	1156 [150]	Non bijective
h_2 (degree 32)	0 [0]	0 [0]	323 [151]	1156 [151]	Non bijective
h_3 (degree 272)	0 [0]	0 [0]	153 [136]	731 [151]	Non bijective
h_4 (degree 4352)	0 [0]	0 [0]	17 [34]	323 [51]	Non bijective
h_5 (degree 65537)	0 [0]	0 [0]	0 [17]	0 [17]	Non bijective
Random quadratic	0 [0]	0 [0]	0 [17]	0 [17]	No trapdoor

Table 1 with $n = 17$.

- f_1 is a random polynomial of degree 17 with Hamming Weight two in x over $\mathbf{F}_2$, *i.e.*:

  $$f_1(x) = x^{17} + \alpha_{16}x^{16} + \alpha_{12}x^{12} + \alpha_{10}x^{10} + \alpha_9x^9 + \alpha_8x^8 + \alpha_6x^6 + \alpha_5x^5 + \alpha_4x^4 + \alpha_3x^3 + \alpha_2x^2 + \alpha_1x,$$

  where the α_i are random elements of $\mathbf{F}_{2^n}$.

- f_2 is a random polynomial of degree 24 with Hamming Weight two in x over $\mathbf{F}_2$, *i.e.*:

  $$f_2(x) = x^{24} + \alpha_{20}x^{20} + \alpha_{18}x^{18} + \alpha_{17}x^{17} + \alpha_{16}x^{16} + \alpha_{12}x^{12} + \alpha_{10}x^{10}$$
  $$+\alpha_9x^9 + \alpha_8x^8 + \alpha_6x^6 + \alpha_5x^5 + \alpha_4x^4 + \alpha_3x^3 + \alpha_2x^2 + \alpha_1x,$$

  where the α_i are random elements of $\mathbf{F}_{2^n}$.

- f_3 is a random polynomial of degree 33 with Hamming Weight two in x over $\mathbf{F}_2$, *i.e.*:

  $$f_3(x) = x^{33} + \alpha_{32}x^{32} + \alpha_{24}x^{24} + \alpha_{20}x^{20} + \alpha_{18}x^{18} + \alpha_{17}x^{17} + \alpha_{16}x^{16} + \alpha_{12}x^{12}$$
  $$+\alpha_{10}x^{10} + \alpha_9x^9 + \alpha_8x^8 + \alpha_6x^6 + \alpha_5x^5 + \alpha_4x^4 + \alpha_3x^3 + \alpha_2x^2 + \alpha_1x,$$

  where the α_i are random elements of $\mathbf{F}_{2^n}$.

- g_1 is a random polynomial of degree 17 with Hamming Weight two in x over $\mathbf{F}_4$, *i.e.*:

  $$g_1(x) = x^{17} + \alpha_{16}x^{16} + \alpha_8x^8 + \alpha_5x^5 + \alpha_4x^4 + \alpha_2x^2 + \alpha_1x,$$

  where α_1, α_2, α_4, α_5, α_8, α_{16} are random elements of $\mathbf{F}_{2^n}$.

- g_2 is a random polynomial of degree 32 with Hamming weight two in x over $\mathbf{F}_4$, *i.e.* with degrees 1, 2, 4, 5, 8, 16, 17, 20, 32.

- g_3 is a random polynomial of degree 128 with Hamming weight two in x over $\mathbf{F}_4$, *i.e.* with degrees 1, 2, 4, 5, 8, 16, 17, 20, 32, 64, 65, 68, 80, 128.
- g_4 is a random polynomial of degree 257 with Hamming weight two in x over $\mathbf{F}_4$, *i.e.* with degrees 1, 2, 4, 5, 8, 16, 17, 20, 32, 64, 65, 68, 80, 128, 256, 257.
- h_1 is a random polynomial of degree 17 with Hamming Weight two in x over $\mathbf{F}_{16}$, *i.e.*:

$$h_1(x) = x^{17} + \alpha_{16}x^{16} + \alpha_2 x^2 + \alpha_1 x,$$

  where α_1, α_2, α_{16} are random elements of $\mathbf{F}_{2^n}$.
- h_2 is a random polynomial of degree 32 with Hamming Weight two in x over $\mathbf{F}_{16}$, *i.e.*:

$$h_2(x) = x^{32} + \alpha_{17}x^{17} + \alpha_{16}x^{16} + \alpha_2 x^2 + \alpha_1 x,$$

  where α_1, α_2, α_{16} and α_{17} are random elements of $\mathbf{F}_{2^n}$.
- h_3 is a random polynomial of degree 272 with Hamming Weight two in x over $\mathbf{F}_{16}$, *i.e.*:

$$h_3(x) = x^{272} + \alpha_{257}x^{257} + \alpha_{256}x^{256} + \alpha_{32}x^{32} + \alpha_{17}x^{17} + \alpha_{16}x^{16} + \alpha_2 x^2 + \alpha_1 x,$$

  where the α_i are random elements of $\mathbf{F}_{2^n}$.
- h_4 is a random polynomial of degree 4352 with Hamming Weight two in x over $\mathbf{F}_{16}$, *i.e.* with degrees 1, 2, 16, 17, 32, 256, 257, 272, 512, 4096, 4097, 4112, 4352.
- h_5 is a random polynomial of degree 65537 with Hamming Weight two in x over $\mathbf{F}_{16}$, *i.e.* with degrees 1, 2, 16, 17, 32, 256, 257, 272, 512, 4096, 4097, 4112, 4352, 8192, 65536, 65537.
- "Random quadratic" stands for 17 randomly chosen quadratic equations (with no trapdoor).

Interpretation of the results

For a well chosen polynomial $f(x)$, equations $[XY]$ and $[XY^2]$ do not give any attack.
The number of equations $[X^2Y]$ is a bit higher than expected. To avoid attacks based on $[X^2Y]$, a degree ≥ 33 is recommended.
The equations $[X^2Y + XY^2]$ do not give better attacks than $[X^2Y]$, except for very special polynomials.

9.3 Simulations with $n = 22$

The examples above were also studied with $n = 22$. The results are given in table 2 below (this table was also computed by Nicolas Courtois).

