Ultra-Short Multivariate Public Key Signatures – with Public Keys of Degree 2 –

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 and propose multivariate schemes with "ultra-short" signatures and public keys of degree 2. In order to do so, we consider that signing a message and verifying a signature could require up to 1 minute of computation on a modern personal computer. Of course, times around 1 second can also be considered but at the cost of 6 to 10 additional bits in the signatures; and, more generally, a trade-off may be found between computation time and signature size, it usually depends on the applications one is targeting.

Despite the fact that a time of 1 minute is far bigger than the time required by general purpose multivariate-based signature schemes, such as Rainbow, GeMMS, and Quartz, it enables us to reach ultra-short signature lengths; for instance, around 70 bit-long signatures for a security of 80 bits.

The main issue that arises when one wants to build a signature scheme with ultra-short signatures is the need to avoid the birthday paradox attack.

In a first part, we describe generic and specific attacks against multivariate public key signature schemes and use them to derive the minimal parameters that an ultra-short signature scheme could have. In a second part, we give explicit examples of such ultra-short signature schemes. We will use "Nude HFE" algorithms (i.e. HFE without perturbation) with public keys of degree 2 for 80, 90 and 100 bit security. We have considered larger security parameters in [1] with public keys in degree 3. Recent progress has been made on attacking the MinRank problem in [3] and attacking HFE$v-$ in [23]. These potential threats against multivariate signature schemes have been taken into account in this paper.

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 (non-quantum) factorization algorithm (General Number Field Sieve [8]) whose complexity is subexponential, 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. Using elliptic curves, signatures are smaller: about 240 or 320 bits for a security in 80 bits, and about 384 or 512 for a security in 128 bits.

Nowadays, many cryptographers focus more on post-quantum cryptography, that is to say on crypto-systems 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 schemes 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 [17]. It was later broken by Patarin ([18]) who then suggested a way to fix it with the Hidden Field Equations (HFE) scheme in 1999 ([19]). Following it, a lot of variations of this scheme were proposed, for instance HFE$v-$ (also in [20]), Quartz [21], and GeMSS [9].

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 [11] is a multivariate signature scheme designed by Ding from the “Unbalanced Oil and Vinegar” scheme [16]. With multivariate-based cryptography, it seems possible to have short signatures, for example for a 128 bit security level, Rainbow provides 528 bit-long signatures, and GeMSS 256 bit-long signatures; for a 80 bit security level, Quartz provides 128 bit-long signatures.

In this paper, we will obtain even shorter signatures.

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 that computes hash values and one that checks either a pair message-signature is valid or not ; but each request to one of those oracles has a cost. We express this cost in term of a minimal number of operations 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 6, 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 the cost of a single operation, as long as the access to the hash oracle has a non-negligible cost.

Our contribution. In this paper, 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. Indeed, for a 80 bit 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 a message 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.

Moreover, this paper gives explicit examples of ultra-short multivariate signature schemes based on HFE variants. We provide many parameters for these schemes according to different security levels (from 80 to 256 bits) and different choices of finite fields (from $\mathbb{F}_2$ to $\mathbb{F}_{17}$).

Structure of our paper. In Section 2, we describe three types of generic attacks against multivariate signature schemes (namely Types 1, 2 and 3 attacks) and we derive minimal lengths for our ultra-short signatures from them.

Then, in Section 3, we propose concrete designs for ultra-short multivariate signature schemes based on HFE variants and we study their specific attacks as well (we named them Type 4 attacks).

2 Generic Attacks

In this section, we describe the four main types of attacks that have to be considered to design a signature scheme. The three first types are generic attacks against the multivariate signature schemes, whereas the last one is specific to the schemes using HFE variants.

While describing those attacks, we will look for the shortest length L in bit a signature can have for a given level of security of z bits. In what follows, a z bit security means classically that the computational power required by an attacker can not exceed 2^z operations. In the same spirit and to ease the wording, we will say that a computation, the evaluation of a function or an algorithm for instance, has a b-bit cost, when it requires 2^b operations. Similarly, a b-bit list or enumeration has 2^b elements.

Here are the aforementioned four types of attacks:

- **Type 1:** refers to attacks that work against any public key signature scheme whose signatures are L bit-long.
- **Type 2:** refers to attacks against signature schemes where the verification of a signature S of a message M is done by checking if $F(S) = H(M)$, where

F is a public function which can be evaluated efficiently and H is an hash function.

Note that this verification process is different from the more classical one where a function F takes the couple (S, M) as an argument and returns 1 if S is a valid signature of the message M and 0 otherwise.

- **Type 3:** refers to attacks that work when F is a set of m multivariate quadratic equations over a finite field $\mathbb{F}_q$.
- **Type 4:** refers to attacks against signature schemes based on HFE with some perturbations. Usually those perturbations, such as "v" (vinegar) or "$-$" (minus), are used in order to introduce a secret trapdoor in F.

This Section focuses on the 3 first types, the attacks of Type 4 are discussed in Section 3.2.

2.1 Type 1 attacks

Let $G(S, M)$ be the signature verification function which outputs 1 if S is a valid signature of message M, and 0 otherwise. Since G is public, a generic attack to find a valid signature S of a message M is to try sufficiently enough values for S until $G(S, M)$ outputs 1.

Let g be the bit cost of G and L the length of the signature S in bits. Assuming that the signature scheme is sound, any message M should admit at least one signature, so it can be found by an L-bit search, with total cost $L + g$. So, in order to avoid this attack and if we want a z bit security, we must have: $L \geq z - g$.

For example, if we want signatures smaller than 80 bits for a 80 bit security, we need the evaluation for G to be *heavy* (its cost must be high enough). This is the core of this paper, that is to say to design multivariate signature schemes relying on heavy modes of operation.

