GeMSS: A Gr$e$at Multivariate Short Signature

Principal submitter

This submission is from the following team, listed in alphabetical order:

- A. Casanova, CS
- J.-C. Faugère, INRIA and Sorbonne Universities/UPMC Univ Paris 06
- G. Macario-Rat, Orange
- J. Patarin, University of Versailles
- L. Perret, Sorbonne Universities/UPMC Univ Paris 06 and INRIA
- J. Ryckeghem, Sorbonne Universities/UPMC Univ Paris 06 and INRIA

E-mail address: `ludovic.perret@lip6.fr`

Telephone : +33-1-44-27-88-35

Postal address:
Ludovic Perret
Université Pierre et Marie Curie
LIP6 - Équipe projet INRIA/UPMC POLSYS
Boite courier 169
4 place Jussieu
F-75252 Paris cedex 5, France

Auxiliary submitters: There are no auxiliary submitters. The principal submitter is the team listed above.

Inventors/developers: The inventors/developers of this submission are the same as the principal submitter. Relevant prior work is credited below where appropriate.

Owner: Same as submitter.

Signature: $\times$. See also printed version of "Statement by Each Submitter".

Document generated with the help of `pqskeleton` version 20170923.

Contents

1 Introduction

sparkling GeMSS *spring up from the night sky*
a dazzling splendor to ever beautify
sequined glories that verily eye smack
sparkling GeMSS *spring up from night sky*
studding the vast backdrop of black

The purpose of this document is to present GeMSS : a Great Multivariate Signature Scheme. As suggested by its name, GeMSS is a multivariate-based [48, 62, 25, 8, 58, 55] signature scheme producing small signatures. It has a fast verification process, and a medium/large public-key. GeMSS is in direct lineage from `QUARTZ` [54] and borrows some design rationale of the `Gui` multivariate signature scheme [26]. The former schemes are built from the *Hidden Field Equations* crypotsystem (`HFE`) [52, published in 1996] by using the so-called minus and vinegar modifiers, i.e. `HFEv-` [45]. It is fair to say that `HFE`, and its variants, are the most studied schemes in multivariate cryptography. `QUARTZ` produces signatures of 128 bits for a security level of 80 bits and was submitted to the *Nessie Ecrypt* competition [50] for public-key signatures. In contrast to many multivariate schemes, no practical attack has been reported against `QUARTZ`. This is remarkable knowing the intense activity in the cryptanalysis of multivariate schemes, e.g. [51, 46, 32, 36, 43, 42, 27, 37, 25, 8, 12, 7, 55, 60]. The best known attack remains [36] that serves as a reference to set the parameters for GeMSS.

GeMSS is a faster variant of `QUARTZ` that incorporates the latest results in multivariate cryptography to reach higher security levels than `QUARTZ` whilst improving efficiency.

Acknowledgement. GeMSS has been prepared with the support of the french Programme d'Investissement d'Avenir under national project RISQ[1] P141580.

2 General algorithm specification (part of 2.B.1)

2.1 Parameter space

The main parameters involved in GeMSS are:

- D, a positive integer that is the degree of a secret polynomial. D is such that $D = 2^i$ for $i \geq 0$, or $D = 2^{i+j}$ for $i \neq j$, and $i, j \geq 0$.
- K, the output size in bits of the hash function,
- λ, the security level of GeMSS,
- m, number of equations in the public-key,
- nb_ite > 1, number of iterations in the verification and signature processes,
- n, the degree of a field extension,
- v, the number of vinegar variables,

[1] `https://risq.fr/?page_id=31&lang=en`

- Δ, the number of minus (the number of equations in the public-key is such that is $m = n - \Delta$).

In Section 3, we specify precisely these parameters to achieve a security level $\lambda \in \{128, 192, 256\}$.

2.2 Secret-key and public-key

The public-key in G*e*MSS is a set $p_1, \ldots, p_m \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]$ of m quadratic equations in $n + v$ variables. These equations are derived from a multivariate polynomial $F \in \mathbb{F}_{2^n}[X, v_1, \ldots, v_v]$ with a specific form – as described in (1) – such that generating a signature is essentially equivalent to find the roots of F.

Secret-key. It is composed by a couple of invertible matrices $(\mathbf{S}, \mathbf{T}) \in \mathrm{GL}_{n+v}(\mathbb{F}_2) \times \mathrm{GL}_n(\mathbb{F}_2)$ and a polynomial $F \in \mathbb{F}_{2^n}[X, v_1, \ldots, v_v]$ with the following structure:

$$\sum_{\substack{0 \leqslant i < j < n \\ 2^i + 2^j \leqslant D}} A_{i,j}\, X^{2^i + 2^j} + \sum_{\substack{0 \leqslant i < n \\ 2^i \leqslant D}} \beta_i(v_1, \ldots, v_v)\, X^{2^i} + \gamma(v_1, \ldots, v_v), \tag{1}$$

where $A_{i,j}, B_i, C \in \mathbb{F}_{2^n}, \forall i, j, 0 \leqslant i < j < n$, each $\beta_i : \mathbb{F}_2^v \to \mathbb{F}_{2^n}$ is linear and $\gamma(v_1, \ldots, v_v) : \mathbb{F}_2^v \to \mathbb{F}_{2^n}$ is quadratic. The variables $v_1, \ldots, v_v$ are called the *vinegar variables*. We shall say that a polynomial $F \in \mathbb{F}_{2^n}[X, v_1, \ldots, v_v]$ with the form of (1) has a `HFEv`-*shape*.

Remark 1. *The particularity of a polynomial $F(X, v_1, \ldots, v_v)$ with* `HFEv`*-shape is that for any specialization of the vinegar variables the polynomial F becomes a* `HFE` *polynomial [52], i.e. univariate polynomial of the following form:*

$$\sum_{\substack{0 \leqslant j < i < n \\ 2^i + 2^j \leqslant D}} A_{i,j}\, X^{2^i + 2^j} + \sum_{\substack{0 \leqslant i < n \\ 2^i \leqslant D}} B_i\, X^{2^i} + C \in \mathbb{F}_{2^n}[X], \tag{2}$$

with $A_{i,j}, B_i, C \in \mathbb{F}_{2^n}, \forall i, j, 0 \leqslant i, j < n$.

By abuse of notation, we will call degree of F the (max) degree of its corresponding `HFE` polynomials, i.e. D.

The special structure of (1) is chosen such that its *multivariate representation* over the base field $\mathbb{F}_2$ is composed by quadratic polynomials in $\mathbb{F}_2[x_1, \ldots, x_{n+v}]$. This is due to the special exponents chosen in X that have all a binary decomposition of Hamming weight at most 2.

Let $(\theta_1, \ldots, \theta_n) \in (\mathbb{F}_{2^n})^n$ be a basis of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$. We set $\varphi : E = \sum_{k=1}^n e_k \cdot \theta_k \in \mathbb{F}_{2^n} \longrightarrow \varphi(E) = (e_1, \ldots, e_n) \in \mathbb{F}_2^n$.

We can now define a set of multivariate polynomials $\mathbf{f} = (f_1, \ldots, f_n) \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^n$ derived from a `HFEv` polynomial $F \in \mathbb{F}_{2^n}[X, v_1, \ldots, v_v]$ by:

$$F\left(\sum_{k=1}^n \theta_k x_k, v_1, \ldots, v_v\right) = \sum_{k=1}^n \theta_k f_k\,. \tag{3}$$

To ease notations, we now identify the vinegar variables $(v_1, \ldots, v_v) = (x_{n+1}, \ldots, x_{n+v})$. Also, we shall say that the polynomials $f_1, \ldots, f_n \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]$ are the *components* of F over $\mathbb{F}_2$.

Public-key. It is given by a set of m quadratic *square-free* non-linear polynomials in $n + v$ variables over $\mathbb{F}_2$. That is, the public key is $\mathbf{p} = (p_1, \ldots, p_m) \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^m$. It is obtained from the secret-key by taking the first $m = n - \Delta$ polynomials of:

$$\Big(f_1\big((x_1, \ldots, x_m)\mathbf{S}\big), \ldots, f_n\big((x_1, \ldots, x_m)\mathbf{S}\big)\Big)\mathbf{T}, \tag{4}$$

and reducing it modulo the field equations, i.e. modulo $\langle x_1^2 - x_1, \ldots, x_{n+v}^2 - x_{n+v}\rangle$. We denote these polynomials by $\mathbf{p} = (p_1, \ldots, p_m) \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^m$.

We summarize the public-key/secret-key generation in Algorithm (1). It takes the security parameter λ as input. As discussed in Section 8, the security level of G*e*MSS will be a function of D, n, v and m. In Section 3 and in Section 9, we specify precisely these parameters. Section 3 presents some parameters in order to achieve a security level $\lambda \in \{128, 192, 256\}$. In section 9, we specify some others possible parameters.

Algorithm 1 PK/SK generation in G*e*MSS

1: **procedure** G*e*MSS.KeyGen(1^λ)
2: Randomly sample $(\mathbf{S}, \mathbf{T}) \in \mathrm{GL}_{n+v}(\mathbb{F}_2) \times \mathrm{GL}_n(\mathbb{F}_2)$ ▷ This step is further detailed in Section 2.5.1.
3: Randomly sample $F \in \mathbb{F}_2[X, v_1, \ldots, v_v]$ with `HFEv`-shape of degree D ▷ This step is further detailed in Section 2.5.2.
4: $\mathsf{sk} \leftarrow (F, \mathbf{S}, \mathbf{T}) \in \mathbb{F}_2[X, v_1, \ldots, v_v] \times \mathrm{GL}_{n+v}(\mathbb{F}_2) \times \mathrm{GL}_n(\mathbb{F}_2)$
5: Compute $\mathbf{f} = (f_1, \ldots, f_n) \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^n$ such that:

$$F\left(\sum_{k=1}^{n} \theta_k x_k, v_1, \ldots, v_v\right) = \sum_{k=1}^{n} \theta_k f_k$$

▷ See Section 2.5.4 for details on Step 5.
6: Compute $(p_1, \ldots, p_n) =$

$$\Big(f_1\big((x_1, \ldots, x_{n+v})\mathbf{S}\big), \ldots, f_n\big((x_1, \ldots, x_{n+v})\mathbf{S}\big)\Big)\mathbf{T} \bmod \langle x_1^2 - x_1, \ldots, x_{n+v}^2 - x_{n+v}\rangle \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^n$$

7: $\mathsf{pk} \leftarrow \mathbf{p} = (p_1, \ldots, p_m) \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^m$ ▷ Take the first $m = n - \Delta$ polynomials computed in Step 6
8: **return** $(\mathsf{sk}, \mathsf{pk})$
9: **end procedure**

2.3 Signing process

The main step of the signature process requires to solve:

$$p_1(x_1, \ldots, x_{n+v}) - d_1 = 0, \ldots, p_m(x_1, \ldots, x_{n+v}) - d_m = 0. \tag{5}$$

for $\mathbf{d} = (d_1, \ldots, d_m) \in \mathbb{F}_2^m$.