$f(x)$	$[XY]$	$[XY^2]$ (-506)	$[X^2Y]$ (-781)	$[X^2Y+XY^2]$ (-1518)	Family
x^3	44 [20]	1012 [20]	968 [251]	2398 [251]	C^*
x^5	22 [20]	550 [20]	682 [253]	2090 [253]	C^*
x^9	22 [20]	528 [20]	726 [251]	2112 [251]	C^*
x^{17}	22 [20]	528 [20]	748 [253]	2134 [253]	C^*
x^{33}	22 [20]	550 [20]	748 [251]	2134 [251]	C^*
x^5+x^3+x	0 [0]	132 [18]	374 [172]	1408 [172]	Dickson
$x^9+x^6+x^5+x^3+x$	0 [0]	0 [0]	242 [150]	836 [151]	Non bijective
$x^{12}+x^5+x^3$	1 [1]	23 [1]	220 [174]	792 [192]	Non bijective
$x^{12}+x^9+x^6+x^5+x$	0 [0]	0 [0]	220 [150]	792 [151]	Non bijective
$x^{17}+x^9+x^4+x^3+x^2+x$	0 [0]	0 [0]	198 [176]	1122 [176]	Non bijective
$x^{17}+x^{16}+x^2+x$	0 [0]	0 [0]	418 [198]	1782 [198]	Non bijective
$x^{17}+x^{16}+x^5+x$	0 [0]	0 [0]	220 [192]	1144 [193]	Non bijective
$x^{24}+x^{10}+x^5$	0 [0]	0 [0]	132 [152]	738 [169]	Non bijective
$x^{66}+x^{34}+x^{24}+x^6+x$	0 [0]	0 [0]	1 [23]	1 [23]	Non bijective
$x^{129}+x^3+x$	0 [0]	0 [0]	264 [176]	1188 [176]	Non bijective
f_1 (degree 17)	0 [0]	0 [0]	88 [110]	176 [149]	Non bijective
f_2 (degree 24)	0 [0]	0 [0]	88 [110]	154 [174]	Non bijective
f_3 (degree 33)	0 [0]	0 [0]	0 [22]	0 [22]	Non bijective
g_1 (degree 17)	0 [0]	0 [0]	176 [154]	1100 [154]	Non bijective
g_2 (degree 32)	0 [0]	0 [0]	132 [110]	1056 [110]	Non bijective
g_3 (degree 128)	0 [0]	0 [0]	22 [44]	506 [44]	Non bijective
g_4 (degree 257)	0 [0]	0 [0]	0 [22]	0 [22]	Non bijective
h_1 (degree 17)	0 [0]	0 [0]	352 [198]	1760 [198]	Non bijective
h_2 (degree 32)	0 [0]	0 [0]	286 [154]	1694 [154]	Non bijective
h_3 (degree 272)	0 [0]	0 [0]	88 [66]	1012 [66]	Non bijective
h_4 (degree 4352)	0 [0]	0 [0]	22 [44]	506 [44]	Non bijective
h_5 (degree 65537)	0 [0]	0 [0]	0 [22]	0 [22]	Non bijective
Random quadratic	0 [0]	0 [0]	0 [22]	0 [22]	No trapdoor

Table 2 with $n = 22$.

Interpretation of the results

For a well chosen polynomial $f(x)$, equations $[XY]$ and $[XY^2]$ do not give any attack. Similarly, for well chosen polynomials $f(x)$, the number of equations $[X^2Y]$ will be similar to the number obtained with truly random quadratic functions (with no trapdoor). However, this condition on $[X^2Y]$ is more restrictive than the similar condition on $[XY^2]$.

Remark: To see how some equations $[X^2Y]$ or $[X^2Y+XY^2]$ can be useful for an attack, see section 8.

10 Examples of attacks that do not work

In [15] I have presented some attacks that work very well against some asymmetric cryptosystems with multivariate polynomials (and a hidden monomial). So a natural question is: do these attacks also work against HFE ? In this section, we will see why it seems that these attacks do not work against HFE.

10.1 First example of an attack that does not work

For the cryptanalysis of the basic HFE, we can think to do this attack in 5 steps. (We will see below that this attack do not really work.)

Step 1. We compute the vector space of all the equations

$$\sum \gamma_{ijk} x_i x_j x_k + \sum \mu_{ij} x_i x_j + \sum \nu_{ij} x_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0 \qquad [X^3 + XY]$$

Step 2. We isolate the terms in $x_i y_j$. We have like this some expressions $d_1, \ldots, d_k$.
Step 3. We compute the vector space of the linear transformations C and D such that the transformation C on $d = (d_1, ..., d_k)$ has the same effect as the transformation D on x. (We give more details about this step 3 below).
Step 4. Similarly, we compute the vector space of the linear transformations C' and D' such that the transformations C' on d has the same effect as the transformation D' on y.
Step 5. From step 3, the idea of the cryptanalysis is to find the analogy of a multiplication on the y_i variables, and from step 4 the analogy of a multiplication on the x_i variables.

Now from these multiplications we may find the analogy of the secret function f and we may be able to compute x from y with this analogy.

Example.

Let $b = f(a) = a^{17} + a^{16} + a^5 + a$ be the secret and hidden equation.
Let $B = a^5$.
Then $a^{16}.B = a.B^4$. (1).
Moreover, $B = b - a^{17} - a^{16} - a$. (2).
So from (1) and (2) we have: $a^{16}(b - a^{17} - a^{16} - a) = a(a^5)^4$. (3).
This equation (3) will be in the space found in step 1 (*i.e.* it will give some equations $[X^3 + XY]$).
Moreover the terms in $x_i y_j$ in this equation (3) come from $a^{16}.b$, and for each $\lambda \in K$ we have $(\lambda a^{16}).b = a^{16}(\lambda b) = \lambda(a^{16}.b)$ (4).
From (4), can we say that steps 3, 4 and 5 will succeed, *i.e.* that we will find the vector space for (C, D) and (C', D') and that these vector spaces will be of dimension n (because we have n degree of liberty for λ) ?
No. Because in step 1 equation (3) will be mixed with all the multiplication of the n public equations by each $x_i, 1 \leq i \leq n$. So for (C, D) we will have much more solutions than only the multiplication by an element λ of K.