Let's consider that a user has a computational acceptable work of, for instance, 38 bits (1 minute of computation for a nowadays personal computer). Thus, for a 80 bit security, the length L of the signature in bits will have to be greater or equal to $80 - 38 = 42$ bits.

The aim of this paper is to study what kind of value L (larger than this bound, but as small as possible) we can choose and to design explicit schemes with it.

Remark 1. If the signature has only, let us say 60 bits for example, the birthday paradox states that if 2^{30} messages are signed, two messages will have, with a high probability, the same signature. This may not be a problem since these two messages have been actually signed by the legitimate user, thus there is no "dangerous" attacks based on the fact that they have the same signature. Moreover, it is always possible to avoid this issue by asking the legitimate user not to sign more than about one billion messages with the same public key.

2.2 Type 2 attacks and our modes of operation

In the multivariate signature schemes studied in this paper, one checks if S is a valid signature of a message M using an equality like this:

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

where G and H are public (H stands for a hash function, and G is given by the public key). This offers other possibilities of generic attacks detailed below.

Collisions on $\boldsymbol{H}$. First, an attacker could look for collision on H. If one can find two messages M_1 and M_2 with the same hash value, one could ask for the signature of M_1 and obtain the signature of M_2. Let E be the 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 has a h bit cost. This birthday attack proceeds like this: we enumerate a list of a bit size of values M and we store M at the address $H(M)$, hence a total cost of $a + h$. Then we will obtain with high probability (thanks to the birthday paradox) that two values share the same address (i.e. we have find a collision $H(M_1) = H(M_2)$) as soon as $a > E/2$. Let z be the desired security level. For any attacker, we have $a + h < z$. Therefore, in order to avoid this collision attack on H, we must have:

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

Note that there also exists time and memory trade-off: with more time we can use less memory; however, in this paper, we focus only on the time complexity.

Recall that in this paper, the hash function H has a cost of about 37 bits. So, if $h = 37$ and if we target an 80 bit security, then E has to be greater or equal to $2 \times 80 - 2 \times 37 = 86$ bits.

However, the length L of the signature can be smaller than the outputs size E as we will see below.

Collisions of the type $\boldsymbol{G(S) = H(M)}$. Second, an attacker could look for collision of the type $G(S) = H(M)$. More precisely, by choosing a list of random S of a bit size, and a list of random messages M of b bit size, an attacker would find a collision $G(S) = H(M)$ with high probability as soon as $a + b > E$.

Let g be the bit cost of G, and let h be the bit cost of H. Since z is the security parameter, for any attacker we have $a + g < z$ and $b + h < z$. Therefore, the attack will be impossible when

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

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

Note that classical signature schemes such as RSA or ECDSA are not threatened by this kind of attack since their signature lengths, way bigger than for multivariate-based cryptography, make the complexity of the birthday paradox attack larger than any other attacks.

To avoid this attack, which was first mentioned in [21], several ways have been designed, usually one calls them "modes of operation". Mainly, there are the Feistel-Patarin, the Gui, the UOV, and the Dragon mode of operations. For instance, the Feistel-Patarin mode of operation is the one used in the NIST Post-Quantum candidate GeMSS, and in Quartz.

Nevertheless, we will not use most of the aforementioned modes of operation since they usually raise the size of the signatures, which is what we want to avoid for ultra-short signature schemes. We will use the Feistel-Patarin mode which becomes interesting when $\Delta \ll 0$.

This is why we designed new modes of operations, namely the "Multiple independent public keys", the "Slow" mode of operation, and both of them mixed. The Slow mode of operation relies on the use of a slow hash function, that is to say an hash function which require around 1 minute to be computed, or has approximately a 37 bit work cost. Indeed, this new mode of operation perfectly fulfills the requirement of ultra-short signature schemes since it does not raise the length of the signature and it is compatible with the 1 minute requirement described above.

The two following paragraphs give details about those new modes of operation and the possible trade-off mixing them. Moreover, the first one gives explicit detail on how to build easily a slow hash function from any existing hash function.

Heavy mode of operation. The Heavy mode of operation relies on the use of a hash function H that requires a "lot" of operations. It can be built using iterations of a standard hash function such as SHA-3 or SHA-256. These functions on optimized platforms can operate at a rate of 13 cycles per byte, which means that for a data of 72 bytes, (36 words of 16 bits) it requires about 2^{10} cycles (so a 10 bit work). For instance, if one wants to build a hash function with 37 bit work, (i.e. around 1 minute to compute a hash value), one needs to consider using 2^{27} iterations of a standard hash function. To prevent an attacker from speeding the computation, one may consider to tweak the hash function, so the ith iteration may be for instance $H_i(x) = \text{SHA-3}(x\|i)$, and $H(x) = H_{2^{27}}(\dots H_2(H_1(x))\dots)$. However, we may also consider a "parallelizable" version such as:
$H(x) = H_1(x) \bigoplus H_2(x) \bigoplus \cdots \bigoplus H_{2^{27}}(x)$, that enables a trade-off between time and computing power. Whatever the choice, what matters is that an attacker can not perform one evaluation of H in fewer operations than the legitimate user.

2.3 Type 3 attacks

This section describes Type 3 attacks, that is to say, when the attacker wants to forge a signature by solving a polynomial system of equations.

To forge a valid signature for a message M, an attacker wants to find a string of bits S such that $G(S) = H(M)$. Since G and H are public (recall that

G is a set of m quadratic multivariate polynomials given by the public key), the attacker is left with solving a system of multivariate equations. In other words, solving this system enables the attacker to find $G^{-1}(H(M))$.