To do so, we randomly sample $\mathbf{r} = (r_1, \ldots, r_{n-m}) \in \mathbb{F}_2^{n-m}$ and append it to $\mathbf{d}$. This gives $\mathbf{d}' = (\mathbf{s}, \mathbf{r}) \in \mathbb{F}_2^n$. We then compute $D' = \varphi^{-1}(\mathbf{d}' \times \mathbf{T}^{-1}) \in \mathbb{F}_{2^n}$ and try to find a root $(Z, z_1, \ldots, z_v) \in$

$\mathbb{F}_{2^n} \times \mathbb{F}_2^v$ of the multivariate equation:

$$F(Z, z_1, \ldots, z_v) - D' = 0.$$

To solve this equation, we take advantage of the special `HFEv`-shape. That is, we randomly sample $\mathbf{v} \in \mathbb{F}_2^v$ and consider the univariate polynomial $F(X, \mathbf{v}) \in \mathbb{F}_{2^n}[X]$. This yields a `HFE` polynomial according to Remark 1. We then find the roots of the univariate equation:

$$F(X, \mathbf{v}) - D' = 0.$$

If there is a root $Z \in \mathbb{F}_{2^n}$, we return $(\varphi(Z), \mathbf{v}) \times \mathbf{S}^{-1} \in \mathbb{F}_2^{n+v}$.

A core part of the signature generation is to compute the roots of $F_{D'}(X) = F(X, \mathbf{v}) - D'$. To do so, we use the Berlekamp algorithm as described in [61, Algorithm 14.15].

Algorithm 2 Algorithm for finding the roots of an univariate polynomial

```
function FindRoots(F_{D'} ∈ F_{2^n}[X])
    X_n ← X^{2^n} − X mod F_{D'}        ▷ This step is further detailed in Section 5.6.3
    G ← gcd(F_{D'}, X_n)
    if degree(G) > 0 then
        Roots ← List of all roots of G, computed by the equal-degree factorization algorithm
described in [61, Section 14.3]
        return (degree(G), Roots)
    end if
    return (degree(G), ∅)
end function
```

The complexity of Algorithm 2 is given by the following general result:

Theorem 1 (Corollary 14.16 from [61])**.** *Let $\mathbb{F}_q$ be a finite field, and $\mathrm{M}_q(D)$ be the number of operations in $\mathbb{F}_q$ to multiply two polynomials of degree $\leq D$. Given $f \in \mathbb{F}_q[x]$ of degree D, we can find all the roots of f over $\mathbb{F}_q$ using an expected number of*

$$O\Big(\mathrm{M}_q(D) \log(D) \log(Dq)\Big)$$

or $\tilde{O}\big(D \log(q)\big)$ operations in $\mathbb{F}_q$.

For $q = 2^n$, we get that finding all the roots of a polynomial of degree D can be done in (expected) quasi-linear time, i.e.:

$$\tilde{O}(nD). \tag{6}$$

We can now present the inversion function (Algorithm 3):

Remark 2. *We sample a root at Step 12 always in the same way. First, we sort the elements of* Roots *in ascending order. We then compute* `SHA3`(D')*, and take the first* 64 *bits H_{64} of this hash. We view H_{64} as an integer, and finally return the* $(H_{64} \mod \#\mathrm{Roots})$*-th element in* Roots*.*

Let $\mathbf{d} \in \mathbb{F}_2^{n+v}$ and $\mathbf{s} \leftarrow \mathrm{Inv}_{\mathbf{p}}\big(\mathbf{d}, \mathrm{sk} = (F, \mathbf{S}, \mathbf{T})\big) \in \mathbb{F}_2^{n+v}$. By construction, we have:

$$\mathbf{p}(\mathbf{s}) = \mathbf{d}, \text{where } \mathbf{p} \text{ in the public-key associated to sk.}$$

Algorithm 3 Inversion in GeMSS

function GeMSS.Inv$_{\mathbf{p}}$($\mathbf{d} \in \mathbb{F}_2^m$, sk $= (F, \mathbf{S}, \mathbf{T}) \in \mathbb{F}_2[X, v_1, \ldots, v_v] \times \mathrm{GL}_{n+v}(\mathbb{F}_2) \times \mathrm{GL}_n(\mathbb{F}_2)$)
  repeat
    $\mathbf{r} \in_R \mathbb{F}_2^{n-m}$ ▷ The notation $\in_R$ stands for randomly sampling.
    $\mathbf{d}' \leftarrow (\mathbf{d}, \mathbf{r}) \in \mathbb{F}_2^n$
    $D' \leftarrow \varphi^{-1}(\mathbf{d}' \times \mathbf{T}^{-1}) \in \mathbb{F}_{2^n}$
    $\mathbf{v} \in_R \mathbb{F}_2^v$
    $F_{D'}(X) \leftarrow F(X, \mathbf{v}) - D'$
    $(\cdot, \mathrm{Roots}) \leftarrow \mathrm{FindRoots}(F_{D'})$
  until Roots $= \emptyset$
  $Z \in_R$ Roots
  return $(\varphi(Z), \mathbf{v}) \times \mathbf{S}^{-1} \in \mathbb{F}_2^{n+v}$
end function

Thus, $\mathbf{s} \in \mathbb{F}_2^{n+v}$ could be directly used as a signature for the corresponding digest $\mathbf{d} \in \mathbb{F}_2^m$. In the case of G$e$MSS, m is small enough to make the cost of simple birthday-paradox attack against the hash function more efficient that all possible attacks (as those listed in Section 8). This problem was already identified in `QUARTZ` and `Gui` [54, 20, 22, 57] who proposed to handle this issue by using the so-called *Feistel-Patarin* scheme.

The basic principle of the Feistel-Patarin scheme is to roughly iterate Algorithm 3 several times. The number of iterations is a parameter nb_ite that will be discussed in Section 6.1. We will see that we can choose nb_ite $= 4$ as in `QUARTZ` [54, 20, 22].

Algorithm 4 Signing process in GeMSS

procedure GeMSS.Sign($\mathbf{M} \in \{0,1\}^*$, sk $\in \mathbb{F}_2[X, v_1, \ldots, v_v] \times \mathrm{GL}_{n+v}(\mathbb{F}_2) \times \mathrm{GL}_n(\mathbb{F}_2)$, G$e$MSS.Inv$_{\mathbf{p}}$)
  $\mathbf{H} \leftarrow \mathrm{SHA3}(\mathbf{M})$
  $\mathbf{S}_0 \leftarrow \mathbf{0} \in \mathbb{F}_2^m$
  for i from 1 to nb_ite **do**
    $\mathbf{D}_i \leftarrow$ first m bits of $\mathbf{H}$
    $(\mathbf{S}_i, \mathbf{X}_i) \leftarrow$ GeMSS.Inv$_{\mathbf{p}}(\mathbf{D}_i \oplus \mathbf{S}_{i-1})$ ▷ $\mathbf{S}_i \in \mathbb{F}_2^m$ and $\mathbf{X}_i \in \mathbb{F}_2^{n+v-m}$, $\oplus$ is the component-wise XOR
    $\mathbf{H} \leftarrow \mathrm{SHA3}(\mathbf{H})$
  end for
  return $(\mathbf{S}_{\text{nb\_ite}}, \mathbf{X}_{\text{nb\_ite}}, \ldots, \mathbf{X}_1)$ ▷ This is of size $m + \text{nb\_ite}(n + v - m) = m + \text{nb\_ite}(\Delta + v)$ bits
end procedure

2.4 Verification process

The verification process corresponding to Algorithm 4 is given in Algorithm 5.

Algorithm 5 Verification process in GeMSS

procedure GeMSS.VERIF($\mathbf{M} \in \{0,1\}^*, \text{nb\_ite} > 0, (\mathbf{S}_{\text{nb\_ite}}, \mathbf{X}_{\text{nb\_ite}}, \ldots, \mathbf{X}_1) \in \mathbb{F}_2^{m+\text{nb\_ite}(n+v-m)}$, $\mathsf{pk} = \mathbf{p} \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^m$)
    $\mathbf{H} \leftarrow \text{SHA3}(\mathbf{M})$
    $(\mathbf{S}_{\text{nb\_ite}}, \mathbf{X}_{\text{nb\_ite}}, \ldots, \mathbf{X}_1) \leftarrow \mathbf{S}$
    for i from 1 to nb_ite **do**
        $\mathbf{D}_i \leftarrow$ first m bits of $\mathbf{H}$
        $\mathbf{H} \leftarrow \text{SHA3}(\mathbf{H})$
    end for
    for i from nb_ite $- 1$ to 0 **do**
        $\mathbf{S}_i \leftarrow \mathbf{p}(\mathbf{S}_{i+1}, \mathbf{X}_{i+1}) \oplus \mathbf{D}_{i+1}$
    end for
    return VALID if $\mathbf{S}_0 = \mathbf{0}$ and INVALID otherwise.
end procedure

2.5 Implementation

We detail here some of the choices done for implementing GeMSS.

2.5.1 Generating invertible matrices

Algorithm 1 requires, at Step 2, to generate a pair of invertible matrices $(\mathbf{S}, \mathbf{T}) \in \text{GL}_{n+v}(\mathbb{F}_2) \times \text{GL}_n(\mathbb{F}_2)$. This problem was already discussed for `QUARTZ` [54] who presented two (natural) methods to generate invertible matrices. The first one ("*Trial and error*") sample random matrices until one is invertible. The second one, that has be chosen in `QUARTZ`, uses the so-called LU decomposition. This method has the advantage to directly return an invertible matrix. It is as follows.

- Generate a square random lower triangular L and upper triangular U matrices over $\mathbb{F}_2$, both with ones on the diagonal (to have a non-zero determinant).
- Return $L \times U$.

It is known that this method is slightly biased. A small part of the invertible matrices can not be generated with this method. For a square matrix of size n, the number of invertible triangular matrices is $2^{\sum_{i=0}^{n-1} i} = 2^{\frac{n^2-n}{2}}$. So, the number of matrices that can be generated with the LU method is $\frac{2^{n^2}}{2^n}$. This don't reduce the search space on the secret matrices sufficiently to impact the security of GeMSS.

In the code, we have implemented both generation methods. The implementation gives the possibility to switch the method with the macro `GEN_INVERTIBLE_MATRIX_LU`, which is in the file `encrypt_keypairHFE.c`. It is initialized to 1 by default.

The matrices $(\mathbf{S}, \mathbf{T}) \in \text{GL}_{n+v}(\mathbb{F}_2) \times \text{GL}_n(\mathbb{F}_2)$ are in fact only used during the generation of the public-key. After, we are only using the inverse of these matrices. So, $\mathbf{S}^{-1}$ and $\mathbf{T}^{-1}$ are computed during the generation and are stored in the secret key.

2.5.2 Generating `HFEv` polynomials

Algorithm 1 requires, at Step 3, to generate a polynomial $F \in \mathbb{F}_{2^n}[X, v_1, \ldots, v_v]$ with `HFEv`-shape of degree D. The polynomial F can be seen as a polynomial in X whose coefficients are in $\mathbb{F}_{2^n}[v_1, \ldots, v_v]$. We store and randomly generate the non-zero exponents of F.

The polynomial F is chosen monic and so the leading coefficient is not stored. This choice makes easier the roots finding part (Algorithm 2).

2.5.3 Data structure for $\mathbb{F}_2[x_1, \ldots, x_{n+v}]^m$

The first idea is to see m equations of $\mathbb{F}_2[x_1, \ldots, x_{n+v}]$ as one element in $\mathbb{F}_{2^m}[x_1, \ldots, x_{n+v}]$. The second idea is to use quadratic forms. Let $\mathbf{x} = (x_1, \ldots, x_{n+v}), C \in \mathbb{F}_{2^m}$ and $\mathbf{Q}, \mathbf{Q'} \in M_{n+v}(\mathbb{F}_{2^m})$, then a quadratic non-linear *square-free* polynomial in $\mathbb{F}_{2^m}[x_1, \ldots, x_{n+v}]$ can be written as

$$C + \mathbf{x}\mathbf{Q'}\mathbf{x}^t.$$

The coefficient $\mathbf{Q'}_{i,j}$ corresponds to the term $x_i x_j$ in the polynomial. Since $x_i^2 = x_i$, the linear term can be stored on the diagonal of $\mathbf{Q'}$.

