Ultra-Short Multivariate Public Key Signatures

Jacques Patarin[1], Gilles Macario-Rat[2], Maxime Bros[3], and Eliane Koussa[1]

[1] Versailles Laboratory of Mathematics, UVSQ, CNRS, University of Paris-Saclay
jpatarin@club-internet.fr, ejkoussa@outlook.com
[2] Orange, Orange Gardens, 46 avenue de la République, F-92320 Châtillon, France
gilles.macariorat@orange.com
[3] University of Limoges, CNRS, XLIM, UMR 7252, F-87000 Limoges, France
maxime.bros@unilim.fr

Abstract. In this paper we study multivariate public key signature schemes with "ultra"-short signatures. In order to do so, we consider that signing and verifying a signature could require up to 1 minute of computation on a modern personal computer. Of course, very close results would be obtained for times around one second, at the cost of 6 to 10 more bits in the signatures, and more generally a trade-off could be found between computation time and signature size at each security level.
Despite the fact that a time of one minute is way bigger than the time required by general purpose multivariate-based signature schemes, such as Quartz or GeMMS, it enables us to reach ultra-short signature lengths, for instance, around 70 bits long signatures for a security of 80 bits.
Two main issues arise when one wants to build a signature scheme with ultra-short signatures: avoiding the birthday paradox attack and having the ability to sign arbitraly long messages, this paper gives ways to overcome both.
In a first part, we describe the attacks against multivariate public key signatures and use them to compute the minimal parameters that an ultra-short signature scheme would have. In a second part, we give an explicit example of such an ultra-short signature scheme using HFE-like algorithms. In the end, we give parameters for several level of security: 80, 90, 100 bits and the classic 128, 192, and 256 bits; for each of them, we propose different choices of finite fields.

Keywords: HFE, Multivariate Cryptography, Public Key Cryptography, Ultra-Short Signature.

1 Introduction

General context. At present, the RSA cryptosystem is the most used public key signature algorithm. According to the current best factorization algorithm

(General Number Field Sieve [10]) whose complexity is sub-exponential, to reach a security of 80 bits (which means that an attacker would need at least 2^{80} operations to recover the secret), one requires a public key of more than 1024 bits, and therefore the length of the signature is at least 1024 bits. Similarly, for a security of 128 bits, the length of an RSA signature is greater than 3000 bits.

Nowadays, cryptographers focus more on post-quantum cryptography, that is to say on cryptosystems that could resist attacks performed by quantum computers. There are mainly five kinds of post-quantum cryptography : multivariate, isogeny, code, lattice and hash-based cryptography. Since we deal with multivariate-based cryptography, our scheme might resist quantum attacks, however this is not the topic of this paper which focuses only on attacks with non-quantum computers.

Multivariate-based cryptography started in 1988 with the C* algorithm of Matsumoto and Imai [22]. It was later broken by Patarin ([24]) who then suggested a way to fix it with the Hidden Field Equations (HFE) scheme in 1999 ([28]). Following it, a lot of variations of this scheme were proposed, for instance HFE$v-$ (also in [28]), Quartz [26], and GeMSS [11].

At present, multivariate cryptography is an active research field since seven multivariate signature schemes have been submitted to the NIST Post-Quantum standardization process in 2018 and two of them (Rainbow and GeMSS) made it to the third round (including Alternate Candidates). Rainbow [13] is a multivariate signature scheme designed by Ding from the "Unbalanced Oil and Vinegar" scheme [21]. With multivariate-based cryptography, it seems possible to have short signatures, for example GeMSS provides 256 bits long signatures for an expected security level of 128 bits. More details about the HFE cryptosystems, its attacks, and its variations, will be given in the second part of this paper.

Our security model. In this article, we present multivariate ultra-short signature schemes which are secure in a security model where the cost of verifying a signature and computing the hash value of a chosen message are non-negligible. More precisely, if an adversary wants to forge a valid signature for a message of his choice, he does have access to two oracles, one which computes hash values and one which checks either a pair message-signature is valid or no ; but each request to one of those oracles have a cost. We express this cost in term of a minimal number of operations which is required to access the oracle's answer. Note that the verification oracle's answers can only be "Yes" or "No", that is to say that these answers do not contain any hash values.
In Section 9, we will describe real-life applications of our schemes which justify the use of this model. Furthermore, we will see that most of our parameters remain secure if the verification cost drops down to zero, as long as the access to the hash oracle has a non-negligible cost.

Our contribution. In the first part, we describe generic attacks against multivariate signature schemes and we use them to draw inferences about minimal values of parameters one would have to set to build secure ultra-short multivariate signature schemes. The purpose of such small signatures is that they could be transmitted orally by phone, stored in a tiny QR code, or combined in watermarking applications. Indeed, for a 80 bits security level, our signatures are about 70 bit-long, which is less than 20 hexadecimal digits.

In order to build signature schemes with ultra-short signatures, we exploit original ideas such as time constraints: one should not be able to sign or check a signature in less than a minute on a modern personal computer with a 3GHz frequency processor, that is to say with a total computation power around $3.10^9 \times 60 \approx 2^{37}$ word operations. In addition to this, our parameters are chosen not to require too much memory, typically less than 350MB.

In a second part, we describe explicit examples of an ultra-short multivariate signature scheme based on HFE variants. Then, we provide many parameters for this scheme according to different security levels (from 80 to 256 bits) and different choices of finite fields (from $\mathbb{F}_2$ to $\mathbb{F}_{17}$).

Part I: Generic Attacks

2 Types of attacks and minimal parameters

2.1 Different types of attacks

For the security of the schemes that we will design, we will have to take care of 4 types of attacks:

- Type 1: Attacks valid for any public key signature with L bits, when we want a given security in 2^z.
- Type 2: Attacks valid because it exists a value S (we will call it the "complete signature") such that from S we can obtain the signature (for us the signature S' will be a part of S) and to check that S is a valid complete signature of M we will proceed like this:
  1. We compute a value $H(M)$
  2. We check if $F(S) = H(M)$, where F is a function that can be computed fast. (For us F will be a set of quadratic equations and thus fast to compute). (These type 2 attacks use the fact that we have here $F(S) = H(M)$, unlike, for example, $F(S, M) = 1$ if the signature is valid , 0 otherwise.

- Type 3: Attacks valid because we will use for F a set of m multivariate quadratic equations over a finite field $\mathbb{F}_q$.
- Type 4: Attacks valid because we will use a HFE with some perturbations in order to introduce a secret trapdoor in F.

In this Part I we will only look for attacks of Types 1, 2 and 3, and how we can avoid them. We will call these attacks "generic attacks". Then, in Part II we will look for concrete design based on HFE variants, and the specific attacks on these designs.

2.2 Type 1 attacks

Let $G(S, M) = 1$ if S is a valid signature of message M, and $G(S, M) = 0$ otherwise. It is possible to compute this function G from the public key. Therefore a generic possible attack to find a valid signature S of a message M is to try various values for S and to check if we have $G(S, M) = 1$. In order to avoid this, if we want a security in 2^z, if the signatures S have L bits, and if G is computed in 2^g operations, we must always have: $L \geq z - g$.

For example, if we want a security in 2^{80} in order to have signatures of less than 80 bits we will need to have a "Slow" evaluation for G. This is what we will often do in this paper: we will design "Slow" modes of operations.

However, we consider in this paper that the time to compute G must be about 1 minute on a standard PC of 2020, i.e. a PC that can perform about 2^{31} operations per second, or about 2^{37} operations per minute. Therefore we see that if we want a security in 2^{80} we will need to have $L \geq 80 - 37 = 43$ bits. The aim of this paper is to study what kind of value L (larger than this bound, but as small as possible) we have with explicit schemes.

Remark 1. If the signature has only, say, 60 bits for example, then from the birthday paradox we know that when about 2^{30} messages will be signed, then with a high probability two messages will have the same signature. This may not be a real problem, because these two messages have been really signed by the legitimate user, and therefore we do not see a very dangerous possible attack based on the fact that they have the same signature. Moreover, it is also possible to ask the legitimate user to not sign more than one billion messages with the same public key if we want to avoid this, but this seems to us unnecessary.

2.3 Type 2 attacks

In most of the schemes that we will study in this paper we will check that S is a valid signature of a message M by an equality like this:

$$G(S) = H(M)$$

where G and H are public. (H is usually a Hash function, and G is given by the public key).

This offers other possibilities of generic attacks we detail here.

Collisions on H. First, it is possible to find collisions on H, i.e. to find 2 messages M_1 and M_2 with the same Hash, to ask for the signature of M_1 and like this to obtain the signature of M_2.

Let 2^z be the wanted security parameter. Let E be number of bits of the output of H (i.e. we can write the equality $G(S) = H(M)$ as an equality on E bits vectors). Let us write that H is computed in 2^h operations.

This birthday attack proceeds like this:

We compute 2^{z-h} values $H(M)$ and we store M at the address $H(M)$. Then we will obtain (with high probability) that two values share the same address (i.e. we have find a collision $H(M_1) = H(M_2)$) when $z - h > E/2$.

Therefore, in order to avoid this collision attack on H, we must always have:

$$E \geq 2z - 2h.$$

This comes from the birthday paradox, with an attack of complexity 2^z in time. (The memory needed is in 2^{z-h} . It also exists time and memory trade-off: with more time we can use less memory ; however we concentrate here in the time complexity).

In this paper we consider that the time to compute H must be (as for G) also about 1 minute on a standard PC of 2020, i.e. a PC that can perform about 2^{31} operations per second, or about 2^{37} operations per minute. Therefore $h \leq 37$, and we see that if we want a security in 2^{80} we will need to have $E \geq 2 \times 80 - 2 \times 37 = 86$ bits.

In this paper $E \geq 86$ will generally not be a problem. (In case of problem, we can use $G(S, M) = 1$ instead of $G(S) = H(M)$ to check a signature, but this limits the number of possible designs).

Collisions of the type: $G(S) = H(M)$. As above: Let 2^z be the wanted security parameter. Let E be number of bits of the output of H (i.e. we can write the equality $G(S) = H(M)$ as an equality on E bit vectors). This means that if in $G(S) = H(M)$ we have m equations on $\mathbb{F}_q$, then $E = m \log_2(q)$.

Let us write that H is computed in 2^h operations, and that G is computed in 2^g operations.

This birthday attack proceeds like this:

1. We compute 2^{z-h} values $H(M)$ and we store M at the address $H(M)$.
2. We compute 2^{z-g} values $G(S)$ and we see if there is a value M at this address $G(S)$.

Then we will obtain (with high probability) that a value M is already at this address (i.e. we have find a collision $G(S) = H(M)$) when $2z - h - g > E$.

The complexity of this birthday attack is 2^z in time.

(The complexity is 2^{z-h} in memory. If this is too much memory, it is possible to use less memory with more time computations, i.e. there are many time-memory trade-offs. We concentrate on the time complexity here).

In order to avoid this attack we must always have:

$$E \geq 2z - h - g. \tag{1}$$

We will denote $\Delta = E - 2z + h + g$. So (1) can be written: $\Delta \geq 0$.

Below (in section 4) we will present various mode of operations. Some will be able to work when $\Delta < 0$ but when $\Delta \geq 0$ more modes will work. For example, when $h = 37$:

- if we want a security in 2^{80} we have $\Delta = E - 123 + g$.
- if we want a security in 2^{90} we have $\Delta = E - 143 + g$.
- if we want a security in 2^{100} we have $\Delta = E - 163 + g$.
- if we want a security in 2^{128} we have $\Delta = E - 219 + g$. , etc.