Remark. More precisely, in step 1 we will find exactly the vector space (of dimension $n(n+1)$) generated from the n original public equations, and these equations multiplied by x_i, and nothing else. (This is because we will find at least this vector space of dimension $n(n+1)$, and exactly this vector space, because if we had more equations, then by Gaussian reductions, we would have a polynomial P such that $P(x_1, ..., x_n)$ for any $(x_1, ..., x_n)$. But this implies $P = 0$). As a result, no information for an attack can be obtained from equations $[X^3 + XY]$ (unlike equations $[XY]$, $[XY^2]$, $[X^2Y]$ of the above section, where the dimensions of the vector spaces were different for random quadratic functions of for C^* or some special functions).

10.2 Another attack that does not work

Example: Let $b = f(a) = a^{17} + a^5 + a$. When $a^{17} \mapsto \lambda a^{17}$, and $a^5 \mapsto \lambda a^5$, and $a \mapsto \lambda a$, then $b \mapsto \lambda b$. So we may try to find some linear transformations (C, D), such that the effect of C on the "quadratic expressions in x_i in the public equations" is the same as the effect of D on the values y_1, ..., y_n.
In [15], a similar attack is done, and works very well against some asymmetric cryptosystems, but here it does not work because the "quadratic expressions in x_i in the public equations" are exactly y_1, ..., y_n, so we will find all (C, D) solutions where $D = C$, and where C is any linear transformation (not only the transformations "multiplication by λ").

Part III: HFE variations

11 Three simple variations

11.1 Less public polynomials

The polynomials $(P_1, \ldots, P_n)$ of the "basic" HFE algorithm give y from x. However, it is possible to keep secret some of these polynomials. 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 2^{km} possibilities for y, compute all the corresponding possible x, and find the good x thanks to the redundancy.

  When m is very small, for example when $m = 1$ or 2, and when k is very small, for example with $k = 1$ or 2, this is clearly feasible.

- In a signature scheme, k may be much larger. However, we must still have enough polynomials 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 secret some polynomials P_i may increase, or not, the security of some schemes. In Part IV, we will study the cryptanalytic effects of this idea on the original C^* scheme.

11.2 Introducing some random polynomials

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 $\frac{1}{2^{km}}$. When m and k are small, the scheme is efficient: after about 2^{km} 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.

Note 2: Of course, it is possible to combine the variations of section 10.1 and 10.2. 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.

11.3 Introducing more x_i variables

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 $a' = (a'_1, \ldots, a'_k)$ 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 functions 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.
Then the public key is given as the n equations $y_i = P_i(x_1, ..., x_{n+k})$.
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: This ideas, as before, may or may not increase the security of some schemes.

12 HFE with a multivariate hidden equation

In this section another variation of the HFE algorithm will be shown.

Here the idea is to change the description of the function f given in section 2.1.
We can notice that what we need for f is that:

1. In a basis, f is a multivariate quadratic function.
2. For any value a, it is easy to find all the x so that $f(x) = a$.
3. f is a function $K^n \to K^n$ with numbers in K^n with at least 64 bits.

The solution given in part I was to choose for f a polynomial in **only one** variable x over K^n so that, in a basis, f is a multivariate quadratic function.
However we can imagine a lot of different solutions than that for f.
For example we can imagine that f is a polynomial in two variables, x_1 and x_2, so that:

1. In a basis, the expression of f is still a multivariate polynomial of total degree two.
2. For any a, it is possible to find all the x_1, x_2 so that $f(x_1, x_2) = a$.

Example

For example let us assume that $n = 128$ and $q = p = 2$, so $K = \mathbf{F}_2$.

$$\text{Let } f: \begin{array}{lcl} K^{64} \times K^{64} & \to & K^{64} \times K^{64} \\ (x_1, x_2) & \mapsto & (y_1, y_2) \end{array}$$

such that:

$$\begin{array}{ll} y_1 = x_1^4 + x_1 x_2 + x_2 & (1) \\ y_2 = x_1^{17} + x_1^4 x_2 + x_2^3 & (2) \end{array}$$

(This is just an example. We do not recommend this specified function f).
Then in order to find (x_1, x_2) from (y_1, y_2) we can proceed like this:
from (1) we have:

$$x_2 = \frac{y_1 - x_1^4}{x_1 + 1} \quad (3)$$

Then from (2) we have:

$$y_2(x_1+1)^3 = x_1^{17}(x_1+1)^3 + x_1^4(y_1 - x_1^4)(x_1+1)^2 + (y_1 - x_1^4)^3. \quad (4)$$

Now we can notice that (4) is a polynomial equation with only one variable x_1 in a finite field. So it is possible to find x_1, and then from (3) we will find x_2.

An advantage of such a scheme is that the secret computations may be easier than the secret computations of the Basic HFE, because in the equation (4) the variable x_1 has only $n/2$ bits instead of n bits for the Basic HFE.

More general cases

More generally, polynomials with 3, 4 or more variables can be used so that an algorithm of inversion of f exists and the expression of f in a basis is quadratic.
Moreover a lot of different algorithms are known to find the roots of specific multivariate polynomials in a finite field (for example Grobner basis, or specific algorithms in Algebric Geometry) and each of this algorithm could lead to a specific HFE algorithm.

13 HFE with more than one branch, and HRE: Hidden Ring Equations

13.1 HFE with more than one branch

In analogy with the Matsumoto-Imai algorithm of [11] we can imagine a HFE algorithm with more than one branch in order to have easier secret computations.
Let d be the number of branchs. The algorithm proceeds like this:

1. The first affine transformation s is performed on x obtaining $a = s(x)$.
2. The value a obtained is splitted in d "branchs", $a = a_1||a_2 \ldots ||a_d$ (where $||$ is the concatenation function).
3. Then $b_1 = f_1(a_1), \ldots, b_d = f_d(a_d)$ are computed, where $f_1, \ldots, f_d$ are d functions as in section 2.1.
4. The last affine transformation t is performed on $b = b_1|| \ldots ||b_d$, obtaining $y = t(b)$.

If we could have very short branchs (for example if we could have branchs which manipulate values a_i of less than 16 bits) then the algorithm would be really very efficient. However it can be proved that for the security of the scheme each branch must manipulate values a_i of at least 64 bits. We do not give too many details here because in the paper [16] written with Louis Goubin, we give a detailed analysis of all the cryptanalysis of short branchs that we have found.
So for 128 bit messages for example there is no more than 2 branches. So there cannot be a lot of small branches, so the main practical interests of more than one branch have disappeared.
So in conclusion we do not recommend to use more than one branch in HFE.