To minimize the size, $\mathbf{Q'}$ can be transformed into a upper triangular matrix $\mathbf{Q}$. By construction, $\mathbf{Q'}_{i,j}$ and $\mathbf{Q'}_{j,i}$ are the coefficients of the same term $x_i x_j$ $(i \neq j)$. The matrix $\mathbf{Q}$ is such that:

$$\mathbf{Q}_{i,j} = \begin{cases} \mathbf{Q'}_{i,j} & \text{if } i = j \\ \mathbf{Q'}_{i,j} + \mathbf{Q'}_{j,i} & \text{if } i < j \\ 0 & \text{else.} \end{cases}$$

2.5.4 Generating the components of a `HFEv` polynomial

We detail here how to obtain the multivariate polynomials $\mathbf{f} = (f_1, \ldots, f_n) \in (\mathbb{F}_2[x_1, \ldots, x_{n+v}])^n$ from a `HFEv` polynomial $F \in \mathbb{F}_{2^n}[X, v_1, \ldots, v_v]$ such that $\sum_{k=1}^n \theta_k f_k$. The principle is to symbolically compute $F\left(\sum_{k=1}^n \theta_k x_k, v_1, \ldots, v_v\right) \in \mathbb{F}_{2^n}[x_1, \ldots, x_{n+v}]$. In the implementation, the basis $(\theta_1, \ldots, \theta_n) \in (\mathbb{F}_{2^n})^n$ is the canonical basis of $\mathbb{F}_{2^n}$.

The polynomial F can be seen as a polynomial in X whose coefficients are in $\mathbb{F}_{2^n}[v_1, \ldots, v_v]$. We first consider terms of the form X^{2^i}. Clearly, $(\sum_{i=k}^n \theta_k x_k)^{2^i} = (\sum_{k=1}^n \theta_k^{2^i} x_k)$. We then get linear terms involved in the $f_1, \ldots, f_n$. It is the same idea for a term of the form $X^{2^i + 2^j}$. We get the quadratic terms in the f_k's by $X^{2^i} X^{2^j} = (\sum_{k=1}^n \theta_k^{2^i} x_k) \times (\sum_{k=1}^n \theta_k^{2^j} x_k)$.

2.5.5 Generation of the public-key $\mathsf{pk} = \mathbf{p} \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^m$

According to Section 2.5.3, $\mathbf{f}$ is stored as $C + \mathbf{x}\mathbf{Q}\mathbf{x}^t \in \mathbb{F}_{2^m}[x_1, \ldots, x_{n+v}]$. We first compute $\left(f_1\left((x_1, \ldots, x_n)\,\mathbf{S}\right), \ldots, f_n\left((x_1, \ldots, x_n)\,\mathbf{S}\right)\right)$ (Step 6, Algorithm 1) with our representation. To do so, we just replace $\mathbf{x}$ by $\mathbf{x}\ \mathbf{S}$. The linear change of variables by $\mathbf{S}$ can be represented as:

$$C + \mathbf{x}\mathbf{Q'}\mathbf{x}^t \in \mathbb{F}_{2^n}[x_1, \ldots, x_{n+v}]$$

with $\mathbf{Q}' = \mathbf{S}\mathbf{Q}\mathbf{S}^{\mathbf{t}}$.

We then symmetrize the matrix $\mathbf{Q}'$ as in Section 2.5.3 to get an upper triangular matrix $\mathbf{Q}''$.

To obtain the public key, we now need to perform linear combinations with the matrix $\mathbf{T}$. With our representation, this is equivalent to apply $\mathbf{T}$ to each coefficient to obtain the public-key in the form:

$$C_{\mathsf{pk}} + (\mathbf{x}\mathbf{Q}_{\mathsf{pk}}\mathbf{x}^t),$$

with $C_{\mathsf{pk}} \in \mathbb{F}_{2^m}$ and $\mathbf{Q}_{\mathsf{pk}} \in M_{n+v}(\mathbb{F}_{2^m})$.

in this form, the evaluation of the public-key reduce to a matrix-vector and vector-vector products in $\mathbb{F}_{2^m}$.

3 List of parameter sets (part of 2.B.1)

Following the analysis of Section 8, we propose below a set of 3 parameters for $128, 192$ and 256 bits of classical security. In Section 8.6, we give a general method allowing to derive others parameters. Also, Section 9 gives others parameters with different tradeoffs.

3.1 Parameter set `sign/GeMSS128`

We choose nb_ite $= 4$, $\Delta = 12, v = 12$ and $m = 162$. This gives $n = 174, n + v = 186$, $D = 513$ and $K = 128$. The extension field is defined as $\mathbb{F}_{2^n} = \frac{\mathbb{F}_2[X]}{X^n+X^{13}+1}$.

This gives a public-key of 352.18 KBytes, a signature of 258 bits, a time to sign of 260 ms and 41 μs to verify (Section 5.3.3).

3.2 Parameter set `sign/GeMSS192`

We choose nb_ite $= 4, \Delta = 22, v = 20$ and $m = 243$. This gives $n = 265, n + v = 285$, $D = 513$ and $K = 192$. The extension field is defined as $\mathbb{F}_{2^n} = \frac{\mathbb{F}_2[X]}{X^n+X^{42}+1}$.

This gives a public-key of 1237.96 KBytes, a signature of 411 bits and a time to sign of 694 ms and 117 μs to verify (Section 5.3.3).

3.3 Parameter set `sign/GeMSS256`

We choose nb_ite $= 4, \Delta = 30, v = 33$ and $m = 324$. This gives $n = 354, n + v = 387$, $D = 513$ and $K = 256$. The extension field is defined as $\mathbb{F}_{2^n} = \frac{\mathbb{F}_2[X]}{X^n+X^{99}+1}$.

This gives a public-key of 3040.69 KBytes, a signature of 576 bits, a time to sign of 1.09 s and 336 μs to verify (Section 5.3.3).

4 Design rationale (part of 2.B.1)

A multivariate scheme. The first design rational of G*e*MSS is to construct a signature scheme producing short signatures. It is well known that multivariate cryptography [62, 8, 25] provides the schemes with the smallest signatures among all post-quantum schemes. Multivariate-based signature schemes are even competitive with ECC-based, pre-quantum, signature schemes (see, for example [9, 49]). This explains the choice of a multivariate cryptosystem for G*e*MSS.

A HFE-based scheme. HFE [52] is probably the most popular multivariate cryptosystem. Its security has been extensively studied since more than 20 years. The complexity of the best known attacks against HFE are all exponential in $O\big(\log_2(D)\big)$, where D is the degree of the secret univariate polynomial. When D is too small, then HFE can be broken, e.g. [46, 36, 7]. In contrast, solving HFE is NP-Hard when $D = O(2^n)$ [46]. However, the complexity of the signature generation – that requires finding the roots of a univariate polynomial – is quasi-linear in D (Theorem 1). All in all, there is essentially one parameter, the degree D of the univariate secret polynomial, which governs the security and efficiency of HFE. The design challenge in HFE is to find a proper trade-off between efficiency and security.

Variants of HFE. A fundamental element in the design of secure signature schemes based on HFE is the introduction of perturbations. These creates many *variants* of the scheme. Classical perturbations include the *minus modifier* (HFE-,[52]) and the *vinegar modifier* (HFEv, [45, 54]).

Typically, QUARTZ is a HFEv- signature scheme where $D = 129, q = 2, n = 103, 4$ vinegar variables and 3 equations removed. The resistance, up to know, of QUARTZ against all known attacks illustrates that minus and vinegar variants permit to indeed strengthen the security of a HFE-based signature. A *nude* HFE, i.e. without any perturbation, with $D = 129$ and $n = 103$ would be insecure whilst no practical attack against QUARTZ has been reported in the literature. The best known attack is [36] that serves as a reference to set the parameters for G*e*MSS.

QUARTZ has the reputation to be solid but with a rather slow signature generation process. The authors of [54] reported a signature generation process taking about a minute. Today, the same parameters will take few hundred milliseconds. This is partly due to the technological progresses on the speed of processors. In fact, it is mostly due to a deeper understanding on algorithms finding the roots of univariate polynomials (see, for example [61]).

A descendant of QUARTZ. A method to improve the efficiency of QUARTZ is to take very small D but consider generalized HFE for field bigger than 2. This is the choice proposed by [57] for Gui. In G*e*MSS, we decided to work on a variant of HFE over $\mathbb{F}_2$ but to increase the degree with respect to QUARTZ. As we explained before, this is possible due to progresses on finding roots and our efficient roots-finding software implementation of roots finding.

5 Detailed performance analysis (2.B.2)

5.1 Description of platform

Computer	OS	Architecture	Processor	Frequency	Version of `g++`
Laptop	Ubuntu 16.04.3 LTS	x86_64	i7-6600U	2.60 GHz	6.3

Table 1: Materials.

Computer	RAM	L1d	L1i	L2	L3
Laptop	31.3 Gio	32 Ko	32 Ko	256 Ko	4096 Ko

Table 2: Memory.

The measurements used one core of the CPU, and the code was compiled with `g++ -O4`.
For the optimized and additional implementations, the code was compiled with `g++ -O4 -mavx2 -mpclmul`. The optimized implementation requires `-mavx2 -mpclmul` only to improve the performance of third-party open source libraries.

5.2 Third-party open source library

We have use the `Keccak` code package[2] and `NTL` library[3]. The optimized implementation uses `gf2x` library[4] which implements fast multiplications of binary polynomials. The additional implementation replaces `gf2x` library by a new implementation of multiplications of binary polynomials. In particular, we use `_mm_clmulepi64_si128` intrinsic to improve the multiplication of binary polynomials.

5.3 Time

The following measurements are for `sign`. For signature, it signs/verifies a document of 32 bytes. For the measures, it runs a number of tests such that the global used time is greater than 10 seconds, and the global time is divided by the number of tests. For the signature, the number of tests is 25.

5.3.1 Reference implementation

`GeMSS128`:
G*e*MSS.KeyGen takes 538 ms.

[2] https://keccak.team/
[3] http://www.shoup.net/ntl/
[4] http://gf2x.gforge.inria.fr/

The time to sign is 1.22 s (in average).
The verification takes 7.56 ms.

GeMSS192:
G*e*MSS.KeyGen takes 2.47s.
The time to sign 3.04s (in average).
Verification takes 25.19 ms.

GeMSS256:
G*e*MSS.KeyGen takes 6.99s.
The signature takes 4.9s (in average).
The verification takes 61.7 ms.

5.3.2 Optimized implementation

GeMSS128:
G*e*MSS.KeyGen takes 44 ms.
The time to sign is 323 ms (in average).
The verification takes 41 μs.

GeMSS192:
G*e*MSS.KeyGen takes 169 ms.
The time to sign is 793 ms (in average).
The verification takes 117 μs.

GeMSS256:
G*e*MSS.KeyGen takes 433 ms.
The time to sign 1.13 s (in average).
The verification takes 346 μs.

5.3.3 Additional (best) implementation

GeMSS128:
G*e*MSS.KeyGen takes 42 ms.
The signature takes 260 ms (in average).
The verification takes 41 μs.

GeMSS192:
G*e*MSS.KeyGen takes 166 ms.
The signature takes 694 ms (in average).
The verification takes 117 μs.