Remark 2. Our slow hash function can be chosen to be the iteration of a standard hash function. A standard Hash function such as SHA-3 or SHA-256 has a speed of about 13 cycles per byte. Then for data of 72 bytes (36 words of 16 bits) we will need about 2^{10} cycles. Therefore 2^{37} operations (as mentioned above) means here to perform about 2^{27} Hash like SHA-3. Alternatively, we can also see things like this: we know that we need about 1 s to perform SHA-3 on 1 million (i.e. 2^{20}) messages of 72 bytes on a modern computer. Then for 2^{27} Hash like SHA-3 we will need about 1 or 2 minutes.

2^g is the time to compute the public equations G. Usually in this paper these equations will be a set of m (dense) quadratic (or cubic) equations in about m variables. Then we have:

For quadratic equations: $2^g \approx m^3/2$, i.e. $g \approx \log_2(m^3/2)$.
For cubic equations: $g \approx \log_2(m^4/6)$.

3 Gröbner bases on random Polynomial Systems (Minimum number of equations with a perfect trapdoor)

3.1 Polynomial system solving using Gröbner basis

In this Section we briefly introduce the Gröbner basis technique that is a fundamental tool for solving systems of multivariate polynomial equations. We refer to [30,3,4] for further details.
Gröbner basis method is a non-linear generalization of Euclid's algorithm for the gcd, as well as a generalization of Gaussian elimination for linear systems. Roughly speaking, a Gröbner basis is a set of multivariate polynomials having special properties that allow easy solutions derivation for complex polynomial systems.
As a matter of fact, it is possible to transform any multivariate polynomial system, even complicated ones, into Gröbner basis form using specific algorithms (like F4 [19] and F5 [20] algorithms).
The computational complexity of such method relies strongly on an important notion, namely the degree of regularity d_{reg}.
Intuitively, d_{reg} is the minimal degree for which a set of polynomials of degree d can form a Gröbner basis, and thus can be solved (see [5,20] for more details).
The complexity of a Gröbner basis computation detailed in [5,20] is in:

$$\mathcal{O}\left(\binom{n+d_{reg}}{d_{reg}}^{\omega}\right),$$

where $2 \leq \omega \leq 3$ is the linear algebra constant. Note also that for random systems, the degree of regularity can be evaluated by the computation of the first non negative coefficient of a Hilbert serie, see [5].

Polynomial system solving using the Hybrid approach. There are two general techniques for solving polynomial systems over finite fields:

- **Exhaustive search technique** (brute-force search) is a very general solving technique that consists of enumerating explicitly all the possible solutions in order to find the right one. The computational cost of such method is quite large.
- **Gröbner basis computation**, defined previously.

In order to speed up the system solving techniques, the authors of [8] combine the exhaustive search and the Gröbner basis methodologies into a hybrid approach for solving multivariate polynomials.
The basic idea of the hybrid approach is to fix some variables and then compute a Gröbner basis of a system with less variables and that is smaller than the original one. Then, a complete solution of the original system is recovered by performing an exhaustive search on the fixed variables combined with the solutions of the subsystem.
The complexity of the Hybrid approach is given in [8]:

$$\min_{0\leq k\leq n}\left(q^k\left(\mathcal{C}_{F5}(n-k,d_{reg}^{\max}(k))+\mathcal{O}\left((n-k)D^{\max}(k)^{\omega}\right)\right)\right),$$

where k is the number of fixed variables, $2 \leq w \leq 3$ is the linear algebra constant, $D^{max}(k)$ is the maximum number of solutions of the system in $\bar{\mathbb{F}_q}$ counted with multiplicity, and $\mathcal{C}_{F5}$ is the Gröbner basis computation complexity using the F5 algorithm [20].
When the trade-off k is well chosen, then the hybrid approach constitutes one of the most efficient algorithms for solving polynomial systems.

In Tables 1, 2, we present some values. These values evaluate the minimum number m such that to solve a system of m random equations in m variables would require 2^t computations from an attacker, for various values of the security parameter t (80, 90, etc.), and about one minute to solve when $m - f$ variables are known.

Table 1. Number f of variables that we can find with Magma in about 1 minute when we have m equations of degree 2 ($\omega = 2.37$), and size L in bits if we can find a perfect trapdoor.

Degree 2 $\omega = \mathbf{2.37}$	$\mathbf{2^{80}}$	$\mathbf{2^{90}}$	$\mathbf{2^{100}}$	$\mathbf{2^{128}}$	$\mathbf{2^{192}}$	$\mathbf{2^{256}}$
$\mathbf{q = 2}$	$m = 86$ $f = 29$ $L = 57+$	$m = 100$ $f = 30$ $L = 70+$	$m = 112$ $f = 31$ $L = 81+$	$m = 145$ $f = 33$ $L = 112+$	$m = 218$ $f = 37$ $L = 181+$	$m = 290$ $f = 42$ $L = 248+$
$\mathbf{q = 4}$	$m = 43$ $f = 21$ $L = 44+$	$m = 50$ $f = 22$ $L = 56+$	$m = 56$ $f = 23$ $L = 66+$	$m = 73$ $f = 26$ $L = 94+$	$m = 113$ $f = 30$ $L = 166+$	$m = 154$ $f = 32$ $L = 244+$
$\mathbf{q = 5}$	$m = 40$ $f = 20$ $L = 47+$	$m = 45$ $f = 20$ $L = 59+$	$m = 50$ $f = 21$ $L = 68+$	$m = 66$ $f = 24$ $L = 98+$	$m = 102$ $f = 27$ $L = 175+$	$m = 139$ $f = 30$ $L = 254+$
$\mathbf{q = 7}$	$m = 35$ $f = 19$ $L = 45+$	$m = 40$ $f = 20$ $L = 57+$	$m = 45$ $f = 20$ $L = 71+$	$m = 59$ $f = 22$ $L = 104+$	$m = 91$ $f = 26$ $L = 183+$	$m = 124$ $f = 29$ $L = 267+$
$\mathbf{q = 8}$	$m = 34$ $f = 19$ $L = 45+$	$m = 39$ $f = 19$ $L = 60+$	$m = 43$ $f = 20$ $L = 69+$	$m = 57$ $f = 23$ $L = 102+$	$m = 88$ $f = 27$ $L = 183+$	$m = 119$ $f = 29$ $L = 270+$
$\mathbf{q = 11}$	$m = 32$ $f = 18$ $L = 49+$	$m = 36$ $f = 19$ $L = 59+$	$m = 40$ $f = 20$ $L = 70$	$m = 52$ $f = 21$ $L = 108+$	$m = 81$ $f = 26$ $L = 191+$	$m = 111$ $f = 28$ $L = 288+$
$\mathbf{q = 13}$	$m = 31$ $f = 18$ $L = 49+$	$m = 35$ $f = 19$ $L = 60+$	$m = 39$ $f = 20$ $L = 71+$	$m = 51$ $f = 21$ $L = 112+$	$m = 79$ $f = 25$ $L = 200+$	$m = 107$ $f = 28$ $L = 293+$
$\mathbf{q = 16}$	$m = 30$ $f = 18$ $L = 48+$	$m = 34$ $f = 18$ $L = 64+$	$m = 38$ $f = 19$ $L = 76+$	$m = 50$ $f = 21$ $L = 116+$	$m = 77$ $f = 26$ $L = 204+$	$m = 104$ $f = 28$ $L = 304+$
$\mathbf{q = 17}$	$m = 29$ $f = 17$ $L = 50+$	$m = 33$ $f = 18$ $L = 62+$	$m = 37$ $f = 19$ $L = 74+$	$m = 49$ $f = 21$ $L = 115+$	$m = 76$ $f = 25$ $L = 209+$	$m = 103$ $f = 28$ $L = 307+$

Table 2. Number f of variables that we can find with Magma in about 1 minute when we have m equations of degree 3 ($\omega = 2.37$), and size L in bits if we can find a perfect trapdoor.

Degree 3 $\omega = \mathbf{2.37}$	$\mathbf{2^{80}}$	$\mathbf{2^{90}}$	$\mathbf{2^{100}}$	$\mathbf{2^{128}}$	$\mathbf{2^{192}}$	$\mathbf{2^{256}}$
$\mathbf{q = 2}$	$m = 82$ $f = 22$ $L = 60+$	$m = 92$ $f = 22$ $L = 70+$	$m = 103$ $f = 23$ $L = 80+$	$m = 131$ $f = 23$ $L = 108+$	$m = 196$ $f = 23$ $L = 173+$	$m = 262$ $f = 23$ $L = 239+$
$\mathbf{q = 4}$	$m = 40$ $f = 13$ $L = 54+$	$m = 45$ $f = 14$ $L = 62+$	$m = 50$ $f = 14$ $L = 72+$	$m = 64$ $f = 14$ $L = 100+$	$m = 96$ $f = 16$ $L = 160+$	$m = 128$ $f = 17$ $L = 222+$
$\mathbf{q = 5}$	$m = 35$ $f = 12$ $L = 54+$	$m = 39$ $f = 12$ $L = 63+$	$m = 44$ $f = 12$ $L = 75+$	$m = 56$ $f = 13$ $L = 100+$	$m = 83$ $f = 14$ $L = 161+$	$m = 112$ $f = 15$ $L = 226+$
$\mathbf{q = 7}$	$m = 29$ $f = 11$ $L = 51+$	$m = 33$ $f = 12$ $L = 59+$	$m = 36$ $f = 12$ $L = 68+$	$m = 47$ $f = 13$ $L = 96+$	$m = 71$ $f = 14$ $L = 161+$	$m = 96$ $f = 15$ $L = 228+$
$\mathbf{q = 8}$	$m = 27$ $f = 11$ $L = 48+$	$m = 31$ $f = 12$ $L = 57+$	$m = 34$ $f = 12$ $L = 66+$	$m = 45$ $f = 13$ $L = 96+$	$m = 68$ $f = 14$ $L = 162+$	$m = 92$ $f = 15$ $L = 231+$
$\mathbf{q = 11}$	$m = 25$ $f = 11$ $L = 49+$	$m = 28$ $f = 11$ $L = 63+$	$m = 31$ $f = 11$ $L = 70$	$m = 40$ $f = 12$ $L = 104+$	$m = 62$ $f = 13$ $L = 170+$	$m = 83$ $f = 14$ $L = 239+$
$\mathbf{q = 13}$	$m = 24$ $f = 11$ $L = 49+$	$m = 27$ $f = 11$ $L = 60+$	$m = 30$ $f = 11$ $L = 71+$	$m = 39$ $f = 12$ $L = 100+$	$m = 59$ $f = 13$ $L = 171+$	$m = 80$ $f = 14$ $L = 245+$
$\mathbf{q = 16}$	$m = 23$ $f = 11$ $L = 48+$	$m = 26$ $f = 11$ $L = 60+$	$m = 28$ $f = 11$ $L = 68+$	$m = 37$ $f = 12$ $L = 100+$	$m = 56$ $f = 13$ $L = 172+$	$m = 76$ $f = 14$ $L = 248+$
$\mathbf{q = 17}$	$m = 22$ $f = 10$ $L = 50+$	$m = 25$ $f = 11$ $L = 58+$	$m = 28$ $f = 11$ $L = 70+$	$m = 37$ $f = 12$ $L = 103+$	$m = 56$ $f = 13$ $L = 176+$	$m = 75$ $f = 13$ $L = 254+$

3.2 Choice of the constant of linear algebra ω

From a practical point of view we should choose $\omega = 2.81$.