13.2 HRE: Hidden Rings Equations

In section 2.1 we said that the field L_N is typically $K[X]/(i_N(X))$, where $i_N(X)$ is an irreductible polynomial over K.
If $i_N(X)$ is not irreductible, then $L'_N = K[X]/(i_N(X))$ will then not be a finite field, but a finite ring. In such a space the resolution of $f(x) = y$, where f is a univariate polynomial is still feasible. For example the linearized polynomial algorithm still works.
So we can design an asymmetric scheme in such a space exactly as HFE in the finite field L_N.
We can call such an algorithm HRE for Hidden Rings Equations. However it seems that we obtain no advantage to use HRE instead HFE.

14 HFE with s and t with values in a subfield

In order to have a smaller value for the length of the public key, an idea is to have a public key with all the values of the coefficients in a subfield k of $K = \mathbf{F}_q = \mathbf{F}_{p^m}$. In order to achieve this:

1. The values of the matrices of s and t will be chosen if k.
2. The irreducible polynomial φ such that $\mathbf{F}_{q^n} \simeq \mathbf{F}_q[X]/\varphi(X)$will be chosen in $k[X]$.
3. The polynomial f will also be chosen in $k[X]$.

To obtain condition (2), we will choose n such that $GCD(n,m) = 1$ because it is possible to prove that is φ is a polynomial of $\mathbf{F}_p[X]$ of degree n, irreducible on $\mathbf{F}_p$ and if $GCD(n,m) = 1$, then φ is also irreducible over $\mathbf{F}_{p^m}$. At the present, it is not clear whether restricting HFE with conditions (1), (2) and (3) is dangerous for its security or not.

Example: Let $k = \mathbf{F}_2$, $K = \mathbf{F}_{256}$, $n = 29$. Then the length of the public key is $29 \cdot (\frac{29 \cdot 30}{2}) \cdot (\frac{1}{8}) = 1.54$ Kbytes (instead of 12.3 Kbytes if $k = K = \mathbf{F}_{256}$).

Remark: Since here $f \in k[X]$, f will commute with all the Frobenius functions of $\mathbf{F}_{q^n}[X]$ that are invariant over k. However, it is not clear whether such a property can be useful for a cryptanalysis.

15 Concatenation of two basic HFE or HRE for fast decryptions

The scheme

Let x be the cleartext. Let $y_1 = HFE_1(x)$ be the encryption of x with a first HFE encryption with secret affine functions s_1 and t_1.
Let $y_2 = HFE_2(x)$ be the encryption of x with another HFE, such that HFE_1 and HFE_2 have different polynomials f_1 and f_2 and independent secret affine functions t_1 and t_2, but the same extension field L_N, and the same secret affines functions $s_1 = s_2$.
Then let $y_1||y_2$ be the encryption of x, where $||$ is the concatenation function.

Speed of decryption

The main advantage of this scheme is that decryption with the secret keys may be very fast, as we will see now. From y_1 and y_2, $f_1(a)$ and $f_2(a)$ will be obtained, and then $GCD(f_1(a), f_2(a))$ will be computed. Then from this GCD the value of a will be obtained with one of the classical algorithm of resolution of equation (like in section 5). Then $x = s_1^{-1}(a)$ will be obtained.
In average the time of computation of $GCD(f_1(a), f_2(a))$ is expected to be dominant.
This time is $\leq \mathcal{O}(d^2 n^2)$, where $d = \sup(d_1, d_2)$. So if d_1 and d_2 are not too large decryption will really be very fast (and much faster than in the basic HFE).

Asymptotic speed

From a theoretical point of view, when n is very large $d = \mathcal{O}(\ln n)$ is expected to be sufficient to avoid all the polynomials-time attacks, for well chosen polynomials f_1 and f_2 with degree $(f_1) \leq d$ and degree $(f_2) \leq d$.
Moreover when n is very large a multiplication in L_n requieres only $\mathcal{O}(n \ln n)$ computations (and not $\mathcal{O}(n^2)$).
So the complexity of decryption with the secret key of the scheme of this section 10 will be $\leq 0(n \ln^3 n)$ asymptoticaly with well chosen f_1 and f_2.
This is really very fast (for example the decryption with the secrets of RSA and of most public key scheme is asymptoticaly $\geq 0(n^2 \ln n)$.

Security

In this scheme we have only encrypt x with two different but not independent HFE to recover x from $y_1||y_2$ (these two HFE are not independents since $s_1 = s_2$).
However it is not recommended generally in cryptography to encrypt the same message twice by two different encryptions. Moreover this is generally particulary not recommended when the two encryptions are not independents. So if this algorithm is really used we recommend to be extra-careful in the choice of the polynomials f_1 and f_2. For example not only f_1 and f_2 should avoid the "Affine multiple attack" of Part II, section 7, but also $f_1 + f_2$.
However if great care are done in the selection of f_1 and f_2 it seems that this scheme may be secure.

Variations

Of course, instead of a concatenation of two HFE, which gives $y_1||y_2$, we can also choose an output which is an affine and bijective secret transformation of $y_1||y_2$. (However this do not change a lot of thing both for secret key computations and from a cryptanalitic point of view).

16 HFE with public polynomials of degree ≥ 3

Of course we can also choose for f a polynomial with some exponents in x of Hamming Weight still small but ≥ 3.
A very important subcase is when this function is $f(x) = x^{1+2^\theta+2^\varphi}$, *i.e.* with only one monomial and Hamming Weight 3. The study of these functions is one of the main subject of [15].

17 The "C^{*-}" algorithm

In order to repair the Matsumoto-Imai scheme we have suggested three transformations:

1. To have only one branch (cf section 11).
2. To change the function f in order to avoid the "affine multiple attack" (cf section 7).
3. To keep secret some polynomials P_i.

We have suggested to do these three transformations. However in the "Basic HFE" just 1 and 2 are done and the "Basic HFE" may be secure despite the fact that 3 is not done.
We can also ask ourselves what happens if we just do the transformations 1 and 3, and not the transformation 2, *i.e.* if $f(x) = x^{1+2^\theta}$, as in the Matsumoto-Imai algorithm. The main interest of doing that is that with such function f the computation of f^{-1} is easier. So the secret key computations may be easier.
We call this candidate algorithm C^{*-} because we have just removed a few equations compared to the original C^* algorithm. *A priori*, C^{*-} does not seem to be a very strong way to repair the C^* scheme ! The analysis of C^{*-} (and other variations around C^*) is done in the paper [17]. In this paper [17], we show that when the number r of equations removed is small (typically when $q^r \leq 2^{40}$), then the scheme is still insecure, which is not too surprising. However, when $q^r \geq 2^{64}$, our cryptanalysis of [17] 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 !