GeMSS256:
G*e*MSS.KeyGen takes 424 ms.
The signature takes 1.09 s (in average).
The verification takes 336 μs.

5.4 Space

Here are the size of public key, secret key and signature in the implementation. The implementation does not optimize the size, so it explains the difference with theorical sizes. In particular, the way to store the signature is very inefficient.

`GeMSS128`:
Public key is 417408 bytes. Secret key is 14208 bytes. Signatures are 48 bytes.

`GeMSS192`:
Public key is 1304192 bytes. Secret key is 39440 bytes. Signatures are 88 bytes.

`GeMSS256`:
Public key is 3603792 bytes. Secret key is 82056 bytes. Signatures are 104 bytes.

5.5 How parameters affect performance

Signature generation is mainly affected by n and the degree D of the secret univariate polynomial. According to Theorem 1, we can find the roots of $F \in \mathbb{F}_{2^n}[X]$ in $\tilde{O}\big(n\,D\big)$ binary operations. So, n and D are the main parameters which influence the efficiency. In Sec. 8, we will see how to choose these parameters in function of the security parameter.

5.6 Optimizations

The optimized implementation modifies the order of computations to have the best possible contiguity, and in this way avoids a maximum of miss in the cache. The implementation avoids to store useless null coefficients (for example, for a triangular matrix), and every data are stored in unidimensional tabular of words.

5.6.1 Improvement of the arithmetic in $\mathbb{F}_{2^n}$

The multiplication in $\mathbb{F}_{2^n}$ is the the most expensive part of GeMSS: the generation of the public-key/secret key requires $O\big(n^2\log(D)^2 + nv\log(D)\big)$ multiplications, and the signature requires $\tilde{O}\big(n\,D\big)$ multiplications.
To improve multiplication, the optimized implementation uses the `gf2x` library. The additional implementation uses a new implementation that is a classical multiplication, using `_mm_clmulepi64_si128` for the basis case. This implementation is faster than `gf2x` for small sizes of n.

The squaring in $\mathbb{F}_{2^n}$ is important in the signature generation. Indeed, the computation of $(X^{2^n} - X)$ mod F (Algorithm 2) requires $O(nD)$ squaring. Squaring consists just to interleave a zero bit between each bit of the input. To do this, the optimized implementation uses a precomputed table of 256 elements: it stores the squaring of all binary polynomials of size 8 bits. The additional implementation uses the intrinsic `_mm_clmulepi64_si128` to compute directly the squaring of an element of size 64 bits.

5.6.2 Evaluation of the public-key

The public-key in represented in the form:

$$C_{\mathsf{pk}} + \mathbf{x}\mathbf{Q}_{\mathsf{pk}}\mathbf{x}^t,$$

with $C_{\mathsf{pk}} \in \mathbb{F}_{2^m}$ and $\mathbf{Q}_{\mathsf{pk}} \in M_{n+v}(\mathbb{F}_{2^m})$.

The optimization is to set to zero the ith row of $\mathbf{Q}_{\mathsf{pk}}\mathbf{v}^t$ (a column vector) if the ith component of $\mathbf{v}$ is null. We avoid a dot product for each null coefficient.

5.6.3 Computation of Fröbenius map

To compute the roots of $F_{D'} = F(X, \mathbf{v}) - D'$ (Algorithm 2) during the signature , the reference implementation uses the `FrobeniusMap` function from NTL. To accelerate this function, the optimized implementation uses a C implementation of $(X^{2^n} - X) \mod F_{D'}$, as this:

Algorithm 6 Algorithm for Frobenius map

> **function** FROBENIUS_MAP($F_{D'}, n$)
>   Choose a such that $2^a <$ degree$(F_{D'})$ but $2^{a+1} \geq$ degree$(F_{D'})$.
> $X_a \leftarrow X^{2^a}$
>   **for** i from $a+1$ to$n$ **do**
> $X_i \leftarrow (X_{i-1})^2$ ▷ Linearity of Fröbenius endomorphism
> $X_i \leftarrow X_i \mod F_{D'}$ ▷ We use the fact that $F_{D'}$ is monic and sparse
>   **end for**
>   **return** $X_n + X$
> **end function**

To compute squaring is equivalent to compute the square of each coefficient, and put a null coefficient between each coefficient.
Since $F_{D'}$ is monic, there is useless to multiply $F_{D'}$ by the inverse of its leading coefficient to compute modular reduction. The fact that $F_{D'}$ is sparse avoids to load and read useless null coefficients, since just the useful coefficients are stored.

6 Expected strength (2.B.4) in general

We review in this part known results on the provable security of GeMSS. This includes the required number of iterations in the Feistel-Patarin scheme (Section 6.1) as well as the security (Section 6.2) in the sense of the existential unforgeability against adaptive chosen-message attack (EUF-CMA).

6.1 Number of iterations nb_ite in Sign and Verif

We explain here how the number of iterations nb_ite > 0 has to be chosen in Algorithms 4 and 5. This follows from the analysis performed already in `QUARTZ` [54, 20].

Theorem 2 (adapted from [20]). *The number of iterations* nb_ite *hast to be chosen such that*

$$2^{m\,\frac{\text{nb\_ite}}{\text{nb\_ite}+1}} \geq 2^{\lambda}.$$

For $\mathsf{G}e\mathsf{MSS}$, we fix nb_ite $= 4$ for all security parameters. This is similar to the choice for `QUARTZ` [54, 20].

6.2 EUF-CMA security

EUF-CMA security of `HFEv-`, over which $\mathsf{G}e\mathsf{MSS}$ is designed, has been mainly investigated in [59]. The authors demonstrated that a minor, but costly, modification of $\mathsf{G}e\mathsf{MSS}.\mathrm{Inv}_{\mathbf{p}}$ (Algorithm 3) permits to achieve EUF-CMA security for $\mathsf{G}e\mathsf{MSS}$. In fact, the result of [59] applies more precisely to a version of $\mathsf{G}e\mathsf{MSS}.\mathrm{Inv}_{\mathbf{p}}$ where nb_ite is equal to one. In this case, the EUF-CMA security of (modified) $\mathsf{G}e\mathsf{MSS}$ follows easily from [59].

We first formalize the security of $\mathsf{G}e\mathsf{MSS}$ against chosen message attacks.

Definition 1 ([59]). *The* $\mathsf{G}e\mathsf{MSS}$ *signature scheme* $(\mathsf{G}e\mathsf{MSS}.\textsc{KeyGen}, \mathsf{G}e\mathsf{MSS}.\textsc{Sign}, \mathsf{G}e\mathsf{MSS}.\textsc{Verif})$ *is* $\big(\epsilon(\lambda), q_s(\lambda), q_h(\lambda), t(\lambda)\big)$*-secure if there is no forger* A *who takes as input a public-key* $(\cdot, \mathsf{pk}_{\mathsf{G}e\mathsf{MSS}}) \leftarrow \mathsf{G}e\mathsf{MSS}.\textsc{KeyGen}()$ *and with at most* $q_h(\lambda)$ *queries to the random oracle,* $q_s(\lambda)$ *queries to the signature oracle, then outputs a valid signature after* $t(\lambda)$ *steps with a probability at least* $\epsilon(\lambda)$*.*

We want to provably reduce EUF-CMA security of $\mathsf{G}e\mathsf{MSS}$ to the the hardness of inverting the public-key of $\mathsf{G}e\mathsf{MSS}$. Formally:

Definition 2 ([59]). *We shall say that the* $\mathsf{G}e\mathsf{MSS}$ *function generator* $\mathsf{G}e\mathsf{MSS}.\textsc{KeyGen}$ *is* $\big(\epsilon(\lambda), t(\lambda)\big)$ *secure, if there is no inverting algorithm that takes* $\mathsf{pk}_{\mathsf{G}e\mathsf{MSS}} = \mathbf{p}_{\mathsf{G}e\mathsf{MSS}}$ *generated via* $(\cdot, \mathsf{pk}_{\mathsf{G}e\mathsf{MSS}}) \leftarrow \mathsf{G}e\mathsf{MSS}.\textsc{KeyGen}(1^{\lambda})$*, a challenge* $\mathbf{d} \in_R \mathbb{F}_2^m$*, and finds a preimage* $\mathbf{s} \in_R \mathbb{F}_2^{n+v}$ *such that*

$$\mathbf{p}_{\mathsf{G}e\mathsf{MSS}}(\mathbf{s}) = \mathbf{d}.$$

after $t(\lambda)$ *steps with success probability at least* $\epsilon(\lambda)$*.*

Following [59], we explain now how to modify $\mathsf{G}e\mathsf{MSS}$ for proving EUF-CMA security. Recall that D is degree of the secret polynomial with `HFEv`-shape in $\mathsf{G}e\mathsf{MSS}$. The main modification proposed by [59] is roughly to repeat D times the inversion step described in Algorithm 3.

Let ℓ be the length of a random salt. The modified inversion process is given in Algorithm 7:

Given Algorithm 7, we can define $\mathsf{G}e\mathsf{MSS}.\textsc{Sign}^*$ as the signature algorithm 4 instantiated with $\mathsf{G}e\mathsf{MSS}.\mathrm{Inv}^*_{\mathbf{p}}$ and with nb_ite $= 1$. Similarly, $\mathsf{G}e\mathsf{MSS}.\textsc{Verif}^*$ is the verification algorithm 5 where nb_ite $= 1$.

Algorithm 7 Modified inversion for GeMSS

```
1: procedure GeMSS.Inv*_p(d ∈ F_2^m, ℓ ∈ N, sk = (F, S, T) ∈ F_2[X, v_1, ..., v_v] × GL_{n+v}(F_2) × GL_n(F_2))
2:     v ∈_R F_2^v
3:     repeat
4:         salt ∈_R {0,1}^ℓ
5:         r ← first n − m bits of SHA3(d||salt)
6:         d' ← (d, r) ∈ F_2^n
7:         D' ← φ^{-1}(d' × T^{-1}) ∈ F_{2^n}
8:         F_{D'}(X) ← F(X, v) − D'
9:         (·, Roots) ← FindRoots(F_{D'})
10:        u ∈_R {1, ..., D}
11:    until 1 ≤ u ≤ #Roots
12:    Z ∈_R Roots
13:    return (φ(Z), v) × S^{-1} ∈ F_2^{n+v}
14: end procedure
```

Theorem 3 ([59]). *Let* GeMSS* *be the signature scheme defined by* (GeMSS.KeyGen, GeMSS.Sign*, GeMSS.Verif*). *Thus, if the* GeMSS *function generator* GeMSS.KeyGen *is* (ϵ', t') *secure, then* GeMSS* *is* (ϵ, t, q_H, q_S) *secure, with:*

$$\begin{aligned} \epsilon &= \frac{\epsilon'(q_H + q_s + 1)}{1 - (q_H + q_s)q_s 2^{\ell}}, \\ t &= \frac{t' - (q_H + q_s + 1)}{t_{\mathsf{GeMSS}} + O(1)} \end{aligned}$$

where t_{GeMSS} *is the time required to evaluate the public-key of* GeMSS.

There are two differences between GeMSS and GeMSS*. First, GeMSS.Inv$^*_{\mathbf{p}}$ is more costly than GeMSS.Inv$^*_{\mathbf{p}}$. The expected number of calls to the root-finding step (Step 9) in GeMSS.Inv$^*_{\mathbf{p}}$ is $\frac{1}{1-1/e}D \approx 1.58 \times D$. In G$e$MSS.Inv$_{\mathbf{p}}$, the average number of calls to the root-finding step (Step 8) is $\frac{1}{1-1/e} \approx 1.58$.