However in order to take care of the fact that the systems are usually not dense, it could be useful, as a rough evaluation to choose $\omega = 2.37$ or $\omega = 2$. Moreover, in order to avoid the attacks of Type 1 and 2 we will have to increases the number m of equations that we will use. So the choice of ω will often not be so important.

When $q = 2$, we use the best-known asymptotic complexity for the direct attack: $2^{0.792m}$ (cf [3]). Maybe a more realistic estimation could be: $2^{0.88m}$.

When $q \geq 5$ we use the complexity results of [8]. The best-known algorithm is a "Hybrid" algorithm, i.e. we will perform exhaustive search on some variables, and then we will perform a Gröbner base algorithm (like F4 or F5 of Jean-Charles Faugère). The formula that they give is valid when $q \geq 5$, therefore we do not present values for $q = 3$ since a specific analysis would be needed.

3.3 Size of the signature with a perfect trapdoor

In table 3 below the symbol "+" in the value L means that since we will design a specific trapdoor function, and since we will have to take care of the generic attacks (Type 1 and Type 2 attacks above) we will have to increase this value for the size of the signature.

This value L is computed by the formula : $L =$ the first integer larger or equal to $(m - f) * \log_2(q)$.

Remark 3. With Magma we had a limitation of 366 M Bytes of RAM. This explain why sometime there is a limitation in memory before the time limitation of about 1 minute. Therefore, all the values in the tables below can be explicitly computed in less than 2 minutes and with less than 350 M bytes of RAM.

One surprise with these tables is the fact that $q = 2$ does not appear so far to be the best choice for very short signatures. Another surprise is the fact that the length L here are very similar for various values of q ($q = 5, 7, 11, \ldots$). However, the game is far to be over: we will have to consider many more attacks and the final best value for q is not at all clear at that point.

4 Modes of operation against the birthday paradox

Various modes of operations have been developed to avoid the birthday paradox attack, i.e. to be able to use a multivariate trapdoor function to sign any message even when the size of the input trapdoor is small. We will present here 8 modes of operation, including our new modes "Mixing Public keys", "Slow", "Memory", and "Slow + Memory.

4.1 Feistel-Patarin mode of operation

This is the mode of operation used in Quartz and GeMMS for example.

This mode of operation was first shown and analysed by Patarin in [28], and studied by Courtois in [12]. This mode of operation is used in the Quartz and GeMMS algorithms.

The general idea is to use the secret function more than one time (typically 4 times) to sign a message and to Xor the partial with more parts of the message. However generally this mode of operation will give us larger signatures than our new "Slow hash" mode that we will see below. The main reason for this is that we will not be able to recover many missing bits of the signature because the degree of the public equations that we have to solve increases: initially they are of degree 2, then 4 after 1 one iteration, then 4, etc. Moreover, at each iteration we need to add a few bits in the signature. This mode of operation is still very interesting when the Slow hash mode becomes too slow, or in order to design general purpose signature schemes (not ultra-short signatures schemes).

4.2 Gui mode of operation

In this mode of operation, a message is signed k times (typically $k = 2$ times) by the same public key. Thanks to this, we avoid the collision limitation, and we keep the length of the public key relatively small. However, this increases the length of the signature by a factor k. Since in this paper we want to have ultra-short signature, we will not use this mode of operation.

4.3 UOV mode of operation

In the UOV scheme, the secret key computation works exactly the same if we add in the public equations any equations in the $x_i y_j$ monomials. Therefore we have no problem of collision limitation with UOV: we check the signature as G(M, S) = 1 instead of G(S) = H(M) (see section 3). However, the length of UOV signatures is bigger than the length of HFEv- signatures (because we use much more vinegar variables). So, we will not use UOV in this paper.

4.4 Dragon mode of operation

Usually the public key of a multivariate scheme is a set of equations of the form:

$$\begin{aligned} y_1 &= P_1(x_1, \ldots, x_n) \\ y_2 &= P_2(x_1, \ldots, x_n) \\ &\ldots \\ y_m &= P_m(x_1, \ldots, x_n) \end{aligned}$$

where $P_1, P_2, \ldots, P_m$ are polynomials of small degree (typically 2).

It is also sometimes possible to use a public key that is a set of equations of this form:

$$\begin{aligned} P_1(x_1, \ldots, x_n, y_1, \ldots, y_m) &= 0 \\ P_2(x_1, \ldots, x_n, y_1, \ldots, y_m) &= 0 \\ &\ldots \\ P_m(x_1, \ldots, x_n, y_1, \ldots, y_m) &= 0 \end{aligned}$$

where $P_1, P_2, \ldots, P_m$ are polynomials of total small degree (typically 2 or 3). Such systems are called "Dragons schemes" in [25]. With these systems we could hope to have shorter signatures since we avoid type 2 attacks (cf section 3: we use the fact that we check a signature by equations of the form $F(S, M) = OK$ unlike $F(S) = H(M)$).

However, it is extremely risky to design a "Dragon scheme" from HFE variants because some attacks (such as generalizations of the Kipnis-Shamir attack) are much more efficient on Dragon schemes. And with other known design than HFE variants the length of the signature is much larger. (For example, Dragon mode works very well with UOV signatures, but the signature length is larger than with HFE). Therefore, in this paper we will not use this Dragon mode of operation.

4.5 Mixing public keys mode of operation

This is an interesting solution. However, it may be a bit risky to publish more than one public key with the same S and T.

Therefore, for security reasons we prefer at present our solutions with a slow Hash function, as we did in this paper.

4.6 "Multiple independent public keys" mode of operation

In this mode of operation, we use a set of k independent public keys (the new public key is a set of k previous public keys). Therefore, the length of the new public key is k times what was previously the length of the public key, but the security of the scheme remains the same. Like this, as we will see below, it is possible to avoid attacks based on the birthday paradox, but this is only realistic when k is not too large. In this paper we will first use a Slow Hash mode (cf below) and sometimes combine this Slow Hash mode with this idea of Multiple independent public keys.

Signature generation. When we want to sign a message M in this mode, we will proceed like this:

1. We first compute $H(M)$ a slow hash of M. This will take 2^h computations (typically $h = 37$ in this paper).
2. From this value $H(M)$ we compute a value $R(H(M))$ between 1 and k. This value gives the number of the public key that we will use (i.e. the m public quadratic equations that must be satisfied to sign M).
3. From the secret key associated with this public key, the signature is computed.

Attack. It is possible to attack this mode of operation with a complexity in 2^z and with a birthday paradox type of attack like this:

1. The attacker computes 2^{z-h} values $R(H(M))$.
2. The attacker select the public key that was obtained the most. In general he will obtain about $2^{z-h}/k$ values for this public key (see Remark below).
3. The attacker then computes 2^{z-g} values $G(S)$ and looks for a collision with a value obtained in 2. (Here 2^g denotes as above the time to compute a value $G(S)$).

This attack is expected to succeed with a good probability when $2^{2z-h-g} \geq k2^E$ where E denotes as above the number of bits of equalities to be satisfied when we check if a signature is valid from the public key.

Since by definition $\Delta = E - 2z + h + g$, we see that in order to avoid this attack we will have to choose: $k \geq 2^{-\Delta}$. This will give an acceptable public key length only if $-\Delta$ is not too large.

Remark 4. When 2^{z-h} is much larger than k, the number of values obtained in 2. for a given public key is a variable of mean value $2^{z-h}/k$, and with a standard deviation about the square root of this. Therefore for the public key with the more solutions will still have about $2^{z-h}/k$ solutions as claimed. For example we did a simple simulation by generating 10 millions random values between 1 and 100. The number that was obtained most was obtained 100 732 times in our simulation, and this number is very near 100 000 as expected.

4.7 Slow mode (and Slow + Multiple public keys trade-off mode) of operation

Here the idea is to use a slow hash function. This is the idea that we have presented in section 2. As long as the time of the slow hash function obtained for security is acceptable this solution is nice. If this time is too large, then we can combine this idea with the memory mode of operation (see 4.6), or choose another mode.

Table 3. Number f of variables that we can find with Magma in about 1 minute and values of m to avoid generic attacks.

Degree 2 $\omega = \mathbf{2}$ $\Delta \to \Delta \approx 0$	$\mathbf{2^{80}}$	$\mathbf{2^{90}}$	$\mathbf{2^{100}}$	$\mathbf{2^{128}}$	$\mathbf{2^{192}}$	$\mathbf{2^{256}}$
$\mathbf{q = 2}$ $r \leq 16$ 1	$\Delta = -10.4$ $94 \to 104$ $30 \to 31$ $64 \to 73+$	$\Delta = -15.8$ $108 \to 124$ $31 \to 32$ $77 \to 92+$	$\Delta = -23.3$ $120 \to 144$ $32 \to 33$ $88 \to 111+$	$\Delta = -40.1$ $158 \to 198$ $34 \to 37$ $124 \to 162+$	$\Delta = -82.2$ $242 \to 325$ $39 \to 44$ $203 \to 281+$	$\Delta = -128.0$ $323 \to 451$ $44 \to 51$ $279 \to 400+$
$\mathbf{q = 4}$ $r \leq 9$ 2	$\Delta = -13.4$ $47 \to 54$ $21 \to 23$ $52 \to 62+$	$\Delta = -18.8$ $54 \to 64$ $23 \to 24$ $62 \to 80+$	$\Delta = -26.3$ $60 \to 73$ $24 \to 26$ $72 \to 94+$	$\Delta = -43.1$ $79 \to 100$ $27 \to 29$ $104 \to 142+$	$\Delta = -83.2$ $122 \to 164$ $31 \to 32$ $182 \to 264+$	$\Delta = -121.8$ $166 \to 227$ $32 \to 37$ $268 \to 380+$
$\mathbf{q = 5}$ $r \leq 7$ 2.32	$\Delta = -8$ $43 \to 47$ $20 \to 21$ $54 \to 61+$	$\Delta = -13.6$ $49 \to 55$ $21 \to 22$ $65 \to 77+$	$\Delta = -19.1$ $55 \to 63$ $22 \to 23$ $77 \to 93+$	$\Delta = -36.8$ $71 \to 87$ $25 \to 26$ $107 \to 142+$	$\Delta = -72.4$ $110 \to 142$ $28 \to 30$ $191 \to 260+$	$\Delta = -106.3$ $150 \to 196$ $31 \to 34$ $276 \to 376+$
$\mathbf{q = 7}$ $r \leq 6$ 2.80	$\Delta = +1$ $m = 39$ $f = 19$ $L = 56+$	$\Delta = -4.4$ $44 \to 46$ $20 \to 21$ $68 \to 70+$	$\Delta = -7.1$ $50 \to 53$ $21 \to 22$ $82 \to 87+$	$\Delta = -20.1$ $65 \to 72$ $23 \to 25$ $118 \to 132+$	$\Delta = -48.1$ $100 \to 117$ $27 \to 29$ $206 \to 247+$	$\Delta = -76.7$ $135 \to 163$ $30 \to 32$ $295 \to 367+$
$\mathbf{q = 8}$ $r \leq 5$ 3	$\Delta = +5.7$ $m = 38$ $f = 19$ $L = 57+$	$\Delta = +1.3$ $m = 43$ $f = 20$ $L = 69+$	$\Delta = -3.2$ $48 \to 50$ $21 \to 21$ $81 \to 87+$	$\Delta = -13.1$ $63 \to 68$ $23 \to 24$ $120 \to 132+$	$\Delta = -40.2$ $96 \to 1104$ $27 \to 29$ $207 \to 243+$	$\Delta = -61.9$ $131 \to 152$ $30 \to 30$ $303 \to 366+$
$\mathbf{q = 11}$ $r \leq 5$ 3.45	$\Delta = +12.1$ $m = 35$ $f = 19$ $L = 56+$	$\Delta = +13.5$ $m = 41$ $f = 20$ $L = 73+$	$\Delta = +7.7$ $m = 45$ $f = 21$ $L = 84+$	$\Delta = +1.1$ $m = 59$ $f = 23$ $L = 125+$	$\Delta = -14.5$ $91 \to 96$ $27 \to 27$ $222 \to 239+$	$\Delta = -30.8$ $123 \to 132$ $30 \to 30$ $322 \to 353+$
$\mathbf{q = 13}$ $r \leq 4$ 3.70	$\Delta = +20.9$ $m = 35$ $f = 19$ $L = 60+$	$\Delta = +16.1$ $m = 39$ $f = 20$ $L = 71+$	$\Delta = +15.2$ $m = 44$ $f = 20$ $L = 89+$	$\Delta = +8.4$ $m = 57$ $f = 22$ $L = 130+$	$\Delta = +0.7$ $m = 89$ $f = 26$ $L = 233+$	$\Delta = -11.2$ $120 \to 123$ $29 \to$ $337 \to 348+$
$\mathbf{q = 16}$ $r \leq 4$ 4	$\Delta = +23.1$ $m = 33$ $f = 18$ $L = 60+$	$\Delta = +23.7$ $m = 38$ $f = 19$ $L = 76+$	$\Delta = +24.3$ $m = 43$ $f = 20$ $L = 92+$	$\Delta = +21.4$ $m = 56$ $f = 21$ $L = 140+$	$\Delta = +15.3$ $m = 86$ $f = 26$ $L = 240+$	$\Delta = +8.5$ $m = 116$ $f = 28$ $L = 352+$
$\mathbf{q = 17}$ $r \leq 4$ 4.08	$\Delta = +25.7$ $m = 33$ $f = 18$ $L = 62+$	$\Delta = +26.7$ $m = 38$ $f = 19$ $L = 78+$	$\Delta = +27.7$ $m = 43$ $f = 20$ $L = 94+$	$\Delta = +21.7$ $m = 55$ $f = 21$ $L = 139+$	$\Delta = +18$ $m = 85$ $f = 26$ $L = 241+$	$\Delta = +13.7$ $m = 115$ $f = 28$ $L = 355+$