Of course, G is not a random set of multivariate equations, otherwise the legitimate user would not be able to invert it. Indeed, G has a secret structure, hidden as much as possible, which gives it what is called a *trapdoor*. With this trapdoor, which is part of the private key, the legitimate user can invert G. The purpose of hidding this special structure is that G looks random to the attacker who can only invert it using generic polynomial system solvers such as Gröbner basis algorithms. One sometimes speaks of "perfect trapdoor" to refer to the hypothetical case where G would be perfectly random.

Thus, the complexity of solving a random system of multivariate equations gives an estimation of the complexity to forge a multivariate signature.

This section first describes the classic way to solve a polynomial system, namely Gröbner basis. Then, from the complexities obtained, we derive minimum sizes multivariate signatures could have.

Remark 2 (Complexity of evaluating G).

Since G, the set of public equations, is composed of m dense quadratic equations in about m variables, the cost of one evaluation can be estimated as $\frac{m^3}{2}$. However, an evaluation can be done quicker when it is not done from scratch, but when reuses another one and only few bits in the input change. So the average cost could be as low as m^2. It is probably possible to do even better, so in our analysis, we will consider that this cost is an arbitrarily low constant.

Polynomial system solving using Gröbner basis. Let us briefly introduce Gröbner basis techniques that are fundamental tools for solving systems of multivariate polynomial equations. Readers may refer to [22, 4, 5] for further details.

The computation of a Gröbner basis is a non-linear generalization of Euclid's algorithm for the greatest common divisor, 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 [13] and F5 [14] 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 [2, 14] for more details).

The complexity of a Gröbner basis computation detailed in [2, 14] 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 [2].

Polynomial system solving using an hybrid approach. In order to speed up the computation of a Gröbner basis, the authors of [6] combine an exhaustive search for some variables and a Gröbner basis computation for the remaining variables. Thus, this approach is called "hybrid".

The complexity of this hybrid approach is:

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

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, counted with multiplicity, of the system in $\overline{\mathbb{F}_q}$, and $\mathcal{C}_{F5}$ is the complexity of the F5 algorithm ([14]).

When the trade-off factor k is well chosen, the hybrid approach can be the most efficient algorithm for solving polynomial systems.

Choice of the constant of linear algebra $\boldsymbol{\omega}$. As seen in the previous formulas, the constant of linear algebra ω plays an important role in the complexities of Gröbner basis algorithms. From a practical point of view, we should choose $\omega = 2.81$ due to Strassen algorithm.

From a theoritical point of view, there exists an algorithm with $\omega \approx 2.37$ but it would not be efficient in practice due to huge constant terms in its complexity.

However, in order to take into account the fact that the linear systems arising in the computation of a Gröbner basis are usually really sparse, one often consider that $\omega = 2$.

Size of the signature with a perfect trapdoor. As we did in the previous Sections for the Type 1 and 2 attacks, we can now look for minimal parameters for ultra-short multivariate signature schemes according to Type 3 attacks. All those information are gathered in Table 1, the rest of this Section is dedicated to the explanation on how to read it.

From now on, let us take the first cell at the top left corner of Table 1as an example. In this table, m stands for the number of equations and it is equal to the number of variables. For instance, the first cell of the table gives that the computational power needed to solve a random quadratic system of 94 multivariate equations in 94 unknowns is 2^{80}.

As the verifier has a computational power up to 2^{37} operations (i.e. around 1 minute), the signature can be a part S' of the signature S, containing only $m-f$ elements in the considered finite field. The value f (30 in our example cell) was chosen so that recovering the signature S from S' takes around 1 minute for the verifier according to our `MAGMA` implementation.

Table 1. Generic degree 2 table: 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 $\omega = \mathbf{2.37}$	$\mathbf{2^{80}}$	$\mathbf{2^{90}}$	$\mathbf{2^{100}}$	$\mathbf{2^{128}}$	$\mathbf{2^{192}}$	$\mathbf{2^{256}}$
$\mathbf{q = 2}$	$m = 86$ $f = 29$ $L = 57+$	$m = 100$ $f = 30$ $L = 70+$	$m = 112$ $f = 31$ $L = 81+$	$m = 145$ $f = 33$ $L = 112+$	$m = 218$ $f = 37$ $L = 181+$	$m = 290$ $f = 42$ $L = 248+$
$\mathbf{q = 4}$	$m = 43$ $f = 21$ $L = 44+$	$m = 50$ $f = 22$ $L = 56+$	$m = 56$ $f = 23$ $L = 66+$	$m = 73$ $f = 26$ $L = 94+$	$m = 113$ $f = 30$ $L = 166+$	$m = 154$ $f = 32$ $L = 244+$
$\mathbf{q = 5}$	$m = 40$ $f = 20$ $L = 47+$	$m = 45$ $f = 20$ $L = 59+$	$m = 50$ $f = 21$ $L = 68+$	$m = 66$ $f = 24$ $L = 98+$	$m = 102$ $f = 27$ $L = 175+$	$m = 139$ $f = 30$ $L = 254+$
$\mathbf{q = 7}$	$m = 35$ $f = 19$ $L = 45+$	$m = 40$ $f = 20$ $L = 57+$	$m = 45$ $f = 20$ $L = 71+$	$m = 59$ $f = 22$ $L = 104+$	$m = 91$ $f = 26$ $L = 183+$	$m = 124$ $f = 29$ $L = 267+$
$\mathbf{q = 8}$	$m = 34$ $f = 19$ $L = 45+$	$m = 39$ $f = 19$ $L = 60+$	$m = 43$ $f = 20$ $L = 69+$	$m = 57$ $f = 23$ $L = 102+$	$m = 88$ $f = 27$ $L = 183+$	$m = 119$ $f = 29$ $L = 270+$
$\mathbf{q = 11}$	$m = 32$ $f = 18$ $L = 49+$	$m = 36$ $f = 19$ $L = 59+$	$m = 40$ $f = 20$ $L = 70$	$m = 52$ $f = 21$ $L = 108+$	$m = 81$ $f = 26$ $L = 191+$	$m = 111$ $f = 28$ $L = 288+$
$\mathbf{q = 13}$	$m = 31$ $f = 18$ $L = 49+$	$m = 35$ $f = 19$ $L = 60+$	$m = 39$ $f = 20$ $L = 71+$	$m = 51$ $f = 21$ $L = 112+$	$m = 79$ $f = 25$ $L = 200+$	$m = 107$ $f = 28$ $L = 293+$
$\mathbf{q = 16}$	$m = 30$ $f = 18$ $L = 48+$	$m = 34$ $f = 18$ $L = 64+$	$m = 38$ $f = 19$ $L = 76+$	$m = 50$ $f = 21$ $L = 116+$	$m = 77$ $f = 26$ $L = 204+$	$m = 104$ $f = 28$ $L = 304+$
$\mathbf{q = 17}$	$m = 29$ $f = 17$ $L = 50+$	$m = 33$ $f = 18$ $L = 62+$	$m = 37$ $f = 19$ $L = 74+$	$m = 49$ $f = 21$ $L = 115+$	$m = 76$ $f = 25$ $L = 209+$	$m = 103$ $f = 28$ $L = 307+$