In GeMSS, we are typically considering $D \geq 512$. For efficiency reasons, we did not incorporated this modification in our implementation.

Remark 3. *The threshold D in Step 10 corresponds to a bound on the number of roots of the univariate polynomial F at Step 9. However, F has a* `HFE`*-shape (Remark 1) and has much less roots than a random univariate polynomial of the same degree. Indded, the roots of a* `HFE` *polynomial correspond to the zeros of a system of n boolean equations in n variables (see (3)). In [38], the authors studied the distribution of the number of zeroes of algebraic systems. In particular, a random system of n equations in n variables has exactly s solutions with probability $\frac{1}{e\,s!}$. Thus, as also mentionned [59], the threshold D in Step 10 can be theoretically much decreased without compromising the proof. The authors of [59] mentioned a value around ≈ 30 for the threshold.*

The second difference between GeMSS and GeMSS* is on the number of iterations. The treatment of [59] did not include the use of a Feistel-Patarin transform. It is an interesting open problem to formally prove EUF-CMA security when nb_ite > 0. This should probably follow from the use of Theorem 2.

All in all, the provable security results mentioned up to know only require minor modifications of the signature process without changing the underlying trapdoor. As a consequence, the security of GeMSS has to be mainly studied with respect to the hardness of inverting the public-key. This question is investigated in Section 8.

6.3 Signature failure

This analysis is essentially similar to the one performed for QUARTZ [54]. A failure can occurs in GeMSS.Inv$_{\mathbf{p}}$ (Algorithm 3), at Step 8, if Roots $= \emptyset$ for all $(\mathbf{r}, \mathbf{v}) \in \mathbb{F}_2^{n-m} \times \mathbb{F}_2^{v}$. The probability that Roots is empty for a given $(\mathbf{d}, \mathbf{v}) \in \mathbb{F}_2^{n-n+v} \times \mathbb{F}_2^{v}$ is $1/e$ [54, 38]. Thus, Algorithm 7 fails with probability $(\frac{1}{e})^{2^{n+v-m}}$.

Finally, GeMSS.Inv$_{\mathbf{p}}$ is called GeMSS.Sign nb_ite times. The probability of failure for GeMSS.Sign is then:

$$1 - \left(1 - \left(\frac{1}{e}\right)^{2^{n+v-m}}\right)^{\text{nb\_ite}}.$$

7 Expected strength (2.B.4) for each parameter set

7.1 Parameter set sign/GeMSS128

Category 1.

7.2 Parameter set sign/GeMSS192

Category 3.

7.3 Parameter set sign/GeMSS256

Category 5.

8 Analysis of known attacks (2.B.5)

This part provides a summary of the main attacks against GeMSS. In Section 8.1, we consider direct signature forgery attacks. This includes, in particular, the analysis of known quantum attacks (Sections 8.1.2 and 8.3) and Gröbner basis attacks (Sections 8.1.2 and 8.3). In Section 8.4, we consider key-recovery attacks.

In almost all cases, the attacks reduce to solving a particular system of non-linear equations derived from the public polynomials.

8.1 Direct signature forgery attacks

The public-key of GeMSS is given by a set of non linear-equations $\mathbf{p} = (p_1, \ldots, p_m) \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^m$. Given a digest $(d_1, \ldots, d_m) \in \mathbb{F}_2^m$, the problem of forging a signature is equivalent to solve the following system of non-linear equations:

$$p_1(x_1, \ldots, x_{n+v}) - d_1 = 0, \ldots, p_m(x_1, \ldots, x_{n+v}) - d_m = 0, x_1^2 - x_1, \ldots, x_{n+v}^2 - x_{n+v} = 0. \quad (7)$$

Stated differently, the task is to invert GeMSS.$\mathrm{Inv}_{\mathbf{p}}$ (Algorithm 3) without the knowledge of the secret-key sk.

In our case, the system is under-defined, i.e. $n + v > m$. As a consequence, we can randomly fix $n + v - m$ variables $\mathbf{r} = (r_1, \ldots, r_{n+v-m}) \in \mathbb{F}_2^{n+v-m}$ in (7) and try to solve for the remaining variables. Note that this is similar to the (legitimate) signature process which requires to randomly fix variables in GeMSS.$\mathrm{Inv}_{\mathbf{p}}$ (Steps 3 and 6 of Algorithm 3).

Thus, the problem of forging a signature reduces to solve a system of m quadratic equations in m variables over $\mathbb{F}_2$:

$$p_1(x_1, \ldots, x_m, \mathbf{r}) - d_1 = 0, \ldots, p_m(x_1, \ldots, x_m, \mathbf{r}) - d_m = 0, x_1^2 - x_1, \ldots, x_m^2 - x_m = 0. \quad (8)$$

8.1.1 Exhaustive search

In [11], the authors describe a fast exhaustive search for solving systems of boolean quadratic equations. They also provide a detailed cost analysis of their approach. To recover a solution of (8), the approach from [11] requires:

$$4 \log_2(m)\, 2^m \text{ binary operations.}$$

For the parameters of GeMSS, we have:

m	Fast exhaustive search ([11])
162	$2^{166.87}$
243	$2^{247.98}$
324	$2^{329.98}$

8.1.2 Quantum exhaustive search

In [17], the authors proposed simple quantum algorithms for solving systems of quadratic boolean equations. The principle of [17] is to perform a fast quantum exhaustive search by using Grover's algorithm. [17] demonstrated that we can solve a system of $m - 1$ binary quadratic equations in $n - 1$ binary variables using $m + n + 2$ qubits and evaluating a circuit of $2^{n/2}\Big(2m(n^2 + 2n) + 1\Big)$ quantum gates. They also describe a variant using less qubits, i. e. $3 + n + \lceil \log_2(m) \rceil$ qubits, but requiring to evaluate a larger circuit, i.e. with $\approx 2 \times 2^{n/2}\Big(2m(n^2 + 2n) + 1\Big)$ quantum gates.

We can now estimate is the cost for solving the system (8) for the parameters of GeMSS. The quantum attacks from [17] require then:

m	#qbits	#quantum gates
162	328	$2^{104.56}$
162	173	$\approx 2^{105.56}$
243	490	$2^{146.8}$
243	254	$\approx 2^{146.8}$
324	652	$2^{188.54}$
324	336	$\approx 2^{189.54}$

8.2 Approximation algorithm

Recently, the authors of [47] proposed a new algorithm for solving systems of non linear equations that is faster than a direct exhaustive search. The techniques from [47] allow for the approximation of a non-linear system, as (8), by a single high-degree multivariate polynomial P with $m' < m$ variables. The polynomial P is constructed such that it vanishes on the same zeroes as the original non-linear system with high probability. We then perform an exhaustive search on P to recover, with high probability, the zeroes of the non-linear system. This leads to an algorithm for solving (8) whose asymptotic complexity is:

$$O^*\left(2^{0.8765\,m}\right).$$

The notation O^* omits polynomial factors. Anyway, we will estimate the cost of this attack by the lower bound $2^{0.8765\,m}$.

For the parameters of GeMSS, we have then:

m	Lower bound on the complexity of [47]
162	$2^{141.99}$
243	$2^{212.98}$
324	$2^{283.98}$

8.3 Gröbner bases

To date, the best methods for solving non-linear equations, including the attack system (8), utilize Gröbner bases [15, 14]. The historical method for computing such bases – known as Buchberger's algorithm – has been introduced by Buchberger in his PhD thesis [15, 14]. Many improvements on Buchberger's algorithm have been done leading – in particular – to more efficient algorithms such as the F4 and F5 algorithms of J.-C. Faugère [30, 31]. The F4 algorithm, for example, is the default algorithm for computing Gröbner bases in the computer algebra software MAGMA [10]. The F5 algorithm, which is available through the FGb [33] software[5], provides today the state-of-the-art method for computing Gröbner bases.

Besides F4 and F5, there is a large literature of algorithms computing Gröbner bases. We mention for instance `PolyBory` [13] which is a general framework to compute Gröbner basis in $\mathbb{F}_2[x_1, \ldots, x_n]/\langle x_i^2 - x_i \rangle_{1 \le i \le n}$. It uses a specific data structure – dedicated to the Boolean ring – for computing Gröbner basis on top of a tweaked Buchberger's algorithm[6]. Another technique

[5] `http://www-polsys.lip6.fr/~jcf/FGb/index.html`

[6] `http://polybori.sourceforge.net`

proposed in cryptography is the `XL` algorithm [21]. It is now clearly established that `XL` is a special case of Gröbner basis algorithm [1]. More recently, a zoo of algorithms such as `G2V` [40], `GVW` [41], ..., flourished building on the core ideas of F4 and F5. This literature is vast and we refer to [29] for a recent survey of these algorithms.

Despite this important algorithmic literature, if is fair to say that MAGMA and FGb remain the references softwares for polynomial system solving over finite fields. We have intensively used both softwares to perform practical experiments and support our methodology to derive secure parameters (Section 8.3.3).

8.3.1 Asymptotically fast algorithms

`BooleanSolve` [6] is the fastest asymptotic algorithm for solving system of non-linear boolean equations. `BooleanSolve` is a hybrid approach that combines exhaustive search and Gröbner bases techniques. For a system with the same number of equations and variables (m), the deterministic variant of `BooleanSolve` has complexity bounded by $O(2^{0.841m})$, while a Las-Vegas variant has expected complexity

$$O(2^{0.792 \cdot m}).$$

It is mentioned in [6] that `BooleanSolve` is better than exhaustive search when $m \geq 200$. This is due to the fact that large constants are hidden in the big-O notation. As a conservative choice, we lower bound here the cost of this attack by $2^{0.792 \cdot m}$. We mention that [56] recently considered a hybrid approach against `HFEv-`. The former result also indicates that our approach is indeed conservative.

In Table 3, we report the security level of GeMSS against `BooleanSolve` (probabilistic version) for the three parameters proposed.

m	Lower bound on the cost of `BooleanSolve` ($2^{0.792 \cdot m}$)
162	$2^{128.3}$
243	$2^{192.45}$
324	$2^{256.6}$

Table 3: Security of GeMSS against `BooleanSolve`.

In fact, we have used `BooleanSolve` as the reference approach to derive the minimal number m of equation required in GeMSS.

`QuantumBooleanSolve`. In a recent paper [35], the authors present a quantum version of `BooleanSolve` that takes advantages of Grover's quantum algorithm [44]. `QuantumBooleanSolve` is a Las-Vegas quantum algorithm allowing to solve a system of m boolean equations in m variables. It uses $O(n)$ qbits, requires the evaluation of, on average, $O(2^{0.462m})$ quantum gates. This complexity is obtained under certain algebraic assumptions.

In Table 4, we report the security level of GeMSS against `QuantumBooleanSolve` (probabilistic version) for the three parameters proposed.

m	Lower bound on the # quantum gates for `QuantumBooleanSolve` ($2^{0.462 \cdot m}$)
162	$2^{74.84}$
243	$2^{112.26}$
324	$2^{149.68}$

Table 4: Security of GeMSS against `QuantumBooleanSolve`.

8.3.2 Practically fast algorithms

The direct attack described in [32, 36] provides reference tools for evaluating the security of `HFE` and `HFE`v- against a direct message-recovery attack. This attack uses the F5 algorithm [31, 4] and has a complexity of the following general form:

$$O\big(\mathrm{poly}(m, n)^{\omega \cdot D_{\mathrm{reg}}}\big), \tag{9}$$

with $2 \leq \omega < 3$ being the so-called *linear algebra constant* [61], i.e. the smallest constant $\omega, 2 \leq \omega < 3$ such that two matrices of size $N \times N$ over a field $\mathbb{F}$ can be multiplied in $O(N^{\omega})$ arithmetic operations over $\mathbb{F}$. The best current bound is $\omega < 2.3728639$ [39]. In this part, we will always use $\omega = 2$ to evaluate the cost of Gröbner bases attacks.