Part IV: IP: Isomorphisms of Polynomials

18 IP with two secrets s and t

Introduction

We will now present a new authentication and signature scheme called "Isomorphism of Polynomials" (IP).
IP has a few nice properties:

- We know exactly the problem on which the security of the scheme relies (it is the problem of finding an isomorphism of quadratic or cubic equations).
- The scheme is very symmetric, and the design of the scheme is very similar to the well known "Graph Isomorphism Authentication scheme" (cf [18] for example).
- IP authentications are proved to be zero-knowledge.
- No hash functions are needed for the authentications.
- IP illustrates the fact that if in HFE the function f is public the HFE scheme may be still secure.

Moreover IP is not only a scheme of theoretical interest: as we will see, with some well chosen parameters it is an efficient algorithm for authentications and signatures. (However, unlike HFE, it seems that IP can not be used for encryptions).

Note: In this part V, we will present IP and how to use it in order to design some cryptosystems for authentication and signature. However, the currently best known algorithms for IP are not presented here: they are presented in [6]. These algorithms are not polynomial, but are much more efficient than exhaustive search.

The IP problem with two secrets s and t

Let u and n be two integers (with $u \neq 1$). Let K be a finite field.
Let A be a public set of u quadratic equations with n variables $x_1, \ldots, x_n$ over the field K. We can write all these equations like this:

$$y_k = \sum_i \sum_j \gamma_{ijk} x_i x_j + \sum_i \mu_{ik} x_i + \delta_k, \quad \text{for } k = 1, \ldots u \tag{1}$$

Now let s be a bijective and affine transformation of the variables $x_i, 1 \leq i \leq n$, and let t be a bijective and affine transformation of the variables $y_k, 1 \leq k \leq u$.
Let $s(x_1, \ldots, x_n) = (x'_1, \ldots, x'_n)$, and $t(y_1, \ldots, y_u) = (y'_1, \ldots, y'_u)$.
From (1) we will obtain k equations that gives the y'_k values from the x'_i values like this:

$$y'_k = \sum_i \sum_j \gamma'_{ijk} x'_i x'_j + \sum_i \mu'_{ik} x'_i + \delta'_k, \quad \text{for } k = 1, \ldots u. \tag{2}$$

Let B be the set of these u equations.
We will say that A and B are "isomorphic", *i.e.* there is a double bijective and affine transformation that gives B from A. And we will say that (s, t) is an "isomorphism" from A to B.
The "Isomorphism of Polynomials Problem" is this problem: when A and B are two public sets of a u quadratic equations, and if A and B are isomorphs, find an isomorphism (s, t) from A to B.

Example. If $u = n$ no polynomial algorithms to solve this problem are known. If such an algorithm were found then it would give us a way to find the keys of the Matsumoto-Imai algorithm (and not only find a way to decrypt most of the messages). So it would give us a new, and more powerful, attack on the Matsumoto-Imai algorithm. Moreover if such an algorithm were found then in HFE it would be essential for security to keep f secret. On the contrary as long as no such algorithm is found HFE may be still secure if f is public.

Note. We could think to proceed like this in order to find s and t: to introduce the matrix of s and t values and to formaly identify the equations (1) and (2). However we will obtain like this some equations of total degree two in the values of s and t and the general problem of solving equations of degree $\geq$ two in a finite field is NP hard. So this idea does not work.

The IP authentication scheme with two secrets s and t

Public: Two isomorphic sets A and B of u quadratic equations with n variables over a field K.
Secret: An isomorphism (s, t) from A to B.

Notations The equations of A are the equations (1) of section 2, they give the y_k values from x_i values, and the equations of B are the equations (2) of section 2, they give the y'_k values from the x'_i values.
Let us assume that Alice knows the secret (s, t) and that Alice wants to convince Bob of this knowledge, without revealing her secret. Alice and Bob will follow this protocol:

Step 1. Alice randomly computes a set C of equations isomorph to A.
For this, she randomly computes an affine bijection s' of the values $x_i, 1 \leq i \leq n$, and an affine bijection t' of the variables $y_k, 1 \leq k \leq u$.
The u equations of C are u equations like this:

$$y''_k = \sum_i \sum_j \gamma''_{ijk} x''_i x''_j + \sum_i \mu''_{ik} x''_i + \delta''_k, \quad \text{for } k = 1, \ldots u. \tag{3}$$

- s gives the transformation $x \to x'$.
- t gives the transformation $y \to y'$.
- s' gives the transformation $x \to x''$.
- t' gives the transformation $y \to y''$.

Step 2. Alice gives the set C of equations (3) to Bob.
Step 3. Bob asks Alice either to

(a) Prove that A and C are isomorphic.

(b) Prove that B and C are isomorphic.

Moreover Bob choose to ask (a) or (b) randomly with the same probability 1/2.

Step 4. Alice complies.
If Bob ask (a), then she reveals s' and t'.
If Bob ask (b), then she reveals $s' \circ s^{-1}$ and $t' \circ t^{-1}$ (*i.e.* the transformations $x' \to x''$ and $y' \to y''$).

It is easy to prove that this protocol is zero-knowledge and that if somebody doesn't know an isomorphism (s, t) from A to B the probability to successfully pass the protocol is at most 1/2.
So if Alice and Bob repeat steps (1) to (4) N times, the probability of success will be at most $1/2^N$.

Parameters

Analogous to the problem of finding the secret affine transformations s and t of HFE when f is public or of the Matsumoto-Imai algorithm, we could have $u = n = 64$ and $K = \mathbf{F}_2$ for example in a IP authentication scheme or $u = n = 16$ and $K = \mathbf{F}_{16}$. However more practical values may be sufficient for security.
For example $u = 64$ and $n = 16$ may be sufficient with $K = \mathbf{F}_2$. (In HFE $n \leq 40$ and $K = \mathbf{F}_2$ will not be sufficient because we could find a cleartext by exhaustive search in 2^{40} computations, but here we have to find a key (s, t) and this is expected to be more difficult than to find a cleartext from a ciphertext).