Knowing m and f, one finds the length of the signature in bits by computing:

$$L := (m - f)\log_2(q).$$

In our example cell, it is $94 - 30 = 64$ bits.

Nevertheless, as described in the previous Section, one should be careful that the value of Δ is not too small, in fact, if it is, the multivariate signature scheme will have a public key far too large. Recall that in this paper we are looking for ultra-short signature scheme whose verification and signing time are reasonable, but we also want the public key not to be too large. In our example cell, the value of $\Delta = -10.4$ is too large.

The purpose of the arrows and the new values on the right (m', f', L') is to set new parameters with the same level of security, this time with Δ as close to zero as possible. Note that when Δ is positive, there is no need to raise the parameters, so there is no right part in the cell. In our example cell, we need to increase $m = 94$ to 104 in order to get a value of Δ closer to 0 (recall that this is done to reduce the size of the public key). With this new $m' = 104$, one needs to update the value of f to $f' = 31$ and L to $L' = 73$.

Finally, the "+" symbols by the signature length L' means that this length is a lower bound which will naturally go up while taking into account all the other possible attacks.

Nevertheless, this table gives a glimpse of the minimal parameters ultra-short multivariate signature could have in the general case.

Remark 3. First, a surprising fact is that $q = 2$ does not appear so far to be the best choice of finite field for ultra-short signatures. Second, the lengths L for various finite fields ($q = 5, 7, 11, \ldots$) are very similar.

3 HFE and our parameters

In this Section, we describe the classic HFE algorithm, called "Nude" HFE and its variants. Then we describe the best currently known attacks against it, and finally, we use them to derive parameters for our ultra-short multivariate schemes.

3.1 Nude HFE, and HFE Variants

Description of (nude) HFE. Hidden field Equations (HFE) algorithm was proposed by Patarin at Eurocrypt [19] to repair the algorithm C* of Matsumoto and Imai [17]. 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:

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

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

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

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

$$F = \sum_{0 \le i \le j \le n}^{q^i + q^j \le D} \alpha_{i,j} X^{q^i + q^j} + \sum_{0 \le i \le n}^{q^i \le D} \beta_i X^{q^i} + \gamma, \qquad (2)$$

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 $\mathcal{F} = \phi \circ \mathcal{F}^* \circ \phi^{-1}$:

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

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, one 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 is difficult to compute the inverse of $\mathcal{P}$ when its decomposition remains secret. Some super-polynomial attacks are known on “Nude” HFE, this is why we will see some perturbations to increase its security.

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

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, one needs to try on average about 1.5 values in order to find a signature.

However, in $\mathbb{F}_{q^n}$, when q is odd, and for an homogeneous random polynomial of degree 2, the probability is only 31.6% (i.e. 2 times less). Therefore, in this case, one needs to try on average about 3 values in order to find a signature. The reason is that, 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 generally choose a non-homogeneous polynomial. Then the secret linear transformations S and T defined in HFE will be chosen to be affine (that is to say linear with constants).

We expect that this non-homogeneous choice (when q is odd) does not create a security problem; 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, we could come back to homogeneous solutions, with a probability to be invertible divided by 2.

3.2 The best known attacks against nude HFE

In this section, we briefly present the best known attacks against HFE; this will enable us to choose our parameters accordingly. There are basically 3 kinds of attacks against HFE schemes:

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

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 a differential attack in [7].

Thus, 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.

Direct attacks. In these attacks, like in [12] or in [15], one tries to solve the public equations for a given value Y (i.e. for a given message, one tries to find a valid signature) by using Gröbner algorithms like F4 or F5.

Let D be the degree of the HFE polynomial. Let

$$r = \lfloor \log_q(D-1) \rfloor + 1,$$

it reflects the rank of the quadratic form associated to the HFE polynomial.

The order of magnitude of the complexity of this attacks is

$$\binom{d+n}{d}^{\omega},$$

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 d.

In [12], an upper bound on this value d is given:

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

This upper bound is interesting since it is proved and it is valid for all values of q. It shows that the degree of regularity of HFE$v-$ public equations is often smaller than the one of a random system.