The complexity (9) is exponential in the *degree of regularity* D_{reg} [2, 5, 3]. However, this degree of regularity D_{reg} can be difficult to predict in general ; as difficult than computing a Gröbner basis. Fortunately, there is a particular class of systems for which this degree can be computed efficiently and explicitly : *semi-regular sequences* [2, 5, 3]. This notion is supposed to capture the behavior of a random system of non-linear equations. In order to set the parameters for `HFE` and variants as well than for performing meaningful experiments on the degree of regularity, we can assume that no algebraic system has a degree of regularity higher than a semi-regular sequence.

In Table 5, we provide the degree of regularity of a semi-regular system of m boolean equations in m variables for various values of m.

m	D_{reg}
$4 \leq m \leq 8$	3
$9 \leq m \leq 15$	4
$16 \leq m \leq 24$	5
$25 \leq m \leq 31$	6
$32 \leq m \leq 40$	7
$41 \leq m \leq 48$	8
$49 \leq m \leq 57$	9
$58 \leq m \leq 66$	10
$154 \leq m \leq 163$	20
$234 \leq m \leq 243$	28
$316 \leq m \leq 325$	36

Table 5: Degree of regularity of m semi-regular boolean equations in m variables.

In the case of `HFE`, the degree of regularity for solving (8) has been experimentally shown to be smaller than $\log_2(D)$ [32, 36]. This behavior has been further demonstrated in [43, 28]. In particular, [43] claims that the degree of regularity reached in HFE is asymptotically upper bounded by:

$$(2+\epsilon)(1-\sqrt{3/4}) \cdot \min\big(m, \log_2(D)\big), \text{ for all } \epsilon > 0. \tag{10}$$

This bound is obtained by estimating the degree of regularity of a semi-regular system of $3\lceil \log_2(D) \rceil$ quadratic equations in $2\lceil \log_2(D) \rceil$ variables. We emphasize that an asymptotic bound such as (10) is not necessarily tight for specified values of the parameters. Thus, (10) can not be directly used to derive actual parameters but still provide a meaningful asymptotic trend.

Indeed, the behavior of HFE algebraic systems is then much different from a semi-regular system of m boolean equations in m variables where the degree of regularity increases linearly with m. Roughly, D_{reg} grows as $\approx m/11.11$ in the semi-regular case [2, 5, 3].

We report below the degree of regularity $D_{\text{reg}}^{\text{Exp}}$ observed in practice for HFE. These bounds are are only meaningful for a sufficiently large m which is given in the first column. Indeed, as we already explained, we can assume that the values from Tab. 5 are upper bounds on the degree of regularity of any algebraic system of boolean equations.

Minimal m	HFE(D)	$D_{\text{reg}}^{\text{Exp}}$
$\geqslant 4$	$3 \leq D \leq 16$	3
$\geqslant 9$	$17 \leq D \leq 128$	4
$\geqslant 16$	$129 \leq D \leq 512$	5
$\geqslant 25$	$513 \leq D \leq 4091$	6
$\geqslant 32$	$D \geq 4092$	7

Table 6: Degree of regularity in the case of HFE algebraic systems.

Following [36], we lower bound the complexity of F5 against HFE, i.e. for solving the attack system (8). The principle is to only consider the cost of performing a row-echelon computation on a full rank sub-matrix of the biggest matrix occurring in F5. At the degree of regularity, this sub-matrix has $\binom{m}{D_{\text{reg}}}$ columns and (at least) $\binom{m}{D_{\text{reg}}}$ rows. Thus, we can bound the complexity of a Gröbner basis computation against HFE by:

$$O\left(\binom{m}{D_{\text{reg}}}^2\right). \tag{11}$$

This is a conservative estimate on the cost of solving (8). This represents the minimum computation that has to be be done in F5. We also assumed that the linear algebra constant ω is 2; the smallest possible value.

Given a value of m, we can now deduce from (11) and Table 3, the (smallest) degree of regularity required to achieve a certain security level. These values are given in Table 7.

From Table (6), we can see that no HFE has a degree of regularity sufficiently large to achieve a reasonable level of security. To do so, we need to use modifiers of HFE for increasing the degree of regularity.

In particular, the practical effect of the minus and vinegar modifiers have been considered in [32, 36]. This has been further investigated in [23, 26] who presented a theoretical upper bound on the degree

m	minimal D_{reg} required	Lower bound on the cost of a Gröbner basis as given in (11)
162	14	$2^{131.16}$
243	20	$2^{192.52}$
324	27	$2^{260.86}$

Table 7: Smallest degree of regularity required.

of regularity arising in HFEv-. Let $R = \lfloor \log_2(D-1) \rfloor + 1$, then the degree of regularity for HFEv- is bounded from above by

$$\frac{R+v+\Delta-1}{2} + 2, \quad \text{when } R+\Delta \text{ is odd,} \tag{12}$$

$$\frac{R+v+\Delta}{2} + 2, \quad \text{otherwise.} \tag{13}$$

We observe that degree of regularity seems to increase linearly with $(n+v-m)$. This is the sum of the modifiers : number of equations removed plus vinegar variables.

Very recently, [56] derived an experimental *lower bound* on the degree of regularity in HFEv-. The authors [56] obtained that the degree of regularity for HFEv- should be at least :

$$\left\lceil \frac{R+\Delta+v+7}{3} \right\rceil. \tag{14}$$

8.3.3 Experimental results for HFEv-

The main question in the design of GeMSS is to quantify, as precisely as possible, the effect of the modifiers on the degree of regularity. To do so, we performed experimental results on the behaviour of a direct attack against HFEv-, i.e. computing a Gröbner basis of (8). We mention that similar experiments were performed in [57].

We first consider $v = 0$, and denote by Δ the number of equation removed, i.e. $m = n - r$. According to the upper bounds (12) and (13), the degree of regularity should increase by 1 when 2 equations are removed.

We report the degree of regularity $D_{\mathrm{reg}}^{\mathrm{Exp}}$ reached during a Gröbner basis computation of a system of $m = n - \Delta$ equations in $n - \Delta$ variables coming from a HFE public-key generated from a univariate polynomial in $\mathbb{F}_{2^n}[X]$ of degree D. We also reported the degree of regularity $D_{\mathrm{reg}}^{\mathrm{Theo}}$ of a semi-regular system of the same size (as in Table (5)).

The experimental results on HFE-, no vinegar, are not completely conclusive. Whilst the degree of regularity appears to increase, it seems difficult to predict its behavior in function of the number of equations removed. This was also observed in [57] where the authors advised against using the minus modifier alone. Thus, the minus modifier should not be used alone.

We now consider the opposite situation, i.e. no minus and we increase the number of vinegar variables, i.e. HFEv.

The experimental results are more stable. In all cases, we need to add 3 vinegar variables to increase the degree of regularity by 1.

n	Δ	$n-\Delta$	D	$D_{\text{reg}}^{\text{Theo}}$	$D_{\text{reg}}^{\text{Exp}}$
32	0	32	4	7	3
33	1	32	4	7	3
34	2	32	4	7	3
35	**3**	32	4	7	**4**
36	4	32	4	7	4
37	5	32	4	7	4
38	6	32	4	7	4
39	7	32	4	7	4
40	**8**	32	4	7	**5**
41	9	32	4	7	5
42	10	32	4	7	5
43	11	32	4	7	5
44	12	32	4	7	5
45	13	32	4	7	5
46	**14**	32	4	7	**6**
47	15	32	4	7	6
48	16	32	4	7	6
49	17	32	4	7	6
49	18	32	4	7	6
50	19	32	4	7	6
51	20	32	4	7	6

n	Δ	$n-\Delta$	D	$D_{\text{reg}}^{\text{Theo}}$	$D_{\text{reg}}^{\text{Exp}}$
41	0	41	4	8	3
42	1	41	4	8	3
43	2	41	4	8	3
44	**3**	41	4	8	**4**
45	4	41	4	8	4
46	5	41	4	8	4
47	6	41	4	8	4
48	7	41	4	8	4

Table 8: `HFE-` with $D = 4$; 32 and 41 equations.

n	Δ	$n-\Delta$	D	$D_{\text{reg}}^{\text{Theo}}$	$D_{\text{reg}}^{\text{Exp}}$
32	0	32	17	7	4
33	1	32	17	7	4
34	2	32	17	7	4
35	**3**	32	17	7	**5**
36	4	32	17	7	5
37	**5**	32	17	7	**6**
38	6	32	17	7	6
39	7	32	17	7	6

n	Δ	$n-\Delta$	D	$D_{\text{reg}}^{\text{Theo}}$	$D_{\text{reg}}^{\text{Exp}}$
41	0	41	17	8	4
42	1	41	17	8	4
43	2	41	17	8	4
44	**3**	41	17	8	**5**
45	4	41	17	8	5

Table 9: `HFE-` with $D = 17$; 32 and 41 equations.

n	v	$m = n - v$	D	$D_{\text{reg}}^{\text{Theo}}$	$D_{\text{reg}}^{\text{Exp}}$
32	**0**	32	6	7	**3**
32	**7**	25	6	7	**5**
32	**8**	25	6	7	**6**
32	9	25	6	7	6
32	10	25	7	7	6
32	**11**	25	6	7	**7**
32	12	25	6	7	7
32	15	25	6	7	7

Table 10: `HFEv`, $D = 6$ and 32 variables.

We also performed experimental results with a combination of vinegar and minus. Similarly to [57], we observed that the behaviour obtained seems similar for `HFEv-` with $\Delta = 0$ and v vinegar

n	v	$m = n - v$	D	$D_{\text{reg}}^{\text{Theo}}$	$D_{\text{reg}}^{\text{Exp}}$
25	**0**	25	9	6	**3**
26	**1**	25	9	6	**4**
27	2	25	9	6	4
28	3	25	9	6	4
29	**4**	25	9	6	**5**
30	5	25	9	6	5
31	6	25	9	6	5
32	7	25	9	6	**6**

Table 11: `HFEv`, $D = 9$ and 25 variables.

n	v	$m = n - v$	D	$D_{\text{reg}}^{\text{Theo}}$	$D_{\text{reg}}^{\text{Exp}}$
25	**0**	25	16	6	**3**
26	**1**	25	16	6	**4**
27	2	25	16	6	4
28	3	25	16	6	4
29	**4**	25	16	6	**5**
30	5	25	16	6	5
31	6	25	16	6	5
32	**7**	25	16	6	**6**

n	v	$m = n - v$	D	$D_{\text{reg}}^{\text{Theo}}$	$D_{\text{reg}}^{\text{Exp}}$
32	**0**	32	16	7	**3**
33	**1**	32	16	7	**4**
34	2	32	16	7	4
35	3	32	16	7	4
36	**4**	32	16	7	**5**
37	5	32	16	7	5

Table 12: `HFEv` with $D = 16$; 25 and 32 equations.

variables than for a `HFEv-` with $\Delta = v/2$ and $v/2$ vinegar variables.

8.4 Key-recovery attacks

We conclude this part by covering key-recovery attacks. This part discusses the so-called *Kipnis-Shamir attack* [46] (Section 8.4.1) and differential attacks (Section 8.4.2).