Honestly it is not actually completely clear what minimal values for u and n could be chosen for security. (Below, we will suggest some explicit values). We have not spend enough time on algorithms to solve IP to have a clear view of the security values. However we will see in section 18 some algorithms much better than exhaustive search on s.

Less computations with larger public keys

Instead of only two sets of public isomorphic equations (A) and (B), less us now assume that we have k sets of public isomorphic equations $(P_1), (P_2), \ldots (P_k)$.
We denote by $x_i^{(1)}$ the variables of $(P_1), \ldots$ and by $x_i^{(k)}$ the variables of (P_k). And we denote by s_j the secret affine transformation from $x^{(1)}$ to $x^{(j)}$, $2 \le j \le k$. (So all the k equations are isomorphic to (P_1), so each couple of these equations are isomorphic).
Of course we can assume if we want that all the secret affine transformations $s_j, 2 \le j \le k$, are computed from one small secret K, for example K is a secret DES key and the matrix of the s_j are obtained by some computations of DES_K.
So the public key is larger, since we have k equations (P_j), but the secret key can be still small.
The authentication now proceed like this:

Step 1. Alice randomly computes, as usual, one equation C isomorph to P_1.

Step 2. Alice gives this equation C to Bob.

Step 3. Bob randomly chose a value u, $1 \le u \le k$, and ask Alice to prove that C and P_u are isomorphic.

Step 4. Alice complies.

It is still easy to prove that this protocol is zero-knowledge and that if somebody doesn't know any isomorphism s from one (P_i) to one (P_j), $i \neq j$, then the probability to successfully pass the protocol is at most $1/k$.
So if Alice and Bob repeat steps (1) to (4) N times, the probability of success will be at most $1/k^N$.

19 IP with one secret s

The IP problem with one secret s

Let n be an integer. Let K be a finite field.
Let A be one public cubic equation with n variables $x_1, \ldots, x_n$ over the field K. (Here in A we have only one equation, but of degree 3 and not 2). We can write this equation (A) like this:

$$\sum\sum\sum \gamma_{ijk} x_i x_j x_k + \sum\sum \mu_{ij} x_i x_j + \sum \alpha_i x_i + \delta_0 = 0. \qquad (A).$$

Now let s be a bijective and affine transformation of the variables $x_i, 1 \le i \le n$.
Let $s(x_1, \ldots, x_n) = (x'_1, \ldots, x'_n)$.
From (A) we will obtain one equation (B) in x'_i like this:

$$\sum\sum\sum \gamma'_{ijk} x'_i x'_j x'_k + \sum\sum \mu'_{ij} x'_i x'_j + \sum \alpha'_i x'_i + \delta_0 = 0 \qquad (B).$$

We will say that (A) and (B) are "isomorphic", *i.e.* there is a bijective and affine transformation that gives B from A. And we will say that s is an "isomorphism" from A to B.
The "Isomorphism of Polynomials Problem" is now this problem: when A and B are two public sets of such equations, and if A and B are isomorphs, find an isomorphism s from A to B.

Note. For equations of total degree ≥ 3, we know no polynomial algorithm to solve this IP problem. However, as we mentioned in note 1 above, for equations of total degree 2 there is a polynomial algorithm to solve the problem, because there is a "canonical" representation of each quadratic equations over a finite field (cf [10], chapter 6 for example). If (A) and (B) are isomorphs, then the two canonical representations of (A) and (B) will be easilly found, will be the same, and this will give the isomorphism from (A) to (B). This is the reason why we have chosen equations of degree ≥ 3. (Another possibility is to have at least two equations in (A) and (B), as we did above, but still with only one secret s: we do not use t).

The IP authentication scheme with one secret s

Public: Two equations A and B of total degree 3 with n variables over a field K (or a set A of at least two equations of total degree 2 with n variables over a field K, and a similar set B).
Secret: An isomorphism s from A to B.

Notations. We denote by $x_i, 1 \leq i \leq n$, the n variables of A, and by $x'_i, 1 \leq i \leq n$, the n variables of B.

Step 1. Alice randomly computes one equation C isomorph to A. For this she randomly computes an affine bijection s' of the values $x_i, 1 \leq i \leq n$.

- s gives the transformation $x \to x'$.
- s' gives the transformation $x \to x''$.

And C is the equation obtained from A with the x''_i variables instead of the x_i variables.

Step 2. Alice gives the equation C to Bob.

Step 3. Bob asks Alice either to

(a) Prove that A and C are isomorphic.

(b) Prove that B and C are isomorphic.

Moreover Bob choose to ask (a) or (b) randomly with the same probability 1/2.

Step 4. Alice complies.
If Bob ask (a), then she reveals s'.
If Bob ask (b), then she reveals $s' \circ s^{-1}$ (*i.e.* the transformation $x' \to x''$).

It is easy to prove that this protocol is zero-knowledge and that if somebody doesn't know an isomorphism s from A to B the probability to successfully pass the protocol is at most 1/2.
So if Alice and Bob repeat steps (1) to (4) N times, the probability of success will be at most $1/2^N$.

Remark: As in section 17.4, this algorithm can be improved if we use k public isomorphic equations: with $k > 2$, we will have a larger public key but les computations to perform.

20 IP for asymmetric signatures

The Fiat-Shamir authentication scheme and the Guillou-Quisquater authentication scheme can be transformed in signature scheme by using a now classical transformation by introducing hash function. This transformation works also very well here for the IP algorithm.
Let M be the message to sign.
The signature algorithm is this one:

Step 1. Alice randomly computes λ equations C_i isomorphs to P_1.

Step 2. Alice computes hash $(M||C_1||\ldots C_\lambda)$, where $||$ is the concatenation function, and hash a public hash function sufficiently large such that the first bits of output can give λ values $e_1, \ldots, e_\lambda$, where each e_i is a value between 1 and k.

Step 3. Alice computes the λ isomorphisms $t_i, 1 \leq i \leq \lambda$, such that each t_i is an isomorphism from C_i to P_{e_i}.

The signature on M by Alice is then (T, E) where T is the vector $(t_1, t_2, \ldots t_\lambda)$ and E is the vector $(e_1, e_2, \ldots e_\lambda)$.