When $q = 2$, many experiments have been done. As long as d is not larger than the degree of regularity of a similar random systems (that we can compute from Hilbert series), these experiments show that d can be estimated by:

$$d = \left\lfloor \frac{r+7}{3} \right\rfloor$$

However, when q is different from 2, at present, very little is known about d. This is why we made our own experiments with `MAGMA`. It is difficult to see if q will be a factor as it is in Equation (3).

Nevertheless, we did observe that d starts with the value q (when $D = 2$), and when $q = 5$, it seems to increase by 1 almost each time r increases by 1; this is very different from what we have with $\mathbb{F}_2$ where r generally has to be increased by 3 to have the same effect.

So, we will assume that when $q \geq 5$, we have: $d \geq q - 2 + r$, as long as this value is smaller or equal to the degree of regularity one would have with similar random equations.

Remark 4. When $q \geq n$, we have noticed that $d = n + 1$. However, in our parameters we will always have $q < n$.

Key recovery attacks. Let n be the number of public equations, equal to the number of variables. let $r = \lceil \log_q(D) \rceil$.

Attacking a HFE instance with parameters (n, D, r) 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 $K := n$ 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 [24].

With the recent progress made for solving the MinRank problem in [3], this key recovery attack is currently the most threatening attack against HFE; previously it was the attack in [10]. [3] 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 $b < q$.

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

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 [3]. 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'.

3.3 Our ultra-short signature scheme and its parameters

In this Section, we present our ultra-short multivariate signature scheme based on nude HFE in degree 2 and on the heavy mode of operation described in Section 2.2. In order to do so, we will explain the choice of its parameters step by step.

Recall that we want our signatures to be as short as possible, with reasonable public key size, and verifiable in at most about 2 minutes on a modern personal computer (1 minutes for the slow hash and 1 minute to recover the whole signature).

Choice of the degree D of the HFE polynomial. First of all, due to many cryptanalytic results such as [7], the HFE polynomial must have at least 2 monomials. This is why our polynomials always have at least the two following monomials: X^2 and X^{1+q}.

Let us denote $r = \lceil \log_q(D) \rceil$. From our aforementioned `MAGMA` simulations, we derived the following maximum values for r (see Table 2) , they are chosen in order to spend about 1 minute for the verification process.

Variants. HFE$v-$ schemes corresponds to the use of only the two perturbations: $-$ (minus) and v (vinegar). The main advantage of those perturbations is that they only have a very limited impact on signatures sizes. However, due to the recent attacks on these variants (see [23]), we will not use these perturbations in this paper. We call “Nude HFE” a scheme with no additional perturbation.

Table 2. 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

Remark 5. It may look surprising to use Nude HFE since super-polynomials attacks are known against it, and therefore it is generally not recommended. However, since our purpose is very specific, this attack is not a problem. Indeed, we want a scheme with very short signatures that can be computed in 2^{37} operations for the legitimate users and 2^{80} computations for the non-legitimate users. One notices that as long as ω is greater than 2.47, the scheme keeps its security of 80 bits. This is generally considered as a very reasonable assumption.

Nevertheless, for larger security requirements, (128 or 256 bit security for example) the super polynomial attacks fall below the security level, and then nude HFE is not useful anymore, but some other multivariate schemes schemes still give very short signatures. See [1] for examples with public keys of degree 3.

4 Examples of parameters for Ultra-Short signatures

Table 3. Examples of parameters for a security in 2^{80} with $q = 2$ and approx. 1s. to check a signature. Nude HFE with L rounds.

m	86	86	92	92	100	100
r	16	17	16	17	16	17
D	32769	65537	32769	65537	32769	65537
h (seconds)	0.5	0.5	0.5	0.5	0.5	0.5
L	9	9	5	5	3	3
f	9	9	10	10	10	10
Exhaust. search ($m + 31$ bits)	117	117	123	123	131	131
Birthday attack	80.5	80.5	81.8	81.8	82.7	82.7
Gröbner	33.14ω	36.69ω	33.79ω	37.43ω	34.6ω	38.35ω
MinRank	83.6	87.6	83.9	87.9	84.3	88.3
Pub. Key Size (kBytes)	39.2	39.2	47.5	47.5	61.0	61.0
Time to sign (mn.)	2.25	5.6	1.25	3.13	0.75	1.88
Time to verify (s.)	1	1	1	1	1	1
Sig. size ($m - f$ bits)	77	77	82	82	90	90
Sig. size for 1mn verif. (bits)	71	71	76	76	84	84

In table 3 (resp. tables 4 and 5), we present possible parameters for 80 bits security (resp. 90 and 100 bits). We use here a Nude HFE with L rounds of Feistel-Patarin mode. See below.

Table 4. Examples of parameters for a security in 2^{90} with $q = 2$ and approx. 1s. to check a signature. Nude HFE with L rounds.

m	95	100	105	150	160	170
r	18	18	18	17	17	17
D	131073	131073	131073	65537	65537	65537
h (seconds)	0.5	0.5	0.5	0.5	0.5	0.5
L	12	6	4	1	1	1
f	8	9	10	27	28	29
Exhaust. search ($m + 31$ bits)	126	131	136	161	191	201
Birthday attack	90.0	90.1	90.2	90.5	95.5	100.5
Gröbner	37.79ω	38.35ω	38.9ω	42.8ω	43.6ω	44.2ω
MinRank	92.0	92.3	92.7	90.0	90.3	90.6
Pub. Key Size (kBytes)	53	62	72	207	252	302
Time to sign (mn.)	20.4	10.2	6.7	0.63	0.63	0.63
Time to verify (s.)	1	1	1	1	1	1
Sig. size ($m - f$ bits)	87	91	95	123	132	141
Sig. size for 1mn verif. (bits)	81	85	89	117	126	135