8.4.1 Kipnis-Shamir attack

In [46], A. Kipnis and A. Shamir demonstrated that key-recovery in `HFE` is essentially equivalent to the problem of finding a low-rank linear combination of a set of m boolean matrices of size $m \times m$. This is a particular instance of the `MinRank` problem [16, 19].

We briefly review the principle of this attack for `HFE`. In the context of this attack, we can assume w.l.o.g. that the `HFE` polynomial has a simpler form:

$$\sum_{\substack{0 \leqslant i < j < n \\ 2^i + 2^j \leqslant D}} A_{i,j} \, X^{2^i + 2^j} \in \mathbb{F}_{2^n}[X], \text{ with } A_{i,j} \in \mathbb{F}_{2^n}. \tag{15}$$

We can then write (15) in a matrix form, that is:

$$\underline{X}\mathbf{F}\underline{X}^{\mathrm{T}}$$

with $\underline{X} = (X, X^2, X^{2^2}, \ldots, X^{2^{n-1}})$ and $\mathbf{F} \in \mathcal{M}(\mathbb{F}_{2^n})_{n\times n}$ is a symmetric matrix with zeroes on the diagonal (i.e. skew-symmetric matrix). Since the degree of F is bounded by D, it is easy to see that $\mathbf{F}$ has rank at most $\lceil \log_2(D) \rceil$. This implies that there exists a linear combinations of rank $\lceil \log_2(D) \rceil$ of the public matrices representing the public quadratic forms [7]. The secret-key can be then recovered easily from a solution of `MinRank` [46, 7].

In [7], the authors evaluated the cost of the Kipnis-Shamir key-recovery attack with the best known tools for solving the `MinRank` [34] instance that occurs in `HFE`. Following [7], the cost of the Kipnis-Shamir attack against `HFE` can be estimated to:

$$O\big(n^{\omega(\lceil \log_2(D) \rceil+1)}\big), \text{ with } 2 \le \omega \le 3 \text{ being the linear algebra constant}$$

and where D is the degree of the secret univariate polynomial.

Until recently, it was not clear how to apply the key-recovery attack from [46, 7] to `HFE-` when $n - m \ge 2$. In [60], the authors explained how to extend `MinRank`-based key-recovery for all parameters of `HFE-`. Their results can be summarized as follows. From key-recovery point of view, `HFE-` with a secret univariate polynomial of degree D and n variables is equivalent to a `HFE` with m variables with secret univariate polynomial of degree $D \times 2^{\Delta}$. Combining with [7], the cost of a `MinRank`-based key-recovery attack against `HFE`$-$ is then:

$$O\big(m^{\omega(\lceil \log_2(D) \rceil+\Delta+1)}\big).$$

For `MinRank`-based key-recovery, then minus modifier has then a strong impact on the security.

In the case of `HFEv`, one can see that the rank of the corresponding matrix (see, for exemple [57]) will be increased by the number of vinegar variables. Combining with the previous result, the cost of solving `MinRank` in the case of `HFEv-` is then:

$$O\big(n^{\omega(\lceil \log_2(D) \rceil+v+\Delta+1)}\big), \tag{16}$$

where D is the degree of the secret univariate polynomial.

For all the parameters proposed for scheme, assuming $\omega = 2$, the cost (16) is always much bigger than the cost of the best direct attack (Section 8.1).

Remark 4. *Recently, [24] proposed set of new attacks whose complexity remains essentially exponential in the parameters. This attacks improved known attacks for some parameters. We quickly verified the complexity of these attacks. They don't decrease the security of* GeMSS *below the security parameter.*

8.4.2 Differential attack

We finally consider so-called *differential attacks*, introduced [27], are structural attacks that can be used to attack multivariate cryptosystems. Differential attacks turned to be very efficient, e.g. [27, 12] against `SFLASH` [53]; a popular multivariate-based signature based on the Mastsumoto and Imai [48].

HFE is the successor, and a generalization, of [48]. Up to know, differential attacks have not really threatened the security of HFEv-. This is due to the fact the univariate polynomial used is much more complex than in [48] variants such as SFLASH [53]. In [18], the authors proved that variants of HFE, such as G*e*MSS, are immune against known differential attacks.

8.5 Deriving number of variables for G*e*MSS

At this stage, we have a methodology for fixing the minimal number of equations m (Table 3). We now need to derive the number of vinegar variables v and minus Δ required to achieve the degree of regularity corresponding to a given security level (Table 7). This is the most delicate point. According to the experiments performed in Section 8.3.3, and the insight provided by the key-recovery attacks (Section 8.4), we make the choice to balance v and Δ.

In addition, we need to fix the degree D of the HFEv polynomial. This will give the initial degree of regularity for a nude HFE (Table 6). For G*e*MSS, we consider a secret univariate polynomial of degree $D = 513$. This corresponds to a degree of regularity of 6 for a nude HFE, i.e. without any modifier. From our experiments, we consider that 3 modifiers allow to increase the degree of regularity by one. Idenpendently of this submission, the authors [56] also derived a similar rule; as one can see from (14).

In Table 13, we then derive the number of modifiers required as $v + \Delta = 3 \times \text{Gap}$, with Gap being the difference with the targeted degree of regularity minus the initial degree of regularity (6 here). We consider the number of equations m and the targeted degree of regularity as in Table 7. The third column of Table 13 gives the number of modifiers required.

	m	Gap	$v + \Delta$
G*e*MSS128	162	$14 - 6 = 8$	24
G*e*MSS192	243	$20 - 6 = 14$	42
G*e*MSS256	324	$27 - 6 = 21$	63

Table 13: Numbers of modifiers required in G*e*MSS.

The exact resulting parameters are given in Section 14.

8.6 A general method to derive secure parameters

We are now in position to provide a general methodology to derive secure parameters for G*e*MSS. Following Section 8.3.1, the number of equations should be chosen such that:

$$m \geq 1.26 \cdot \lambda.$$

Thus, we can assume that $m = \alpha \cdot \lambda$ with $\alpha \geq 1.26$.

From (11), the degree of regularity D_{reg} required for a given security level should verify:

$$O\left(\binom{m}{D_{\text{reg}}}^2\right) \geq 2^{\lambda}.$$

Using a loose approximation of the binomial and ignoring the coefficient in the big-O, we get that:

$$D_{\text{reg}} \geq \frac{\lambda}{\log_2(m^2)} = \frac{\lambda}{2\log_2(\alpha \cdot \lambda)}.$$

The last step requires to compute the number of vinegar variables required to reach D_{reg}. We first need to have the initial degree of regularity. We can assume that this is a function of $\log_2(D)$; as explained in Section 8.3.2. From table 6, we can interpolate an expression for the degree of regularity $D_{\text{reg}}^{\texttt{HFE}}$ of a nude HFE:

$$D_{\text{reg}}^{\texttt{HFE}} \approx 2.03 + 0.36\log_2(D).$$

The number of modifiers, using the experimental rule of Section 8.5, can be then approximated by:

$$\Delta + v \approx \frac{3\lambda}{\log_2(m^2)} - 6.06 - 1.08\log_2(D) = \frac{1.5\lambda}{\log_2(\alpha \cdot \lambda)} - 1.08\log_2(D) - 6.06. \tag{17}$$

Below, we computed this approximation for the parameters of GeMSS.

(λ, m, D)	Approximation (17) of $\Delta + v$
$(128, 162, 512)$	36.53
$(196, 243, 512)$	56.9
$(256, 324, 512)$	76.30

This has to be compared with the exact values provided in Table 13. The difference is mainly due to the loose approximation of the binomial for deriving (17). However, we can see that (17) captures rather well the global trend and can be used to derive others secure parameters.

We can see that there is two strategies to derive secure parameters. In GeMSS, the goal is to minimize the size of the public-key. To do so, we are taking $m = 1.26 \cdot \lambda$. From (17), we can see that the number of modifiers decreases when D increases. We take the same number of vinegar variables v and the same number of minus Δ. To minimize the total number of variables m, we have then to increase the degree D of the univariate polynomial. However, the time to sign increases with D.

The strategy differs if the goal is to have a faster signing process together with a shorter signature. In this case, we have to take m bigger than $1.26 \cdot \lambda$. As a consequence, the number of iterations nb_ite can be decreased. We repeat then less the inversion process GeMSS.$\text{Inv}_{\mathbf{p}}$ in GeMSS.$\text{Inv}_{\mathbf{p}}$. The verification will be also faster. From (17), we can see that maximizing the number of modifiers makes possible to choose smaller D. However, this will increase the number of vinegar variables v and so the total number of variables m.

For 128 bits of security, we can take for example $m = 256$. In this case nb_ite can be set to 1. For $D = 129$, the total number of verifiers should be 21. We can take then $v = 11$ and $\Delta = 10$. The total number of variables is then 277 and we can choose $D = 129$. The size of the public-key is 1.14 MB but the time to sign is ≈ 3 ms. (To verify)

9 A larger family of GeMSS parameters

9.1 Why more parameters

In multivariate schemes, we have many parameters that can be adjusted. This is an advantage since, for example, for a given security we can decrease the time to sign if we increase the length of the public key, i.e. some interesting tradeoffs are possible.

However, when a new cryptanalysis idea is found, it is not always easy for a non multivariate specialist to see how to adjust the parameters in order to maintain a given security level against the best known attacks. For example, when RSA-512 was factored, it was natural to suggest to use a larger modulo and to look at what value of n should be used from the best known attacks (instead of designing another scheme). But when an attack on QUARTZ was published with a security expected [36] to be slightly smaller than 2^{80} it was not so easy to adjust the security parameters since we have here many possibilities. Therefore, we see that it is sometime convenient to have a "*dimension 1*" family instead of a single point (like QUARTZ) or a many dimension family (like the variants of HFE). This is why in GeMSS we will:

- Suggest 2 or 3 sets of parameters for expected security in 2^{128}, in order to have some interesting tradeoffs with this security level. (However, we give only experimental results for the first set of parameter given Section 3).
- Design a family of possible values that depends on only one parameter n. We call this family FGeMSS(n). Then, when FGeMSS(n) is broken for a value $n \leq n_0$, we can adjust to larger values of n where FGeMSS(n) is not broken.

9.2 Parameters for expected security in 2^{128}

9.2.1 Set 1 of parameters (see Section 3)

This is GeMSS128.
Time to sign: 323 ms.
Size of public key: 352 Kbytes.

9.2.2 Set 2 of parameters

- nb_ite = 1
- $m = 256$
- $D = 129$
- $\Delta = v = 21$.
- Time to sign: about 3 ms.
- Size of the public key: 1.14 Mbytes

We call this scheme FastGeMSS128. Thus by using a public key 3.32 times larger, the time to sign is 100 times faster. This is because in this Set 2 of parameter we use a smaller D and nb_ite = 1 (instead of 4).

9.2.3 Set 3 of parameters

It is expected that FGeMSS(n) (see below) will have a security of about 2^{128} when n is about 187 from the best known attacks at present.

9.2.4 FGeMSS(n) family

Here we want to design a family of "*dimension 1*". In this family, called FGeMSS(n), we will have:

- nb_ite = 1
- n is again $m + \Delta$
- $\Delta = v = 15 + \lceil 0.1(n - 187) \rceil$
- $D = \lceil 3.82n - 454 \rceil$

10 Advantages and limitations (2.B.6)

Since the first scheme of Mastumoto and Imai [48] in 1988, almost 30 years ago, multivariate-based cryptosystems have been extensively analysed in the literature. We have designed GeMSS using this knowledge and taking conservative choices for deriving parameters. We also performed practical experiments using the best known tools for computing Gröbned bases.