To verify this signature, Bob proceeds like this:

Step 1. Bob computes $C_1, \ldots, C_\lambda$ such that $t_i, 1 \leq i \leq \lambda$ is an isomorphism from C_i to P_{e_i}.

Step 2. Bob checks that the first bits of hash $(M||C_1|| \ldots C_\lambda)$ are the entries e_i of E.

21 About the security of IP schemes

In [6], the best known algorithms for IP are presented. Moreover, in [6], some links between IP and some famous problems (such as the Graph Isomorphism of Fast Matrix Multiplication) are presented. These results suggest that there is probably no polynomial algorithm for IP. However, some algorithms presented in [6] are much more efficient than exhaustive search. The typical complexity for IP with two secrets seems to be about $\mathcal{O}(q^n)$ or even $\mathcal{O}(q^{n/2})$ (instead of $\mathcal{O}(q^{(n^2)})$ for exhaustive search).

22 Numerical examples

Numerical examples for IP authentications and signatures with one secret s

Notations: As above, k is the number of public isomorphic equations, N is the number of rounds (for authentications), λ is the number of random isomorphims used (for signatures).

In authentications

For security we must have $k^N \geq 2^{20}$ because the probability of success without the secrets is $1/k^N$ and we must have $q^{n(n+1)}$ larger than 2^{64}, where $q = |K|$, in order to avoid exhaustive search on s. More precisely, we must have $q^{n\sqrt{2}\sqrt{n}}$ larger than 2^{64} to avoid some attacks similar as those given above. So for example if $K = \mathbf{F}_2$ we suggest to have $n \geq 16$. However $n \geq 12$ might be sufficient...

In signatures

For security we must have $k^\lambda \geq 2^{64}$ (because the probability of success without the secrets is $1/k^\lambda$) and as before we must have $q^{n\sqrt{2}\sqrt{n}}$ larger than 2^{64}, where $q = |K|$ (in order to avoid exhaustive search on s). If s is linear (not only affine) then the lenght of the public key (with equations of degree three) is $k.\frac{n(n-1)(n-2)}{6} \ln_2(q)$ bits, and the lenght of the signature is $\lambda.(\ln_2(k) + n^2 \ln_2(q))$ bits.

Example 1. If $K = \mathbf{F}_2, n = 16, \lambda = 4$ and $k = 2^{16}$, then the lenght of the public key is $k.16.15.14/3!$ bits $= 4.4$ Megabytes (with one equation of degree three, or $2k.16.15/2$ bits $= 1.9$ Megabytes with two equations of degree two). This is huge but can be store in a hard disc of a Personnal Computer, and the lenght of the signature is $\simeq 4.(16 + 16.16) = 1088$ bits.

Note. Here again, instead of $n = 16$, $n = 12$ is perhaps sufficient for security. In this case the lenght of the signature will be only 576 bits.

Example 2. If $K = \mathbf{F}_2, n = 16, \lambda = 16$ and $k = 16$, then the lenght of the public key is 1120 bytes with one equation of degree three, or 480 bytes with two equations of degree two, and the lenght of the signature is $\simeq 16.(4 + 16.16) = 4128$ bits.

Example 3. If $K = \mathbf{F}_{16}, n = 6, \lambda = 16$ and $k = 16$, then the lenght of the public key is only 160 bytes with one equation of degree three, or 240 bytes with two equations of degree two, and the lenght of the signature is 2368 bits.

Example 4. If $K = \mathbf{F}_{16}, n = 6, \lambda = 4$ and $k = 2^{16}$ then the lenght of the public key is 640 Kbytes and the lenght of the signatures is 640 bits.

Numerical examples for IP with two secrets

As we said in section 2 we haven't enough results on algorithms to solve the IP problem to have a precise evaluation of the parameters needed for security. However we will suggest now some examples of parameters (smaller values may be enough for security, or larger values may be needed... it is not clear right now).
In signature, for security we must have $k^\lambda \geq 2^{64}$ (because the probability of success without the secrets is $1/k^\lambda$) and $q^{\frac{n^2+n+2u}{2}}$ "much larger" than 2^{64} and $u \neq 1$ (because of the attack seen in section 2). The words "much larger" comes to the fact that the algorithm of section 2 can probably be improved a lot. The lenght of the public key is $u.k.\frac{n(n-1)}{2}\ln_2(q)$ bits and the lenght of the signature is $\lambda.(\ln_2(k) + n^2\ln_2(q))$ bits.

Example 1. $K = \mathbf{F}_2, u = n = 64, \lambda = 16, k = 16$.
Then the lenght of the public key is $\simeq$ 256 bytes, and the lenght of the signature is $\simeq$ 65600 bits.

Example 2. $K = \mathbf{F}_{16}, u = n = 16, \lambda = 16, k = 16$.
Then the lenght of the public key is $\simeq 16k$ bytes, and the lenght of the signatures is $\simeq$ 16000 bits.

Example 3. $K = \mathbf{F}_{256}, u = n = 8, \lambda = 16, k = 16$.
Then the lenght of the public key is 3584 bytes, and the lenght of the signature is 8256 bits.

23 More variations of the IP schemes

It is easy to design a lot of variations of the IP schemes. For example we can have secret change of variables of degree two (instead of one), or non polynomial secret change of variables in mathematical equations in very general sets (not necessarily on finite fields). In [6] we will also present some variations where the security is relied to some famous mathematical problems (such as fast matrix multiplication, or fast cross product mutiplication). However, these schemes are based on Morphisms of Polynomials (instead of Isomorphism of Polynomials), so that they are generally less efficient.

24 A HFE challenge

We will present below a challenge on two explicit (quadratic) HFE signature schemes.

First scheme: This HFE scheme gives signatures of length 80 bits. The field K is $\mathbf{F}_2$, $n = 80$, and the general scheme is the scheme of section 4.3, where the hash function is SHA-1. The hidden polynomial f is

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

where the expression contains all the possible monomials of (monovariate) degree ≤ 100 (and a multivariate degree 2 in a basis over $\mathbf{F}_2$), and where the values β_{ij}, α_i and μ_0 are secret values randomly chosen in $\mathbf{F}_{2^{80}}$. (Hence, we have terms in a, a^2, a^3, a^4, a^5, a^6, a^8, a^9, a^{10}, a^{12}, a^{16}, a^{17}, a^{18}, a^{20}, a^{24}, a^{32}, a^{33}, a^{34}, a^{36}, a^{40}, a^{48}, a^{64}, a^{65}, a^{66}, a^{68}, a^{72}, a^{80}, a^{96}.)
Here, the length of the public key is $80 \cdot (\frac{80 \cdot 81}{2}) \cdot \frac{1}{8} = 32$ Kbytes.