Table 5. Examples of parameters for a security in 2^{100} with $q = 2$ and approx. 1s. to check a signature.

m	110	240	600
r	20	19	18
D	524289	262145	131073
h	0.5	0.5	0.5
L	7	1	1
f	9	34	40
Exhaust. search ($m + 31$ bits)	141	271	631
Birthday attack	100.1	135.5	315.5
Gröbner	43.1ω	48.1ω	58.6ω
MinRank	100.7	100.1	100.0
Pub. Key Size	82 kB	847 kB	12.9 MB
Time to sign (mn.)	73	4.2	1.7
Time to verify (s.)	1	1	1
Sig. size ($m - f$ bits)	101	206	560
Sig. size for 1mn verif. (bits)	95	200	554

4.1 Feistel-Patarin mode of operation

Very often in our parameters, the number of bits in inputs or output of our functions F is smaller than twice the desired security level. Then, in order to avoid Birthday paradox attacks, we have to design a specific mode of operation. Usually, we will choose for this a so-called "Feisel-Patarin" scheme with L rounds. (Another possibility will be explained in Appendix C that will usually give short signatures but also huge public keys.) This is similar to what was used in Quartz (see [21]) or Gemss (see [9]). For L rounds, L values $Y_1, \dots, Y_L$ are derived from the hash of the message to sign, and then $L+1$ values $X_0, \dots, X_L$ are computed, satisfying $X_0 = 0$, and $F(X_i) = Y_i \bigoplus X_{i-1}$, for $i = 1, \dots, i = L$, and finally X_L is the signature. To verify a signature, the L values $Y_1, \dots, Y_L$ are computed from the message, the $L+1$ values $X_0, \dots, X_L$ are computed in reversed order, starting with X_L equal to the signature and then $X_{i-i} = F(X_i) \bigoplus Y_i$, for $i = L, \dots, 1$. The signature is valid if and only if X_0 is 0. For a security level λ, the number L of iterations must be chosen such that (see [21] or [9]) :

$$2^{m\frac{L}{L+1}} \geq 2^{\lambda}.$$

5 Signature size versus security level

For a security level of 80, 90 or 100 bits, Nude HFE with our specific mode of operation gives as far as we know the shortest signature size for a public key scheme. However, for a security level of 128 bits and above, Nude HFE becomes completely inefficient. This comes from the fact that there is a super-polynomial attack on HFE. Another multivariate scheme, UOV-Salt in degree 3 (cf. [1]) seems to be much more efficient with still very short signature sizes for these levels of security.

Table 6. Typical signature length and time to sign on a PC.

	Nude HFE Feistel-Patarin		UOV-SALT	
Security (bits)	Signature (bits)	Time to sign	Signature	Time to sign
80	71	2.25mn	162	0.04s
90	81	20.4mn	182	0.07s
90	117	38s		
100	101	73mn	203	0.1s
100	206	4.2mn		
128	600	2h	261	0.15s
192	Huge		386	0.2s
256	Huge		540	0.2s

6 Discussion about our security model

6.1 Examples of implementation of our security model together with our ultra-short signatures

As our signatures are "ultra-short", they could be stored in a tiny QR code, or combined in watermarking applications. One of their most valuable interests is also that they could, for instance, be spelled by a human to another one over the phone. Indeed, we can deal with 64 bit-long signatures easily when they are represented as 16 hexadecimal values such as: `45A5F352CDE20240`. This is less than the size of a strong Wifi-Box password that has to be typed into a device. Moreover, if one uses alphanumeric value (consisting in digits $0 \dots 9$, letters $a \dots z$ and $A \dots Z$), i.e. 62 characters, then 64 bits can be written with only 11 Alphanumeric characters such as: `4fDjK457GfD`.

Recall that, in this paper, all of our schemes rely on a security model where the oracles for the verification 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.

A concrete example involving non-negligible costs 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 forge a valid signature for a given message.

6.2 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, or in other words at cost one since it is a single operation), our scheme does not hold since it relies on slow verification and slow hash functions. 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 was free as well, this would improve significantly the birthday attack (such as the one described in Section 2).

If the access to the verification oracle is free, for a given l bit-long signature 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 parameter λ, we only need to adjust slightly the parameters. For example, for a security level of 2^{80}, with $q = 2$ (see sec.), 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 sec. since the lengths of the signatures are already larger than the security parameter λ.

Remark 6. 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), 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 page), a puzzle in 2^{16} operations would usually be enough.

7 Conclusion

At present the shortest public key signatures are obtained with multivariate signature schemes such as Rainbow, Quartz, or GeMMS. Usually, their lengths are between 128 and 256 bits, and the times needed to sign and verify them are in milliseconds.

In this paper, we have studied how to design shorter signatures as long as one accepts to spend about 1 minute to sign a message or to verify a signature. For instance, for a security of 2^{80} operations, we have designed a signature scheme whose signatures are only 64 bit-long. Interestingly, there are many designs, variants, and different parameters to achieve such a short signature.

In order to avoid problems arising with such short signatures, we have designed some specific new modes of operations.

Our ultra-short signature schemes were achieved thanks to the following ideas and tradeoffs:

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

With the future progress in cryptanalysis, it is likely that more efficient attacks will be found, and that we will therefore have to add some bits in our signatures to keep the same security levels. Nevertheless, as long as the best known attacks are still exponential in the "$-$" and "v" perturbations, only a few more bits would be needed (more information are given on this topic in the last paragraph of the conclusion). Moreover, we have confidence in the schemes we have presented here since their security relies on problems that have been extensively studied for more than 20 years.

Our implementation. We implemented our ultra-short signature schemes in `MAGMA`, our source code is available on our website. Moreover, the `MAGMA` programs we used to generates the values in the tables are on our website as well.