From a practical point of view, the main drawback of GeMSS is the size of the public-key. However, we mention that the generation of a (public-key,secret-key) remains rather efficient in GeMSS. The main advantages of GeMSS are the size of the signatures generated, about 2λ bits, and the fast verification process.

References

[1] Gwénolé Ars, Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita. Comparison between XL and gröbner basis algorithms. In Pil Joong Lee, editor, *Advances in Cryptology - ASIACRYPT 2004, 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004, Proceedings*, volume 3329 of *Lecture Notes in Computer Science*, pages 338–353. Springer, 2004.

[2] Magali Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie.* PhD thesis, Université de Paris VI, 2004.

[3] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *International Conference on Polynomial System Solving – ICPSS*, pages 71–75, 2004.

[4] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the Complexity of the F5 Gröbner basis Algorithm. *Journal of Symbolic Computation*, pages 1–24, September 2014. 24 pages.

[5] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Bo-Yin Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In *The Effective Methods in Algebraic Geometry Conference – MEGA 2005*, pages 1–14, 2005.

[6] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Pierre-Jean Spaenlehauer. On the complexity of solving quadratic boolean systems. *Journal of Complexity*, 29(1):53–75, February 2013.

[7] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Cryptanalysis of hfe, multi-hfe and variants for odd and even characteristic. *Des. Codes Cryptography*, 69(1):1–52, 2013.

[8] Olivier Billet and Jintai Ding. *Overview of Cryptanalysis Techniques in Multivariate Public Key Cryptography*, pages 263–283. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[9] Andrey Bogdanov, Thomas Eisenbarth, Andy Rupp, and Christopher Wolf. Time-area optimized public-key engines: Mq-cryptosystems as replacement for elliptic curves? *IACR Cryptology ePrint Archive*, 2008:349, 2008.

[10] Wieb Bosma, John J. Cannon, and Catherine Playoust. The Magma algebra system I: The user language. *Journal of Symbolic Computation*, 24(3-4):235–265, 1997.

[11] Charles Bouillaguet, Hsieh-Chung Chen, Chen-Mou Cheng, Tung Chou, Ruben Niederhagen, Adi Shamir, and Bo-Yin Yang. Fast exhaustive search for polynomial systems in F_2. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 203–218. Springer, 2010.

[12] Charles Bouillaguet, Pierre-Alain Fouque, and Gilles Macario-Rat. Practical key-recovery for all possible parameters of SFLASH. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 667–685. Springer, 2011.

[13] Michael Brickenstein and Alexander Dreyer. Polybori: A framework for gröbner-basis computations with boolean polynomials. *J. Symb. Comput.*, 44(9):1326–1345, 2009.

[14] Bruno Buchberger. Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of Symbolic Computation*, 41(3-4):475–511, 2006.

[15] Bruno Buchberger, Georges E. Collins, Rudiger G. K. Loos, and Rudolph Albrecht. Computer algebra symbolic and algebraic computation. *SIGSAM Bull.*, 16(4):5–5, 1982.

[16] Jonathan F Buss, Gudmund S Frandsen, and Jeffrey O Shallit. The computational complexity of some problems of linear algebra. *Journal of Computer and System Sciences*, 58(3):572–596, 1999.

[17] Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors. *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*, volume 10076 of *Lecture Notes in Computer Science*. Springer, 2016.

[18] Ryann Cartor, Ryan Gipson, Daniel Smith-Tone, and Jeremy Vates. On the differential security of the hfev- signature primitive. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings*, volume 9606 of *Lecture Notes in Computer Science*, pages 162–181. Springer, 2016.

[19] Nicolas Courtois. Efficient zero-knowledge authentication based on a linear algebra problem minrank. In *ASIACRYPT*, volume 2248, pages 402–421. Springer, 2001.

[20] Nicolas Courtois. Generic attacks and the security of quartz. In *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 351–364. Springer, 2003.

[21] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer, 2000.

[22] Nicolas T. Courtois. Short signatures, provable security, generic attacks and computational security of multivariate polynomial schemes such as hfe, quartz and sflash. *IACR Cryptology ePrint Archive*, 2004:143, 2004.

[23] Jintai Ding and Thorsten Kleinjung. Degree of regularity for HFE-. *IACR Cryptology ePrint Archive*, 2011:570, 2011.

[24] Jintai Ding, Ray Perlner, Albrecht Petzoldt, and Daniel Smith-Tone. Improved cryptanalysis of hfev- via projection. Cryptology ePrint Archive, Report 2017/1149, 2017. https://eprint.iacr.org/2017/1149.

[25] Jintai Ding and Bo-Yin Yang. *Multivariate Public Key Cryptography*, pages 193–241. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[26] Jintai Ding and Bo-Yin Yang. Degree of regularity for HFEv and HFEv-. In Philippe Gaborit, editor, *Post-Quantum Cryptography - 5th International Workshop, PQCrypto 2013, Limoges, France, June 4-7, 2013. Proceedings*, volume 7932 of *Lecture Notes in Computer Science*, pages 52–66. Springer, 2013.

[27] Vivien Dubois, Pierre-Alain Fouque, Adi Shamir, and Jacques Stern. Practical cryptanalysis of SFLASH. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2007.

[28] Vivien Dubois and Nicolas Gama. The degree of regularity of HFE systems. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference*

on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings, volume 6477 of *Lecture Notes in Computer Science*, pages 557–576. Springer, 2010.

[29] Christian Eder and Jean-Charles Faugère. A survey on signature-based algorithms for computing gröbner bases. *J. Symb. Comput.*, 80:719–784, 2017.

[30] J.-C. Faugère. A new efficient algorithm for computing gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.

[31] J.-C. Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero : F5. In *ISSAC'02*, pages 75–83. ACM press, 2002.

[32] Jean-Charles Faugère. Algebraic cryptanalysis of HFE using Gröbner bases. Reasearch report RR-4738, INRIA, 2003.

[33] Jean-Charles Faugère. FGb: A Library for Computing Gröbner Bases. In Komei Fukuda, Joris Hoeven, Michael Joswig, and Nobuki Takayama, editors, *Mathematical Software - ICMS 2010*, volume 6327 of *Lecture Notes in Computer Science*, pages 84–87, Berlin, Heidelberg, September 2010. Springer Berlin / Heidelberg.

[34] Jean-Charles Faugère, Mohab Safey El Din, and Pierre-Jean Spaenlehauer. On the complexity of the generalized minrank problem. *Journal of Symbolic Computation*, 55:30–58, 2013.

[35] Jean-Charles Faugère, Kelsey Horan, Delaram Kahrobaei, Marc Kaplan, Elham Kashefi, and Ludovic Perret. Fast quantum algorithm for solving multivariate quadratic equations. To appear.

[36] Jean-Charles Faugère and Antoine Joux. Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using gröbner bases. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 2003.

[37] Pierre-Alain Fouque, Gilles Macario-Rat, Ludovic Perret, and Jacques Stern. Total break of the l-ic signature scheme. In Ronald Cramer, editor, *Public Key Cryptography - PKC 2008, 11th International Workshop on Practice and Theory in Public-Key Cryptography, Barcelona, Spain, March 9-12, 2008. Proceedings*, volume 4939 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2008.

[38] Giordano Fusco and Eric Bach. *Phase Transition of Multivariate Polynomial Systems*, pages 632–645. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[39] François Le Gall. Algebraic complexity theory and matrix multiplication. In Katsusuke Nabeshima, Kosaku Nagasaka, Franz Winkler, and Ágnes Szántó, editors, *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*, page 23. ACM, 2014.

[40] Shuhong Gao, Yinhua Guan, and Frank Volny, IV. A new incremental algorithm for computing groebner bases. In *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*, ISSAC '10, pages 13–19, New York, NY, USA, 2010. ACM.

[41] Shuhong Gao, Frank Volny IV, and Mingsheng Wang. A new framework for computing gröbner bases. *Math. Comput.*, 85(297), 2016.

[42] Louis Goubin and Nicolas Courtois. Cryptanalysis of the TTM cryptosystem. In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, volume 1976 of *Lecture Notes in Computer Science*, pages 44–57. Springer, 2000.

[43] Louis Granboulan, Antoine Joux, and Jacques Stern. Inverting HFE is quasipolynomial. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 345–356. Springer, 2006.

[44] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *STOC*, pages 212–219. ACM, 1996.

[45] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222. Springer, 1999.

[46] Aviad Kipnis and Adi Shamir. Cryptanalysis of the HFE public key cryptosystem by relinearization. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 19–30. Springer, 1999.

[47] Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, R. Ryan Williams, and Huacheng Yu. Beating brute force for systems of polynomial equations over finite fields. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2190–2202. SIAM, 2017.

[48] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynominal-tuples for efficient signature-verification and message-encryption. In *EUROCRYPT*, volume 330 of *Lecture Notes in Computer Science*, pages 419–453. Springer, 1988.

[49] Mohamed Saied Emam Mohamed and Albrecht Petzoldt. The shortest signatures ever. In Orr Dunkelman and Somitra Kumar Sanadhya, editors, *Progress in Cryptology - INDOCRYPT 2016 - 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings*, volume 10095 of *Lecture Notes in Computer Science*, pages 61–77, 2016.

[50] NESSIE. New european schemes for signatures, integrity, and encryption, 2003.

[51] Jacques Patarin. Cryptanalysis of the matsumoto and imai public key scheme of eurocrypt'88. In Don Coppersmith, editor, *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, volume 963 of *Lecture Notes in Computer Science*, pages 248–261. Springer, 1995.

[52] Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 1996.

[53] Jacques Patarin, Nicolas Courtois, and Louis Goubin. Flash, a fast multivariate signature algorithm. In *CT-RSA*, volume 2020 of *Lecture Notes in Computer Science*, pages 298–307. Springer, 2001.

[54] Jacques Patarin, Nicolas Courtois, and Louis Goubin. Quartz, 128-bit long digital signatures. In David Naccache, editor, *Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings*, volume 2020 of *Lecture Notes in Computer Science*, pages 282–297. Springer, 2001.

[55] Ludovic Perret. *Bases de Gröbner en Cryptographie Post-Quantique. (Gröbner bases techniques in Quantum-Safe Cryptography).* 2016.

[56] Albrecht Petzoldt. On the complexity of the hybrid approach on hfev-. Cryptology ePrint Archive, Report 2017/1135, 2017. https://eprint.iacr.org/2017/1135.

[57] Albrecht Petzoldt, Ming-Shing Chen, Bo-Yin Yang, Chengdong Tao, and Jintai Ding. Design principles for hfev- based multivariate signature schemes. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 311–334. Springer, 2015.

[58] ETSI ISG QSC. Quantum-safe cryptography (QSC); quantum-safe algorithmic framework. http://www.etsi.org/deliver/etsi_gr/QSC/001_099/001/01.01.01_60/gr_QSC001v010101p.pdf.

[59] Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari. On provable security of UOV and HFE signature schemes against chosen-message attack. In *PQCrypto*, volume 7071 of *Lecture Notes in Computer Science*, pages 68–82. Springer, 2011.

[60] Jeremy Vates and Daniel Smith-Tone. Key recovery attack for all parameters of HFE-. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*, volume 10346 of *Lecture Notes in Computer Science*, pages 272–288. Springer, 2017.

[61] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra (3. ed).* Cambridge University Press, 2013.

[62] Christopher Wolf. *Multivariate quadratic polynomials in public key cryptography.* Univ. Leuven Heverlee, 2005.