Let us see what the values in the top-left square of Table 3 mean; with $\omega = 2$, in order to have a security in 2^{80}, we need to have a number m of equations ≥ 94 (unlike $m \geq 86$ with $\omega = 2.37$ as seen in Table 1. (This comes from the known results of the complexity of hybrid Gröbner bases on random quadratic systems as explained above). Then with $m = 94$ we would have a signature of at least 64 bits. However with $m = 94$ we have $\Delta = -10.4$ so this would give a very big public key if we use the independent key mode of section 4.4. since about $2^{-\Delta} = 1351$ public keys would be needed. In this paper we want very short signature but we are also interested in what we obtain with a reasonable length of the public key. Therefore it is interesting to see what length of signature we would have when $\Delta \approx 0$. Here this means that we increase m by an amount of $-\Delta/\log_2(q)$, since we have $E = m\log_2(q)$. Therefore we use $m = 104$ (instead of $m = 94$). This also explain why the value of ω is generally not so important: when we increases m in order to have $\Delta \approx 0$ then very often we will obtain the same value m whatever was the starting point obtained with ω. (Moreover the complexity of the best attacks on HFE variants schemes that we will use in Part II will often not depend on ω). Now with $m = 104$ we see that the length of the signature will be at least 73 bits. However on a specific scheme (such as variants of HFE) this could be more: therefore we write here "73+" and we will see more precisely in Part II what we will obtain.

Part II: Examples of parameters on HFE variants

5 Nude HFE, and HFE Variants

5.1 Description of (nude) HFE

Hidden field Equations (HFE2) algorithm was proposed by J. Patarin at Eurocrypt [27] to repair the algorithm C* of Matsumoto and Imai [22]. The basic idea of HFE is to hide the special structure of a univariate polynomial F over some finite field (usually $\mathbb{F}_{2^n}$) which allows F to have a quadratic polynomials representation in the small field.

HFE(q, n, D) shape. Let $\mathbb{F} = \mathbb{F}_q$ be a finite field of $q = p^m$ elements for some prime number p, $\mathbb{E} = \mathbb{F}_{q^n}$ its n-th degree extension, and $\phi : \mathbb{E} \to \mathbb{F}^n$ the canonical isomorphism between $\mathbb{E}$ and the corresponding vector space $\mathbb{F}^n$. Given $(\theta_1, \ldots, \theta_n)$ a basis of $\mathbb{E}$ as an $\mathbb{F}$-vector space, we have:

$$\begin{array}{rccc} \phi : & \mathbb{E} = \mathbb{F}_{q^n} & \longrightarrow & \mathbb{F}^n \\ & V = \sum_{i=1}^{n} v_i \theta_i & \longmapsto & (v_1, \ldots, v_n). \end{array}$$

Let $\mathcal{F}^*$ be the following map:

$$\begin{array}{rccc} \mathcal{F}^* : & \mathbb{F}_{q^n} & \longrightarrow & \mathbb{F}_{q^n} \\ & V & \longmapsto & F(V), \end{array}$$

with $F \in \mathbb{E}[X]$ is a univariate polynomial of the special form:

$$F = \sum_{0 \leq i \leq j \leq n}^{q^i+q^j \leq D} \alpha_{i,j} X^{q^i+q^j} + \sum_{0 \leq i \leq n}^{q^i \leq D} \beta_i X^{q^i} + \gamma, \tag{2}$$

where $\alpha_{i,j},\ \beta_i, \gamma\ \in \mathbb{F}_{q^n}$, and F is of degree at most $D \in \mathbb{N}$. Then, F has the HFE(D) shape that allows to have multivariate quadratic polynomials representation over $\mathbb{F}$ using the map $\mathcal{F} = \phi \circ \mathcal{F}^* \circ \phi^{-1}$:

$$\begin{array}{rccc} \mathcal{F} : & \mathbb{F}_q^n & \longrightarrow & \mathbb{F}_q^n \\ & (v_1, \ldots, v_n) & \longmapsto & (f_1(v_1, \ldots, v_n), \ldots, f_n(v_1, \ldots, v_n)), \end{array}$$

with the quadratic polynomials $(f_1, \ldots, f_n) \in (\mathbb{F}_q[x_1, \ldots, x_n])^n$ such that:

$$F\big(\phi^{-1}(x_1, \ldots, x_n)\big) = \phi^{-1}(f_1, \ldots, f_n)$$

$$F\Big(\sum_{i=1}^{n} \theta_i x_i\Big) = \sum_{i=1}^{n} \theta_i f_i$$

.

HFE problem. Basically, $\mathcal{F}^*$ is chosen to be an easily invertible and evaluated map. Using the canonical isomorphism ϕ, the map $\mathcal{F}^*$ can be transformed into a quadratic map $\mathcal{F} = \phi \circ \mathcal{F}^* \circ \phi^{-1}$. Thus, F can be written as a set of n quadratic polynomials $(f_1, \ldots, f_n)$ in n variables $(x_1, \ldots, x_n)$ over $\mathbb{F}$.
In order to build a cryptosystem based on the inversion of an HFE shaped polynomial, the original structure of $\mathcal{F}$ must be hidden since it is possible to find solutions of $F(x) = a,\ a \in \mathbb{F}_{q^n}$ in polynomial time. To do so, on uses two invertible affine maps

$$\mathcal{S}, \mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n.$$

Therefore, the public key consists of

$$\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} = \mathcal{S} \circ \phi \circ \mathcal{F}^* \circ \phi^{-1} \circ \mathcal{T},$$

and the secret key that yields the inversion of the public key is given by $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{F}^*$.
Thus, it's difficult to compute the inverse of $\mathcal{P}$ when its decomposition remains secret.

HFE is one of the most studied algorithm in cryptography. It can be used for authentication, encryption and also signature purposes.

5.2 HFE with some "perturbations": − (minus), v (external vinegar), w (internal vinegar)

In the previous sections, we presented the basic HFE. In this section, we recall several variations that can increase the security of HFE or its efficiency [27], The families HFEv and HFE$v-$ were introduced by Patarin in [28], and in 2001 a specific choice of parameters called "Quartz" [26] was published in order to have 128 bits signatures with an expected security in 2^{80} (see Appendix A).

The minus modifier −. This variant consist of omitting some multivariate polynomials from the public key of an HFE-based scheme $\mathcal{P} = (p_1, \ldots, p_n)$. It's named HFE- due to the act of hiding public equations which leads to an under determined system. Hence, this modifications is more often used for signatures schemes.
More precisely, HFE- involves using the following projection:

$$\pi : \mathbb{F}^n \longrightarrow \mathbb{F}^{n-s},$$

where $s \in \mathbb{N}^*$ denotes the number of removed polynomials. Thus, the construction of the public key comes as $\mathcal{P} = \pi \circ \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ for the original HFE-based scheme. Hence, the public key is seen as $n-s$ quadratic multivariate polynomials in n variables.

The vinegar modifiers v and w. While the first modification discussed earlier affects the public key, in this section we will present a variation which affects the structure of the private HFE shaped polynomial. It consists of adding some extra variables, namely vinegar variables $(z_1, \ldots, z_v) \in \mathbb{F}^v$, where $v \in \mathbb{N}$ denotes the number of vinegar variables. The secret HFE shaped polynomial is now a function of these vinegar variables. Its multivariate representation over the ground field should remain quadratic in the nude HFE case.
Note that, there are two classifications of vinegar variants:

- External vinegar modifier v, where the extra variables $(z_1, \ldots, z_v) \in \mathbb{F}^v$ are randomly chosen.
- Internal vinegar modifier w, where each extra variable $z_i \in \mathbb{F}$, for $i \in \{1, \ldots, v\}$ is a linear combination of the original variables $(x_1, \ldots, x_n) \in \mathbb{F}^n$.

Recall the special form of the the HFE shaped polynomial

$$F(X) = \sum_{0 \leq i \leq j \leq n}^{q^i + q^j \leq D} \alpha_{i,j} X^{q^i + q^j} + \sum_{0 \leq i \leq n}^{q^i \leq D} \beta_i X^{q^i} + \gamma,$$

where $\alpha_{i,j},\ \beta_i, \gamma\ \in \mathbb{F}_{q^n}$.
HFEv (respectively HFEw) consists of replacing F with a complicated map from $\mathbb{F}_{q^n} \times \mathbb{F}^v$ to $\mathbb{F}_{q^n}$:

$$F(X, z_1, \ldots, z_v) = \sum_{0 \le i \le j \le n}^{q^i+q^j \le D} \alpha_{i,j} X^{q^i+q^j} + \sum_{0 \le i \le n}^{q^i \le D} \beta_i(z_1, \ldots, z_v) X^{q^i} + \gamma(z_1, \ldots, z_v),$$

where $\alpha_{i,j} \in \mathbb{F}_{q^n}$, and $\beta_i, \gamma : \mathbb{F}^v \longrightarrow \mathbb{F}_{q^n}$ are respectively a linear and a quadratic random maps.