Second scheme: This second signature scheme gives signatures of length 144 bits. The field K is $K = \mathbf{F}_{16}$, $n = 36$, and 4 of the 36 equations are not given public as explained in section 11. The hidden polynomial f is

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

where the expression contains all the possible monomials with a (monovariate) degree ≤ 4352 (and a multivariate degree 2 in a basis over $\mathbf{F}_{16}$), and where the values β_{ij}, α_i and μ_0 are secret values randomly chosen in $\mathbf{F}_{16^{36}}$. (Hence, we have terms in a, a^2, a^{16}, a^{17}, a^{32}, a^{256}, a^{257}, a^{272}, a^{4096}, a^{4097}, a^{4112}, a^{4352}.)
Here, the length of the public key is $32 \cdot (\frac{36 \cdot 37}{2} + 36) \cdot \frac{1}{2} = 11$ Kbytes.

The hash function here is MD5, and S is a valid signature of a message M if and only if HFE(S) =MD5(M). (Here HFE has a 144 bits input and 128 bits output.)

A prize of US $500 will be given to the first person able to give a valid signature of a message that we did not signed, for any of the two examples (*i.e.* the total prize is of US $1000 if the two public keys are broken).

25 Conclusion

We have designed two new classes of algorithms: HFE and IP. HFE can be used in asymmetric cryptography for encryption, for signatures, or for authentication. IP can be used in asymmetric cryptography for signatures or for authentication.
These algorithms are based on multivariate polynomials over a finite field of total degree two (or three for the IP algorithms with one secret).
One interesting point of HFE is that these algorithms can lead to very short asymmetric signatures (128 bits for example). Similarly they can encrypt messages by blocks whith very short blocks (128 bits blocks for example).
Another interesting point of these algorithms is that their security do not depend on factorisation or discret log, and very few algorithms for encryption or signatures in asymmetric cryptography are known that do no rely on these problems.
However a lot of problems are still open, for example:
Are these algorithms really secure ?
Is it possible to design strong HFE, with public polynomials of degree two and a secret function f with two or more monomials, that are also permutations ?
Is it possible to solve a general system of multivariate quadratic equations over $GF(2)$ much more quickly than with an exhaustive search ?

Acknowledgments

I want to thank Isabelle Guérin-Lassous and Nicolas Courtois for doing the simulations of Part II, section 8. I want also to thank Daniel Augot for doing the simulations of Part II, section 7.
However Daniel Augot told me that he does not want to endorse the security of HFE...

References

[1] I. Blake, X. Gao, R. Mullin, S. Vanstone and T. Yaghoobian, *"Applications of Finite Fields"*, Kluwer Academic Publishers.

[2] G. Brassard, *"A note on the complexity of cryptography"*, IEEE Tran. Inform. Theory, Vol. IT-25, pp. 232-233, 1979.

[3] E. Brickell, A. Odlyzko, *Cryptanalysis, A Survey of Recent Results*, p. 506, in "Contemporary Cryptology", IEEE Press, 1992, edited by Gustavus J. Simmons.

[4] D. Coppersmith, M. Franklin, J. Patarin and M. Reiter, *"Low-Exponent RSA with Related Messages"*, EUROCRYPT'96, Springer Verlag, pp. 1-9.

[5] D. Coppersmith and S. Winograd, *"Matrix Multiplication via Arithmetic Progressions"*, J. Symbolic Computation, 1990, Vol. 9, pp. 251-280.

[6] N. Courtois, J. Patarin, L. Goubin, *Improved algorithms for Isomorphisms of Polynomials*, to be published at EUROCRYPT'98.

[7] H. Dobbertin, *Almost Perfect Nonlinear Power Functions on $GF(2^n)$*, paper available from the author.

[8] J.C. Faugère, *personal communication.*

[9] M. Garey, D. Johnson, *"Computers and intractability, A Guide to the Theory of NP-Completeness"*, FREEMAN.

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

[11] T. Matsumoto and H. Imai, *"Public Quadratic Polynomial-tuples for efficient signature-verification and message-encryption"*, EUROCRYPT'88, Springer Verlag 1988, pp. 419-453.

[12] A. Menezes, P. van Oorschot and S. Vanstone, *"Some computational aspects of root finding in $GF(q^m)$"*, in Symbolic and Algebraic Computation, Lecture Notes in Computer Science, 358 (1989), pp. 259-270.

[13] G. Mullen, *"Permutation Polynomials over Finite Fields"*, in "Finite Fields, Coding Theory, and Advances in Communications and Computing", Dekker, Volume 141, 1993, pp. 131-152.

[14] J. Patarin, *"Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88"*, CRYPTO'95, pp. 248-261.

[15] J. Patarin, *"Asymmetric Cryptography with a Hidden Monomial"*, CRYPTO'96, pp. 45-60.

[16] J. Patarin, L. Goubin, *Asymmetric Cryptography with S-boxes*, ICICS'97, LNCS n. 1334, Springer-Verlag, pp. 369-380.

[17] J. Patarin, L. Goubin, N. Courtois, *C^{*-+} and HM: Variations around two schemes of T. Matsumoto and H. Imai*, this paper is not yet published, but is available from the authors.

[18] B. Schneier, *"Applied Cryptography"*, John Wiley and Sons, first edition, pp. 88-89, or second edition, pp. 104-105.

[19] A. Shamir, *"An efficient Identification Scheme Based on Permuted Kernels"*, CRYPTO'89, pp. 606-609.

[20] J. Stern, *"A new identification scheme based on syndrome decoding"*, CRYPTO'93, pp. 13-21.

[21] P. van Oorschot and S. Vanstone, *"A geometric approach to root finding in $GF(q^m)$"*, IEEE Trans. Info. Th., 35 (1989), pp. 444-453.

[22] J. von zur Gathen and V. Shoup, *"Computing Frobenius maps and factoring polynomials"*, Proc. 24th Annual ACM Symp. Theory of Comput., ACM Press, 1992.