The advantage of these programs is that they can be executed with a free `MAGMA` license, allowing everybody to sign messages and verify signatures with our schemes or to verify the figures in our tables. In addition to this, according to `MAGMA` free license limitations, our programs always return a value within at most 2 minutes and using less than 366MB of memory.

To avoid desanonymization, we can not put the link in this very version of the paper; nevertheless, our code is available upon request for the reviewers, and, of course, we will put it in the putative accepted version of this paper.

To go further: evolution of our scheme with time. In the near future, let us say in 10 or 30 years, it is expected that computers will be more powerful than now, whereas humans' brain will remain the same.

As seen above, when the security parameter increases, the length of the signature must also increase. In the future, if the computational power of an attacker and of a legitimate user increase in the same proportion (for instance, from 2^{80} to 2^{90} for the attacker and from 2^{37} to 2^{47} for the legitimate user), will our signatures sizes inevitably grow a lot ?

To answer that question, we could look in the past in order to extend the results to the future. More precisely, for a former security level of 2^{70} and with a time allowed to sign or check a signature of 60ms (about 2^{27} operations), the signature length would be almost the same as it is for 80 bits of security nowadays.

To be very specific, here are the parameters that enable us to get those figures: $q = 2$, $r = 6$ (i.e. a degree $D = 129$), $n = 95$ (in order to have $\Delta \approx 0$), and $f = 20$ gives an HFE signature which can be computed or checked in 60ms. Its length is $95 - 20 = 75$ bits, i.e. about the same as in this paper with 1024 times more computations for the legitimate user and the attacks.

More investigations would be required here, nevertheless this example shows that it is likely that the length of our ultra-short signature will grow very slowly, even if the computational power of both the attacker and the legitimate user goes up.

References

1. Anonymous: UOV-Salt: New public key short signature in degree 3. Asiacrypt 2021 submission (2021)
2. Bardet, M.: Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie. Ph.D. thesis, Pierre and Marie Curie University, Paris, France (2004), https://tel.archives-ouvertes.fr/tel-00449609
3. Bardet, M., Bros, M., Cabarcas, D., Gaborit, P., Perlner, R., Smith-Tone, D., Tillich, J.P., Verbel, J.: Improvements of algebraic attacks for solving the rank decoding and minrank problems. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 507–536. Springer (2020)
4. Bardet, M., Faugère, J., Salvy, B., Spaenlehauer, P.: On the complexity of solving quadratic boolean systems. J. Complex. **29**(1), 53–75 (2013)
5. Bardet, M., Faugère, J.C., Saly, B.: On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In: Proc. International Conference on Polynomial System Solving Paris, France, 2004. pp. 71–75 (2004)
6. Bettale, L., Faugère, J., Perret, L.: Hybrid approach for solving multivariate systems over finite fields. Journal of Mathematical Cryptology **3**(3), 177–197 (2009)
7. Bouillaguet, C., Fouque, P.A., Macario-Rat, G.: Practical key-recovery for all possible parameters of SFLASH. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 667–685. Springer (2011)
8. Buhler, J.P., Lenstra, H.W., Pomerance, C.: Factoring integers with the number field sieve. In: The development of the number field sieve, pp. 50–94. Springer (1993)
9. Casanova, A., Faugère, J.C., Macario-Rat, G., Patarin, J., Perret, L., Ryckeghem, J.: GeMSS: A Great Multivariate Short Signature. Research report, UPMC - Paris 6 Sorbonne Universités ; INRIA Paris Research Centre, MAMBA Team, F-75012, Paris, France ; LIP6 - Laboratoire d'Informatique de Paris 6 (Dec 2017), https://hal.inria.fr/hal-01662158
10. Ding, J., Perlner, R., Petzoldt, A., Smith-Tone, D.: Improved cryptanalysis of HFE$v-$ via projection. In: Lange, T., Steinwandt, R. (eds.) Post-Quantum Cryptography. pp. 375–395. Springer International Publishing, Cham (2018)
11. Ding, J., Schmidt, D.: Rainbow, a new multivariable polynomial signature scheme. In: Proceedings of the Third International Conference on Applied Cryptography and Network Security. p. 164–175. ACNS'05, Springer-Verlag, Berlin, Heidelberg (2005)

12. Ding, J., Yang, B.Y.: Degree of regularity for HFEv and HFE$v-$. In: International Workshop on Post-Quantum Cryptography. pp. 52–66. Springer (2013)
13. Faugere, J.C.: A new efficient algorithm for computing Gröbner bases (F4). Journal of pure and applied algebra **139**(1-3), 61–88 (1999)
14. Faugere, J.C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In: Proceedings of the 2002 international symposium on Symbolic and algebraic computation. pp. 75–83 (2002)
15. Faugère, J., Joux, A.: Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using Gröbner bases. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2729, pp. 44–60. Springer (2003)
16. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 206–222. Springer (1999)
17. Matsumoto, T., Imai, H.: 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 (1988)
18. Patarin, J.: Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88. In: Annual International Cryptology Conference. pp. 248–261. Springer (1995)
19. Patarin, J.: Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 33–48. Springer (1996)
20. Patarin, J.: La cryptographie multivariable. Mémoire d'habilitation à diriger des recherches de l'Université Paris **7**, 354 (1999)
21. Patarin, J., Courtois, N., Goubin, L.: Quartz, 128-bit long digital signatures. In: Cryptographers' Track at the RSA Conference. pp. 282–297. Springer (2001)
22. Sturmfels, B.: What is... a Gröbner basis? Notices-American Mathematical Society **52**(10), 1199 (2005)
23. Tao, C., Petzoldt, A., Ding, J.: Improved key recovery of the HFE$v-$ signature scheme. IACR Cryptol. ePrint Arch **1424** (2020)
24. Vates, J., Smith-Tone, D.: Key recovery attack for all parameters of HFE. In: International Workshop on Post-Quantum Cryptography. pp. 272–288. Springer (2017)