5.3 Probability to have 0 solutions in signature

For a random function f from E to E, where E is a finite set, the probability that for a value y of E there is at least one value x such that $f(x) = y$ is about 63.2% (i.e. 1 - 1/e).

For a random non homogeneous polynomial of degree 2 in n variables, or for a random homogeneous polynomial of degree 2 in $\mathbb{F}_{2^n}$ it is also about 63%. (Therefore, we will need to test in average about 1.5 values in order to find a signature).

However, on $\mathbb{F}_{2^n}$ when q is odd, and for a homogeneous random polynomial of degree 2 the probability is only 31.6% (i.e. 2 times less). (Therefore, in this case we will need to test in average about 3 values in order to find a signature). This is because in this case for all value X we have $f(X) = f(-X)$ and $X \neq -X$ (except if $X = 0$).

So, when q is odd we will generally choose a non-homogeneous polynomial. Then the secret linear transformations S and T defined in HFE will be chosen to be affine (linear with constants).

We expect that this non-homogeneous choice (when q is odd) does not create a security problem, because since S and T are very general linear bijections we expect that no attack should exist by exploiting only the degree 1 part of the public equations. (However, if in the future it appears that non-homogeneous equations are not a good choice, then we will come back to homogeneous solutions, with a probability to be invertible divided by 2).

6 The best known attacks on HFE$v-$

HFE$v-$ was first described by Patarin in [28]. Then a specific set of parameters was given in [26] (see also Appendix A for more details on Quartz). Since then a lot of work has been done on HFE$v-$ schemes. We will have to choose the parameters in a precise way in order to avoid all the known attacks.

There are essentially 3 kinds of attacks on HFE$v-$ schemes:

1. Differential attacks
2. Direct attacks
3. Key recovery attacks

We will present here a summary on these attacks.

6.1 Differential attacks

These attacks are very efficient when only one monomial is used, even when some perturbations are used. (For example, SFLASH was broken with them). Therefore, in order to avoid these attacks, we will always use at least 2 monomials of weight 2 in the secret HFE polynomial, i.e. we will always have at least the monomials X^2 and X^{q+1} if q is not a power of 2, and at least the monomials X^{q+1} and X^{2q+1} if q is a power of 2. Differential attacks are also important when we use the "w" perturbation (cf next section). Therefore, we will give more information on them in the next sections on the w perturbation.

6.2 Direct attacks

In these attacks we try to solve the public equations for a give value Y (i.e. for a given message we try to find a valid signature from the public equations) by using Gröbner algorithms like F4 or F5. We will denote by D the degree of the HFE polynomial, by a the number of equations removed, and by v the number of (external) vinegar variables. We will denote $r = \lfloor \log_q(D-1) \rfloor + 1$, that reflects the rank of the quadratic form associated with the HFE polynomial.

The complexity of this attacks is in the order of $\binom{d+n}{d}^{\omega}$

Where d is the "degree" of regularity of the system, and ω is the linear algebra constant ($2 \leq \omega \leq 3$, usually we consider $\omega = 2$ or $\omega = 2.37$). Therefore, the main difficulty in order to evaluate this complexity is to evaluate this value d.

In [15], an upper bound on this value d has been found:

$$d \leq 2 + (q-1).(r+a+v)/2. \tag{3}$$

This upper bound is interesting, is proved, and is valid for all q. It shows that often a HFE$v-$ scheme will have a smaller degree of regularity than a random system. Moreover, generally d is significantly smaller (cf below).

Remark 5. When we want to solve random systems, the best algorithms are usually hybrid algorithms, i.e. we will fix some variables and find the other with Gröbner bases algorithms. When we attack HFE$v-$ schemes however, it seems to be a bad idea to use Hybrid algorithms, therefore here we use only pure Gröbner bases algorithms.

When $q = 2$, many experiments have been done. As long as d is not larger than the d_{reg} value of random systems (that we can compute from Hilbert series), these experiments show that d can be estimated to be:

When $q = 2$ we have: $d = \lfloor (a + v + r + 7)/3 \rfloor$

However, when q is different from 2, very little is known at present on d. This is why we have made our own experiments with Magma (cf Appendix D). It is difficult to see if q will be in factor as in the formula (3) above. However d starts with the value q (when $D = 2$) and when $q = 5$ it increases apparently of 1 almost each time that a, v or r increases by 1, unlike what we have on $\mathbb{F}_2$ where a, v, and r generally have to be increased by 3.

So, we will assume that when $q \geq 5$, we have: $d \geq q - 2 + r + a + v$, (as long as this value is $\leq$ the degree of regularity that we have with random equations).

Remark 6. When $q \geq n$ we have noticed that $d = n + 1$. However, in our parameters we will always have $q < n$.

6.3 Key recovery attacks

Let m be the number of public equations, let a be the number of equations removed, let v be the number of external vinegar variables, let $r = \lceil \log_q(D) \rceil + (a + v)$, and let $n = m + a$.
Attacking a HFE$v-$ instance with parameters (m, D, r, a, v) reduces to solving a MinRank instance of the following form: one wants to find a linear combination with coefficients in $\mathbb{F}_{q^n}$ of $K := m$ square matrices of size n (with entries in $\mathbb{F}_q$) with a small rank r or less. This reduction correspond to the key recovery attack described in [31].
With the recent progresses made for solving the MinRank problem in [2], this key recovery attack is currently the most threatening attack against HFE$v-$; previously it was the attack in [16]. [2] uses a clever algebraic modeling of the MinRank problem in order to solve it by direct linearization instead of using generic Gröbner basis algorithms such as F4 or XL.
For the MinRank parameters mentioned above, this attack requires

$$\mathcal{O}\left(K(r+1)\left(\binom{n}{r}\binom{K+b-1}{b}\right)^2\right) \tag{4}$$

operations in $\mathbb{F}_{q^n}$ as long as there exists an integer b in $\{1, \ldots, r+2\}$ which fulfills the following condition (5) and such that $b < q$.

$$\binom{n}{r}\binom{K+b-1}{b} - 1 \leq \sum_{i=1}^{b} (-1)^{i+1} \binom{n}{r+i}\binom{l+i-1}{i}\binom{K+b-i-1}{b-i}. \quad (5)$$

In the previous complexity formula, $l := n$ is the number of rows of the matrices; in order to get the smallest complexity, one can delete a few columns to get a new instance with n' columns as mentioned in [2]. Thus, one replaces n by n' in (4) and (5), but $l := n$ has to remain the same. One should be careful that this optimization works if and only if the new MinRank instance still has a single solution. Thus, the complexity of the attack is the minimum value obtained from (4) for a valid choice of b and n'.

7 HFE$v-$ parameters in degree 2

In this section we will select parameters when our trapdoor function is a HFE$v-$ scheme. (As everywhere in this paper, we want the shortest signatures computable and verifiable in at most about 1 minute on a PC).

7.1 Choice of the degree D of the HFE polynomial

We must have at least 2 monomials, due to many powerful cryptanalytic results on one monomial (even with perturbations): see [9]. So we will always have at least the two monomials: X^2 and X^{1+q} .

Let us denote $r = \lceil \log_q(D) \rceil$. From our `MAGMA` computations (c.f. Appendix C), we assume that our maximum value for r (in order to spend less than about 1 minute of computation) will be approximately the ones given in Table 4.

Table 4. Values of r according to q.

q	2	4	5	7	8	11	13	16	17
r	16	9	7	6	5	5	4	4	4

With HFE$v-$ schemes we use only the two perturbations $-$ (minus) and v (vinegar) and they cost almost nothing in signature. Therefore, with HFE$v-$ we will be able to choose to the maximum value for D compatible with a time of at most about 1 minute. (However, for other most costly perturbations (such as the w perturbation below) we will have to choose a smaller value for D).

7.2 Choice of the v and $-$ parameters

When D is chosen as above, we add the required (and minimum) number of v and $-$ parameters in order to have the direct attack and key recovery attack from [2] with the expected security wanted.

7.3 Examples: our candidate solution

We will present only examples where the verification is limited to 1 minute. The complexities are evaluated according to the best known cryptanalysis, i.e. this is usually from the recent key recovery attack mentioned in section 6.

Table 5. 80 bit security parameters

q	r	m	a	v	Complexity	f	Signature Size (bits)	Public Key Size (kBits)
2	16	104	0	0	84.4	31	73	69.3
4	9	54	3	4	81.6	23	76	24.9
5	7	47	5	4	81.0	21	82	21.2
7	6	39	5	5	80.2	19	85	16.3
8	5	38	6	5	80.1	19	90	17.0
11	5	35	6	6	83.7	19	97	16.6
13	4	35	7	6	83.7	19	108	18.6
16	4	33	7	6	83.5	18	112	17.4
17	4	33	7	6	83.5	18	115	17.8

In Appendix A, we present a "Slow-Quartz" solution based on a previous Quartz function mixed with our new slow hash mode, and with 16 independent public keys. Its parameters are given in table 6:

Table 6. Slow-Quartz 80 bit Security

q	r	m	a	v	Complexity	f	Signature Size (bits)	Public Key Size (kBits)
2	8	100	3	4	80.3	30	77	1128

We see that the results given above with $q = 2$ are slightly better than what we obtain with slow-Quartz. However the Quartz parameters have been studied for many years, so it may be nice to use these conservative values. Moreover with these parameters it is also possible to encrypt, not only to sign (with a time to decrypt of about 0.5 s at present, cf Appendix A).

Table 7. 90 bit Security

q	r	m	a	v	Complexity	f	Signature Size (bits)	Public Key Size (kBits)
2	16	124	1	1	93.2	32	94	121.1
4	9	64	5	4	90.3	24	98	42.2
5	7	55	6	6	93.7	22	105	35.5
7	6	46	7	6	92.9	21	107	27.9
8	5	43	7	7	92.6	20	111	26.0
11	5	41	7	7	92.4	20	122	26.6
13	4	39	8	7	92.2	20	126	26.1
16	4	38	8	7	92.1	19	136	26.5
17	4	38	8	7	92.1	19	139	27.1

Table 8. 100 bit Security

q	r	m	a	v	Complexity	f	Signature Size (bits)	Public Key Size (kBits)
2	16	144	2	2	101.8	33	115	193.8
4	9	73	6	6	102.9	26	118	65.2
5	7	63	7	7	102.3	23	126	53.6
7	6	53	8	7	101.5	22	130	42.6
8	5	50	8	8	101.3	21	135	40.4
11	5	45	8	8	100.8	21	139	35.9
13	4	44	9	8	100.7	20	152	37.5
16	4	43	9	8	100.6	20	160	38.4
17	4	43	9	8	100.6	20	164	39.2

Variants. Many variants of these schemes are possible.

For example, instead of having maximum r for 1 minute computation and using v and a, we can choose $r = 2$ (i.e. minimum r, since $r = 1$ is broken) and use maximum w for 1 minute computation (where w is the internal vinegar variable) and then the same v and a as before. It is expected that the result and the security will be similar.

Another variant consists in using k independent public keys, as explained above with the "independent public keys" mode. Then the public key will be bigger, but the signature will be a few bits smaller. This is what we do in the table 12 below.

Remark 7. Here in table 12, we have $a = v = 0$, i.e. we use a "nude" HFE scheme (HFE with no perturbation). This may look surprising since since super-polynomials attacks are known on nude HFE, and therefore it is generally not recommended to use nude HFE. However here we have a very specific application: we want a scheme with very short signature with 2^{37} computations for the

Table 9. 128 bit Security

q	r	m	a	v	Complexity	f	Signature Size (bits)	Public Key Size (kBits)
2	16	198	6	5	131.2	37	173	530
4	9	100	9	9	128.3	29	178	171.4
5	7	87	11	10	131.7	26	191	145.1
7	6	72	11	11	130.8	25	194	110.1
8	5	68	12	11	130.6	24	201	104.2
11	5	59	12	11	130.0	23	205	84.7
13	4	57	12	12	129.8	22	219	85.5
16	4	56	12	12	129.8	21	236	88.6
17	4	55	12	12	129.7	21	238	86.7

Table 10. 192 bit Security

q	r	m	a	v	Complexity	f	Signature Size (bits)	Public Key Size (kBits)
2	16	325	13	13	193.4	44	307	2450
4	9	164	17	17	194.4	32	332	788.8
5	7	142	18	18	193.6	30	344	641.2
7	6	117	19	18	192.4	29	351	478.5
8	5	110	19	19	192.0	29	357	444.1
11	5	96	19	19	192.1	27	371	366.6
13	4	89	20	20	195.8	26	382	337.1
16	4	86	20	20	195.6	26	400	335.9
17	4	85	20	20	195.6	26	405	334.0

legitimate users and 2^{80} computations for the non-legitimate users. Therefore for this application, a super polynomial attack (or even a polynomial attack with a degree ≥ 2.16) might not be a problem.

Remark 8. The complexity of C2 is larger than 2^{80} if $32.32\omega > 80$, i.e. if $\omega > 2.47$. This is generally considered as a very reasonable assumption. However if we want to assume a smaller value of ω (for example in order to take into account the fact the systems are not dense) then we can choose $a = 1$ and $v = 0$ (or $a = 0$ and $v = 1$) instead of $a = 0$ and $v = 0$. This will add only one bit in the signature, and the degree of regularity of C2 should became 8, instead of 7 (because it is expected that this degree is the integer part of $((r+a+v+7)/3)$). Then the complexity of C2 becomes at least 35.6ω, and this is larger than 80 as long as $\omega > 2.24$.

Parallelization. For many solutions, some parallel computations are possible. First, it is always possible to choose a slow hash function where parallel computations are possible. For example the slow hash function can be the XOR of k

Table 11. 256 bit Security

q	r	m	a	v	Complexity	f	Signature Size (bits)	Public Key Size (kBytes)
2	16	451	21	21	257.9	51	444	6704
4	9	227	25	25	257.0	37	488	2134
5	7	196	26	26	256.8	34	509	1715
7	6	163	27	26	256.7	32	517	1309
8	5	152	27	27	256.2	30	528	1186
11	5	132	28	27	259.2	30	544	980
13	4	123	28	27	256.7	29	552	885
16	4	116	28	27	256.1	28	572	833
17	4	115	28	27	256.0	28	581	834

Table 12. Ultra Short signatures with expected security about 2^{80}, with $q = 2$, $D = 32769$, (i.e. $r = 16$) $a = v = 0$ (i.e. "Nude" HFE) and large public keys (i.e. multiple public keys mode). The time to sign and verify a signature is about 1 minute (2^{37} computations).

m	r	g	Δ	k	f	$C1$	$C2$	Signature Size (bits)	Public Key Size
86	16	18.2	-18.8	456000	29	83.6	32.32ω	57	17 GBytes
88	16	18.3	-16.7	106000	29	83.7	32.56ω	59	4.2 GBytes
92	16	18.5	-12.5	5800	30	83.9	33.02ω	62	271 MBytes
94	16	18.6	-10.4	1352	30	84.0	33.27ω	64	67.6 MBytes
96	16	18.7	-8.3	316	30	84.1	33.47ω	66	16.8 MBytes
100	16	18.9	-4.1	18	31	84.3	33.89ω	69	1.08 MBytes
104	16	18.2	0.1	1	31	84.4	34.30ω	73	69.3 kBytes

functions SHA-3$(i||x)$, where k is well chosen. It is not so easy to use parallelization on the Berlekamp algorithm. However some perturbations have a natural parallelization, such as the w (i.e. internal vinegar) parallelization.

Fast Verification. It is also possible to have a slow hash computation when we sign a message, and a fast verification of this hash to check the signature. For example, by including 10 bits on the signature, we can check the hash 1024 times faster. However in this paper we look for the shortest signature, so these variants are not our main concern.

8 HFE variants with public keys of degree 3

Above we have given some Tables of values for "perfect" multivariate schemes of degree 2 and degree 3, in order to have very short signatures. We have seen that

the length of signature is similar in degree 2 and in degree 3. Then, in degree 2 we have designed some specific schemes based on variants of HFE. Thus, it is natural to similarly design some specific schemes in degree 3 based on variants on HFE. The number n of variables can be relatively small when q is not too small, and then the length of the public key may be still reasonable. Moreover, direct attacks on HFE variants of degree 3 are more difficult than direct attacks on HFE variants of degree 2 (see Appendix E).

However, we will not present here some candidate parameters for HFE variants in degree 3. This is because to attack a HFE in degree 3 we can always look at the differentials (they are of degree 2) and then find the secret keys with Kipnis-Shamir style attacks, i.e. MinRank attacks. (For a generalization of the Kipnis-Shamir attack, to find the secret key in degree 3, some results are given in [1]).

Maybe in the future some new multivariate schemes (different from HFE$v-$) will have very short signature in degree 3, and will have more interesting properties in degree 3 than in degree 2. But in this paper, we will present only candidates with public keys of degree 2.

9 Discussion about our security model

9.1 Examples of implementation of our security model

Recall that, in this paper, all of our schemes rely on a security model where the oracles for the verifications and for the computation of hash values have limited resources; that is to say that each request to one of those oracles has a non-negligible cost for an attacker.

As our signatures are "ultra-short", one of their most valuable interests is for intance that they could easily be spelled by a human to another one over the phone. For example in a standard PC we can manage signatures of about 64 bits that can be represented as 16 hexadecimal values such as: `45A5F352CDE20240`. This is less than the size of a strong Wifi-Box password to be type into a smartphone for example. Moreover, if we use Alphanumeric value (Numbers $0 \dots 10$ + Letters $a \dots z$ and $A \dots Z$, i.e. 62 characters) then 64 bits represents only 11 Alphanumeric characters such as: `4fDjK457GfD`.

A concrete example involving the non-negligible cost of verification and hash could be the following: in order to activate a software, a user needs to enter an ultra-short signature in it, this signature will be given to the user over the phone; in this case, the cost of the verification would not be negligible as it would be done by the user's personal computer. In this example, if the user manages to

get an illegal copy of the software and is looking for a valid signature to activate it, he would have to "pay" the cost of every verification on his computer.

Another example of our security model involves QR codes; as the ultra-short signatures can fit in small QR codes, a hand device such a phone could be used to verify them. In such a case, the verification cost would not be free as well.

Last but not least, if the verification process is done online by a server which receive requests from a user (a client), it is really easy to create artificially a verification cost. Indeed, the server just has to require the client to solve a puzzle before answering. The cost of solving the puzzle, to match our security parameters, would never be more than 1 minute or, similarly, 2^{37} bit operations.

These real-life applications of our security model seem legitimate since for all of them an attacker would require billions of cell phones or personal computers to be able to verify a signature or to compute an hash value instantly.

9.2 Security of our ultra-short signature scheme in a classical security model

In the classical security model used to prove the existential unforgeability under chosen message attack (EUF-CMA), an attacker has access to an oracle which can tell him if a given string of bits is a valid signature for a message. As our signature scheme involves the use of hash functions, the attacker also requires an oracle which computes hash values. If, like in the classical security model, the attacker has access to those oracle "for free" (that is to say at no extra cost than generating the request), our scheme does not hold since it relies on slow verifications and hash. Nevertheless, we will see in this section that as long as the access to the hash-oracle has a non-negligible cost, our scheme remains secure only by changing a few of its parameters. Indeed, if the access to the hash-oracle were free, this would improve significantly the birthday attack (such as the one given in Section 2).

If the access to the verification oracle is free, for a given signature of length l bits and for a security of 2^λ operations, the attacker can brute force the 2^l possible signatures. Usually, as $l \geq \lambda$, this is not a problem, but for the few parameters for which the length of the signature is smaller than the security λ, we only need to adjust slightly the parameters. For example, for a security level of 2^{80}, with $q = 2$ (see Section 7.3), we should set $a+v = 7$ (instead of 0), and $r = 8$ (instead of 16), then the signature length would be 80 bits instead of 73 bits. Similarly, with $q = 4$, we should set $a + v = 9$ (instead of 7), and $r = 8$ (instead of 9), then the signature length would be 80 bits as well. With these parameters, the other attacks are less efficient, so it enables us to reduce the value of D (from 65536 to 256 for the former and from 262144 to 65536 for the latter) and thus obtain significantly faster signature.

Note that for 90 bits of security (or more), we do not have to change the parameters given in Section 7.3 since the lengths of the signatures are already larger than the security parameter λ.

Remark 9. To ensure our signatures against brute-force attacks (i.e. finding at least one valid signature among the 2^l possible bit strings), there is not necessarily a need for the verification cost to be as big as 2^{37} operations. Indeed, for the two sets of parameters for which the signatures are shorter than the security parameter λ, if the verification oracle asks the user/attacker to solve a *simple* puzzle before answering his request, it is usually enough. More precisely, with $q = 2$ and 73 bit-long signatures (c.f. Section 7.3), a puzzle requiring 2^7 operations to be solved would usually be enough to reach a security of 80 bits, and with $q = 2$ and 64 bit-long signatures (Table 12 page 27), a puzzle in 2^{16} operations would usually be enough.

10 Evolution with time

In the near future, say in 10 or 30 years, it is expected that computers will be more powerful than now (we consider only classic, i.e. non quantum, computers, since quantum computers are out of the scope of this paper). It is also expected that the capacity of the human's brain will be very similar as now. If a signature has to be sent, for example over the phone, to a human, will we have to increase the length of the signature to keep the same level of security?

As seen above, when the security parameter increases, the length of the signature must also increase. However, this was not the same problem because in 1 minute much more computations could be done in the future.

Alternatively, we can see things like this: in this paper the legitimate user makes about 2^{37} computations and we have looked for expected security in 2^{80} (i.e. 2^{43} times more than the legitimate user), then 2^{90} (i.e. 2^{53} times more than the legitimate user), then 2^{100} (i.e. 2^{63} times more than the legitimate user), etc. However in this section we imagine that the legitimate user makes more (future) or less (past) than 2^{37} computations, but that the expected security is a fixed time this value, typically 2^{43} times. What will then be the evolution of the length of the signature ?

For example, let us assume that the legitimate user makes about 2^{27} computations (instead of 2^{37}) and that we look for a security in 2^{70} (instead of 2^{80}).