Appendices

A Best known attacks complexities on nude HFE

B 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

Table 7. Complexity of attacks against nude HFE, $D = 16385$ ($r = 15$)

n	Comp	Columns
50	77.3	31/50
100	80.3	31/100
200	83.3	31/200
300	85.0	31/300
400	86.3	31/400
500	87.2	31/500
600	88.0	31/600

Table 8. Complexity of attacks against nude HFE, $D = 32769$ ($r = 16$)

n	Comp	Columns
50	81.3	33/50
100	84.3	33/100
200	87.3	33/200
300	89.0	33/300
400	90.3	33/400
500	91.2	33/500
600	92.0	33/600

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 $\mathbb{F}_{q^n}$.

In table 18, in addition to the time obtained by Magma, we present the time obtained with the improved software of the GeMMS team. Note that for example, when $D = 19$, the improved software is 23 times faster than Magma.

C Multiple independent public keys mode of operation.

In the main body of this paper, we have decided to use the “Feistel-Patarin” mode of operation in order to avoid the birthday attacks. We will present here another possible solution: the “Multiple independent keys” mode. In terms of ultra short signatures, this mode is slightly better, however in terms of public key length, the “Feistel-Patarin” mode is much better. In fact the Multiple independent keys mode is often not realistic due to huge public key sizes.

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. When we want to sign a message M, we will first compute a public function $f(M)$ that gives an integer between 1 and k, and this integer will be the number of the public key that we will use. 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

Table 9. Complexity of attacks against nude HFE, $D = 65537$ ($r = 17$)

n	Comp	Columns
50	85.3	35/50
100	88.3	35/100
200	91.3	35/200
300	93.0	35/300
400	94.3	35/400
500	95.2	35/500
600	96.0	35/600

Table 10. Complexity of attacks against nude HFE, $D = 131073$ ($r = 18$)

n	Comp	Columns
50	89.3	37/50
100	92.3	37/100
200	95.3	37/200
300	97.0	37/300
400	98.3	37/400
500	99.2	37/500
600	100.0	37/600

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 "Multiple independent public keys"-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 selects 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.

Table 11. Complexity of attacks against nude HFE, $D = 262145$ ($r = 19$)

n	Comp	Columns
50	93.3	39/50
100	96.3	39/100
200	99.3	39/200
240	100.1	39/240
300	101.0	39/300
400	102.3	39/400
500	103.2	39/500
600	104.0	39/600

Table 12. Complexity of attacks against nude HFE, $D = 524289$ ($r = 20$)

n	Comp	Columns
50	97.3	41/50
100	100.3	41/100
110	100.7	41/110
200	103.3	41/200
300	105.0	41/300
400	106.3	41/400
500	107.2	41/500
600	108.0	41/600

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 7. 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.

Table 13. Complexity of attacks against nude HFE, $D = 1048577$ ($r = 21$)

n	Comp	Columns
50	101.3	43/50
100	105.3	43/100
200	107.3	43/200
300	109.0	43/300
400	110.3	43/400
500	111.2	43/500
600	112.0	43/600

Table 14. Complexity of attacks against nude HFE, $D = 2097153$ ($r = 22$)

n	Comp	Columns
50	105.3	45/50
100	108.3	45/100
200	111.3	45/200
300	113.0	45/300
400	114.3	45/400
500	115.2	45/500
600	116.0	45/600

Table 15. Complexity of attacks against nude HFE, $D = 4194305$ ($r = 23$)

n	Comp	Columns
50	109.3	47/50
100	112.3	47/100
200	115.3	47/200
300	117.0	47/300
400	118.3	47/400
500	119.2	47/500
600	120.0	47/600

Table 16. Complexity of attacks against nude HFE, $D = 8388609$ ($r = 24$)

n	Comp	Columns
50	118.3	47/50
100	120.3	51/100
200	123.3	51/200
300	125.0	51/300
400	126.3	51/400
500	127.2	51/500
600	128.0	51/600

Table 17. Complexity of attacks against nude HFE, $D = 16777217$ ($r = 25$)

n	Comp	Columns
50	109.3	47/50
100	112.3	51/100
200	115.3	51/200
300	117.0	51/300
400	118.3	51/400
500	119.2	51/500
600	120.0	51/600

Table 18. Time to compute roots of a polynomial of degree D, $q = 2$, $n = 127$.

D	r	Time (Magma)	Time (GeMMS team)
129	8	0.095 s.	0.0041 s.
32769	16	98.3 s.	10 s.
65537	17	>2mn.	25 s.
131073	18	>2mn.	68 s.

Table 19. 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 20. 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 21. Time to compute roots of a polynomial of degree D, $q = 7$, $n = 40$.

q=7, n=40			
D	8	50	344
Time (ms)	9.35	105	1630
D	2402	16808	
Time (s)	10.2	110	

Table 22. Time to compute roots of a polynomial of degree D, $q = 11$, $n = 35$.

q=11, n=35				
D	12	122	1332	14642
Time	15 ms	360 ms	11.7 s	86.9 s

Table 23. Time to compute roots of a polynomial of degree D, $q = 13$, $n = 35$.

q=13, n=35			
D	14	170	2198
Time	23 ms	540 ms	11.4 s

Table 24. Time to compute roots of a polynomial of degree D, $q = 17$, $n = 33$.

q=17, n=33			
D	18	290	4914
Time	55 ms	1680 ms	29.2 s