Then typical parameters are : $q = 2$, $r = 6$ (i.e. a degree $D = 129$ of HFE for computations in about 60 ms), $n = 95$ (in order to have $\Delta \approx 0$), $f = 20$ (in order to check the signature in about 60 ms). Then the signature will be about

$95 - 20 = 75$ bits, i.e. about the same as in this paper with 1024 times more computations for the legitimate user and the attacks.

(Our Magma simulation shows that to solve a system of 95 quadratic equations in 20 variables over $\mathbb{F}_2$ we need about 85 ms, and the expected security of the parameters given here is about 2^{70}).

A more precise analysis would be required here, but as a first evaluation we can consider that the length of our signatures seems to evoluate very slowly when the computing power of the legitimate user evoluates proportionally with the power of the attacks.

11 Conclusion

At present the shortest public key signatures are obtained with multivariate signature schemes such as Quartz, or GeMMS, with signature size typically between 128 and 256 bits, and time to sign and verify the signature in milliseconds.

In this paper we have studied how to design even shorter signatures when we accept a time to sign and verify the signature of about 1 minute (i.e. about 2^{37} computations).

For example, in expected security 2^{80} we have designed a signature schemes with about 64 bits. Interestingly, there are many designs, variants, and parameters to achieve this.

In order to avoid problems from the birthday paradox, we have also designed some specific new modes of operations.

These signature lengths are much shorter than with other public key signatures schemes. This was also obtained thanks mainly to the following ideas and tradeoffs:

- We find some missing bits of the signature much faster than by exhausting search by using Hybrid Gröbner bases algorithms.
- We use very slow hash functions.
- Sometimes we use many independent public keys.

In the future, with future progress in cryptanalysis, it is likely that more efficient attacks will be found, and therefore that we will have to add some bits in the signature in order to keep the same security parameter. As long as the best known attacks are still exponential in the $-$ and v perturbations, only a few more bits should be enough. Some of the schemes that we have presented here have a security that rely on problems studied for more than 20 years (for example our "Slow Quartz" signatures).

Acknowledgments

We would like to thanks Daniel Bernstein whose useful remarks have suggested us to clarify our security model.

References

1. Baena J., Cabarcas D., Escudero D., Khathuria K., Verbel J. (2018). Rank Analysis of Cubic Multivariate Cryptosystems, 10.1007/978-3-319-79063-3_17.
2. Bardet M., Bros M., Cabarcas D., Gaborit P., Perlner R., Smith-Tone D., Tillich J.-P., and Verbel J. (2020, February). Improvements of Algebraic Attacks for solving the Rank Decoding and MinRank problems. Preprint: http://arxiv.org/pdf/2002.08322.
3. Bardet, M., Faugère, J. C., Salvy, B., Spaenlehauer, P. J. (2013). On the complexity of solving quadratic boolean systems. Journal of Complexity, 29(1), 53-75.
4. Bardet, M., Faugère, J. C., Salvy, B. (2004, November). On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In Proceedings of the International Conference on Polynomial System Solving (pp. 71-74).
5. Bardet, M. (2004). Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie (Doctoral dissertation).
6. Bettale, L., Faugère, J.-C., Perret, L. (2011, March). Cryptanalysis of multivariate and odd-characteristic HFE variants. In International Workshop on Public Key Cryptography (pp. 441-458). Springer, Berlin, Heidelberg.
7. Bettale, L., Faugère, J.-C., Perret, L. (2013). Cryptanalysis of HFE, multi-HFE and variants for odd and even characteristic. Designs, Codes and Cryptography, 69(1), 1-52.
8. Bettale L., Faugère J.-C., Perret Ludovic. (2010). Hybrid approach for solving multivariate systems over finite fields. Journal of Mathematical Cryptology, De Gruyter, 3 (3), pp.177–197. (10.1515/jmc.2009.009). (hal-01148127).
9. Bouillaguet C., Fouque P.-A., Macario-Rat G. (2011). Practical Key-Recovery for All Possible Parameters of SFLASH. In: Lee D.H., Wang X. (eds) Advances in Cryptology – ASIACRYPT 2011. ASIACRYPT 2011. Lecture Notes in Computer Science, vol 7073. Springer, Berlin, Heidelberg.
10. Buhler, J. P., Lenstra H. W., Pomerance C. (1993). Factoring integers with the number field sieve. In The development of the number field sieve, Springer (pp. 50-94).
11. Casanova, A., Faugère, J.-C., Macario-Rat, G., Patarin, J., Perret, L., Ryckeghem, J. (2017). GeMSS: A great multivariate short signature. Submission to NIST Post-Quantum Standardization Process.
12. Courtois N. T. (2004). Short Signatures, Provable Security, Generic Attacks and Computational Security of Multivariate Polynomial Schemes such as HFE, Quartz and Sflash. IACR Cryptol. ePrint Arch. 2004: 143.
13. Ding, J., Chen, M.-S., Petzoldt, A., Schmidt, D., Yang, B.-Y. (2019). Rainbow. Submission to the Second Round of NIST Post-Quantum Standardization Process.
14. Ding, J., Schmidt, D. (2005, January). Cryptanalysis of HFEv and internal perturbation of HFE. In International Workshop on Public Key Cryptography (pp. 288-301). Springer, Berlin, Heidelberg.

15. Ding J., Yang B.-Y. (2013) Degree of Regularity for HFEv and HFE$v-$. In: Gaborit P. (eds) Post-Quantum Cryptography. PQCrypto 2013. Lecture Notes in Computer Science, vol 7932. Springer, Berlin, Heidelberg.
16. Ding, J., Perlner, R., Petzoldt, A., Smith-Tone, D. (2018, April). Improved cryptanalysis of HFE$v-$ via projection. In International Conference on Post-Quantum Cryptography (pp. 375-395). Springer, Cham.
17. Dubois, V., Granboulan, L., Stern, J. (2007, April). Cryptanalysis of HFE with internal perturbation. In International Workshop on Public Key Cryptography (pp. 249-265). Springer, Berlin, Heidelberg.
18. Faugère J.-C., Joux A. (2003). Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases. In: Boneh D. (eds) Advances in Cryptology - CRYPTO 2003. CRYPTO 2003. Lecture Notes in Computer Science, vol 2729. Springer, Berlin, Heidelberg.
19. Faugère, J.-C. (1999). A new efficient algorithm for computing Gröbner bases (F4). Journal of pure and applied algebra, 139(1-3), 61-88.
20. Faugère, J.-C. (2002, July). A new efficient algorithm for computing Gröbner bases without reduction to zero (F 5). In Proceedings of the 2002 international symposium on Symbolic and algebraic computation (pp. 75-83).
21. Kipnis, A., Patarin, J., Goubin, L. (1999, May). "Unbalanced Oil and Vinegar Signature Schemes". Advances in Cryptology — EUROCRYPT '99, Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, p. 206–222.
22. Matsumoto, T., Imai, H. (1988, May) Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In Workshop on the Theory and Application of of Cryptographic Techniques (pp. 419-453). Springer, Berlin, Heidelberg.
23. Mohamed, M., Ding, J., Buchmann, J. (2011). Towards Algebraic Cryptanalysis of HFE Challenge 2. International Journal of Multimedia and Ubiquitous Engineering. 6. 123-131. 10.1007/978-3-642-23141-4_12.
24. Patarin J. (1995). Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88. In: Coppersmith D. (eds) Advances in Cryptology — CRYPT0' 95. CRYPTO 1995. Lecture Notes in Computer Science, vol 963. Springer, Berlin, Heidelberg.
25. Patarin J. (1996). Asymmetric Cryptography with a Hidden Monomial. In: Koblitz N. (eds) Advances in Cryptology — CRYPTO '96. CRYPTO 1996. Lecture Notes in Computer Science, vol 1109. Springer, Berlin, Heidelberg.
26. Patarin J., Courtois N., Goubin L. (2001). QUARTZ, 128-Bit Long Digital Signatures. In: Naccache D. (eds) Topics in Cryptology — CT-RSA 2001. CT-RSA 2001. Lecture Notes in Computer Science, vol 2020. Springer, Berlin, Heidelberg.
27. Patarin J. (1996). Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In: Maurer U. (eds) Advances in Cryptology — EUROCRYPT '96. EUROCRYPT 1996. Lecture Notes in Computer Science, vol 1070. Springer, Berlin, Heidelberg.
28. Patarin, J. (1999). La Cryptographie Multivariable. Mémoire d'habilitation à diriger des recherches de l'Université Paris, 7, 354.
29. Petzoldt, A. (2017). On the Complexity of the Hybrid Approach on HFE$v-$. IACR Cryptology ePrint Archive, 2017, 1135.
30. Sturmfels, B. (2005). WHAT IS... a Grobner Basis?. Notices-American Mathematical Society, 52(10), 1199.
31. Vates J., Smith-Tone D. (2017.) Key Recovery Attack for All Parameters of HFE−. In: Lange T., Takagi T. (eds) Post-Quantum Cryptography. PQCrypto 2017. Lecture Notes in Computer Science, vol 10346. Springer, Cham.

Appendices

A "Slow-Quartz" signature (77 bit signature)

Quartz is a HFE$v-$ signature scheme introduced in [26] in 2001. It generates signatures of 128 bits. The base field is $\mathbb{F}_2$, with $n = 103$, $D = 129$, $a = 3$ equations removed, and $v = 4$ external vinegar variables.

Quartz was designed in 2001 in order to have an expected security in 2^{80}. It seems that at present the best known cryptanalysis still have a complexity of about 2^{80} (see [16]).

More precisely, if we evaluate the direct Gröbner attack with our formula above we find $r = 8$ (since $D = 129$) and $d_{reg} = 7$. Therefore the complexity of this attack is about $\binom{110}{7}^{\omega}$. This gives $2^{82.7}$ with $\omega = 2.37$ (and $2^{69.8}$ with $\omega = 2$, however $\omega = 2.37$ is probably more realistic). The complexity to find the secret key is much higher because the MinRank step can be evaluated to be about $2^{63.\omega}$.

At present (2020), with Magma, the time to compute the roots of a polynomial of degree 129 on $\mathbb{F}_{2^{103}}$ is $0,095$ s. With the improved software of the GeMMS team this time is 23 times smaller: 4.1 ms (cf Appendix C).

Quartz for encryption

Quartz was designed as a signature scheme. However, this time of 4.1 ms is so small that we can imagine to use the scheme as an encryption scheme. Then, the time to decrypt will be $2^7 \times 0,095$ s $= 12.1$ s with Magma and about 0.5 s with the improved software. Nevertheless, in this paper we are mainly interested in short signature, so we won't say much more about encryption.

Quartz for signature

Quartz uses a mode of operation called "Feistel-Patarin" (see [12]) in order to be able to sign messages of any length. For this, will have to perform the root finding at least 4 times, and more precisely in average about 6 times because the probability to have at least one root is about 63% (i.e. 1- 1/e).

Remark 10. Sometimes the value $Y = \text{SlowHash}(M)$ that we want to sign has no solution, and we must try again. The probability of this to occur is about $1/e$, i.e. about $36,79\%$. Then we can compute $Y = \text{SlowHash}(U||M)$ where U takes the value 0, then 1, 2 etc. until we find a solution. We can include U in the signature in order to accelerate the verification of the signature (this is what is done in Quartz and GeMMS) but since we want the ultra-short signatures in this paper, we will prefer to keep U, so the various values of U will be tested when we check the signature.

In this section we will modify the scheme by using our "Slow hash" mode instead of the Feistel-Patarin mode. will keep all the parameters of Quartz except that we will use another mode of operation that we will call "Slow mode". The time to sign will then be much bigger, but the length of the signature will be shorter. (We will also have to perform the root finding 4 times less, but this point is not essential).

As seen above, the "Slow mode" of operations uses two ideas:

1. We will use a slow Hash function in order to avoid the birthday paradox(i.e. the complexity to find collisions) on the hash values.
2. We will not give all the bits that are usually the HFE$v-$ signature. The missing bits will be found by using the public keys and Gröbner bases algorithms (or a hybrid mix of Gröbner bases algorithms and exhaustive search).

We have performed some computer tests by using the free access Magma calculator (Magma V2.25-4). This free Magma is limited to 2 minutes of computation and 300 M Bytes of RAM but this is not a problem for us since it models the computer that will generate the signature and the computer that will check the signature. With Magma it is easy to compute polynomials on finite fields, and the F4 Gröbner algorithm of Jean-Charles Faugère is included.

Here are the time given by Magma to solve a system of $n = 100$ quadratic equations over $\mathbb{F}_2$ with f variables:

$f = 15$ variables: 8,13 ms
$f = 16$ variables: 10,6 ms
$f = 17$ variables: 16,5 ms
$f = 18$ variables: 26,8 ms
$f = 19$ variables: 45 ms
$f = 20$ variables: 75,0 ms
$f = 21$ variables: 0,125 s
$f = 22$ variables: 0,198 s
$f = 23$ variables: 0,32 s
$f = 24$ variables: 0,89 s (or 0,64 s from $f = 23$)

$f = 25$ variables: 1,7 s (or 1,28 s from $f = 23$)
$f = 26$ variables: 3,36 s (or 2,56 s from $f = 23$)
$f = 27$ variables: 6,53 s (or 5,12 s from $f = 23$)
$f = 28$ variables: 12,6 s (or 10,2 s from $f = 23$)
$f = 29$ variables: 23,9 s (or 20,4 s from $f = 23$)
$f = 30$ variables: 40,9 s

(To compute these 30 variables the best way is maybe here to do exhaustive search on 7 variables and to use the fact that we need only 0,32 s to find the remaining 23 variables. We will use like this less memory for about the same time).

We see that if we accept a time of 41 s to check the signature, then we can have a signature of about 103 (value of n) + 4 (external vinegar) – 30 = 77 bits. (Instead of 128 bits for the initial Quartz signature).

However in order to be able to sign messages of any length we must now see if the results of the "Slow Hash mode" are acceptable.

Here we have:

$g = \log_2(100.100.101/2) = 18.94$, $\Delta = m \log_2(q) + g - 123 = -4.06$

So with our "Slow mode" we will need to use a hash of $2^{4.06}$ minutes i.e. about 16 minutes. Or, we use a hash of 1 minute and 16 independent public keys. Or we use a hash of 4 minutes and 4 independent public keys.

We can summarize our results for "Slow-Quartz" and compare it with Quartz by this table of values (we present here the solution with 16 independent public keys):

Quartz ($f = 0$, Feistel-Patarin mode)
Length of the public key: 70.5 kBytes
Time to sign: 6×4.1 ms = 24.6 ms in average.
Time to check the signature: less than 1 ms
Signature length: 128 bits
Expected security: 2^{80} (with $\omega = 2.37$)

Slow-Quartz ($f = 30$, Slow Hash and Memory trade-off mode)
Length of the public key: 16×70.5 kBytes = 1.1 MBytes
Time to sign: 1.5×4.1 ms + 1 minute of slow hash.
Time to check the signature: 41 s + 1 minute of slow hash.
Signature length: 77 bits
Expected security: 2^{80} (with $\omega = 2.37$)

Signatures of 77 bits is not too far from the best signature schemes of this paper. However, in this paper we design schemes with even smaller signatures. Moreover we designed smaller signatures with expected security 2^{80} even if we assume $\omega = 2$, and sometimes with smaller public keys.

B "Slow-GeMMS-128" signatures

GeMMS-128 (cf [11]) is a signature scheme candidate to the Post-Quantum NIST competition, with an expected security in 2^{128} on classical (non-quantum) computers based on HFE$v-$. Its parameters are: $q = 2$, $n = 174$, $m = 162$, $a = 12$, $v = 12$, $D = 513$ (so $r = 10$). It uses 4 rounds of the Feistel-Patarin mode of operation. The length of the signature is 258 bits. The length of the public key is 352 KBytes.

Let us see here what we obtain if we use our new "Slow Hash and Multiple public keys" mode of operation instead of the Feistel-Patarin mode, with exactly the same HFE$v-$ parameters.

Here $\Delta = m - 219 + \log_2(m^3/2) = 162 - 219 + 21 = -36$. This value of Δ is so negative that it gives unrealistic size for the public key: about 22 528 Terabyte, with computations of about 1 minute, and signature size of 152 bits (with $f = 34$ variables to find from the 162 quadratic equations). Some trade-offs exist in order to reduce this huge public key, for example by spending more time, or by using more than one computer in a given time (parallelisation), but here it seem to be more reasonable to increases the number m of equations in order to have a smaller Δ: the parameters of GeMMS-128 are well adapted to the Feistel-Patarin mode but not well adapted to our new mode of operation.

For example, as we have seen in this paper, with $q = 2$, $m = 198$, $a + v = 11$, we have a signature of 173 bits with an expected security in 2^{131}. Here the length of the signature, 173 bits, is larger than what we would have with Slow-GeMMS (152 bits with a huge public key) but significatively smaller than what we have with the classical GeMMS (258 bits).

C Berlekamp algorithm, and roots finding with Magma

Berlekamp algorithm. The Berlekamp algorithm is generally what we use to find the roots of a polynomial of degree D in $\mathbb{F}_{q^n}$ for typical cryptographic values. There are two parts in this algorithm: the computation of the Frobenius application, and the computation of a GCD. For the Frobenius the complexity (when D is larger than n) is in $O(nD\log^2(D))$. For the GCD the complexity is in $O(nD^2)$. (Asymptotically the complexity is about in $O(nD)$ but from a practical point of view the asymptotic algorithms are not expected to be useful for our parameters).

Here are in the tables below, the times taken by Magma to find the roots of a polynomial of degree D on $\mathbb{F}_{q^n}$.

Remark 11. In Table 13, $D = 129$ (Quartz parameter) gives 95 ms. If we use the improved software of the GeMMS team the time here becomes 4.1 ms (i.e. 23 times faster than Magma) or 11,2 MegaCycles.

Table 13. Time to compute roots of a polynomial of degree D, $q = 2$, $n = 103$.

D	5	9	17	33	65	129	257	513
Time (ms)	1.13	2.06	5.05	13.3	40.8	95	260	430
D	1025	2049	4097	8193	16385	32769		
Time (s)	1.1	4.4	9.8	21.0	45.5	98.3		

Table 14. Time to compute roots of a polynomial of degree D, $q = 4$, $n = 47$.

D	17	65	257	1025
Time (ms)	4.29	31.2	140	650
D	4097	16385	65537	
Time (s)	2.17	13.4	80.6	

Table 15. Time to compute roots of a polynomial of degree D, $q = 5$, $n = 43$.

D	6	26	126
Time (ms)	20.2	76	840
D	626	3126	15626
Time (s)	4.5	17.4	117

Table 16. Time to compute roots of a polynomial of degree D, $q = 7$, $n = 40$.

D	8	50	344
Time (ms)	9.35	105	1630
D	2402	16808	
Time (s)	10.2	110	

Table 17. Time to compute roots of a polynomial of degree D, $q = 11$, $n = 35$.

D	12	122	1332	14642
Time	15 ms	360 ms	11.7 s	86.9 s

Table 18. Time to compute roots of a polynomial of degree D, $q = 13$, $n = 35$.

D	14	170	2198
Time	23 ms	540 ms	11.4 s

Table 19. Time to compute roots of a polynomial of degree D, $q = 17$, $n = 33$.

D	18	290	4914
Time	55 ms	1680 ms	29.2 s

D Our Magma simulation on the direct attack on HFE$v-$ with public equations of degree 2

We work on $\mathbb{F}_q$. m is the number of quadratic equations. v is the number of vinegar variables (so we have $m+a+v$ variables). a is the number of removed equations (initially we had $m+a$ equations and now m).

For a random system of quadratic equations (random, not HFE), the degree of regularity, that we denote d_{regmax} is like this:

- If $n < q$ then $d_{regmax} = \mathrm{n} + 1$
- If $n \geq q$ then we estimate that $d_{regmax} = q+1+(n-q-1).\alpha$ with $\alpha \approx 0.19$ for $q = 3$, $\alpha \approx 0.2$ for $q = 5$, $\alpha \approx 0.33$ for $q = 7$, $\alpha \approx 0.40$ for $q = 13$, $\alpha \approx 0.41$ for $q = 17$.

For HFE$v-$ systems we have obtain these results with Magma:

$q = 3$, $m = 17$, $D = 4$ (so here $r = 2$, and we have 2 monomials X^2 and X^4). Here the degree of regularity is $d_{regmax} = 7$.

$v = 0$ and $a = 0 : d_{reg} = 4$
$v = 0$ and $a = 1 : d_{reg} = 4$
$v = 0$ and $a = 2 : d_{reg} = 5$
$v = 0$ and $a = 3 : d_{reg} = 6$
$v = 1$ and $a = 0 : d_{reg} = 4$
$v = 2$ and $a = 0 : d_{reg} = 5$
$v = 3$ and $a = 0 : d_{reg} = 6$

$q = 3$, $m = 16$, $a = v = 0$ (nude HFE)

$r = 3 : d_{reg} = 4$
$r = 4 : d_{reg} = 5$
$r = 5 : d_{reg} = 6$ (Max)

$q = 4$, $m = 14$, $a = v = 0$ (nude HFE)

$r = 3 : d_{reg} = 6$
$r = 4 : d_{reg} = 6$
$r = 5 : d_{reg} = 7$ (Max)

$q = 5$, $m = 13$, $D = 6$ (we have here 2 monomials X^2 and X^6). Here the degree of regularity is $d_{regmax} = 8$.

$v = 0$ and $a = 0 : d_{reg} = 6$
$v = 0$ and $a = 1 : d_{reg} = 7$
$v = 0$ and $a = 2 : d_{reg} = 8$ (Max)
$v = 1$ and $a = 0 : d_{reg} = 7$
$v = 2$ and $a = 0 : d_{reg} = 8$ (Max)
$v = 1$ and $a = 1 : d_{reg} = 8$ (Max)

E Some Magma simulation on the direct attack on HFE$v-$ with public equations of degree 3

$q = 3$, $m = 11$, $a = v = 0$ (nude HFE with public equations of degree 3)

$r = 2 : d_{reg} = 4$
$r = 3 : d_{reg} = 5$
$r = 4 : d_{reg} = 6$
$r = 5 : d_{reg} = 7$ (Max)

$q = 4$, $m = 10$, $a = v = 0$ (nude HFE with public equations of degree 3)

$r = 2 : d_{reg} = 5$
$r = 3 : d_{reg} = 6$
$r = 4 : d_{reg} = 8$
$r = 5 : d_{reg} = 8$
$r = 6 : d_{reg} = 9$ (Max)