DualModeMS: A Dual Mode for Multivariate-based Signature

20170918 draft

Principal submitter

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

- J.-C. Faugère, INRIA and Sorbonne Universities/UPMC Univ Paris 06
- 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

The purpose of this document is to present DualModeMS; a multivariate-based signature scheme with a rather peculiar property. Its public-key is small whilst the signature is large. This is in sharp contrast with traditionnal multivariate signature schemes [44, 57, 23, 8, 52, 49] based on the so-called *Matsumoto and Imai* (MI) construction [44], such as QUARTZ [48] or Gui [24], that produce short signatures but have larger public-keys.

DualModeMS is based on the method proposed by A. Szepieniec, W. Beullens, and B. Preneel (SBP) in [54] where present a generic technique permitting to transform any MI-based multivariate signature scheme into a new scheme with much shorter public-key but larger signatures. We emphasize that the technique from [54] can be viewed as a *mode of operations* that offers a new flexibility for MI-like signature schemes. Thus, we believe that DualModeMS could also be useful for others multivariate-based signature candidates proposed to NIST.

DualModeMS is composed by two distinct layers. The first one (Inner.DualModeMS), that we shall call *inner layer* (Section 2.1), is a classical MI-like multivariate scheme based on HFEv. The second part, *outer layer* (Section 2.2), is the mode of operations specified in [54].

This submission is somewhat a complement to another multivariate-based signature scheme proposed to NIST : G*e*MSS [18]. In particular, the security analysis for Inner.DualModeMS is largely similar to the one performed for G*e*MSS. In fact, Inner.DualModeMS is a re-parametrization of G*e*MSS imposed by a specificity of SBP [54].

Acknowledgement. G*e*MSS 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 Description of the inner layer – Specification of Inner.DualModeMS

2.1.1 Parameter space

The main parameters involved in Inner.DualModeMS 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 security level of SHA3 (in bits),
- λ, the security level of Inner.DualModeMS,

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

- m, number of equations in the public-key,
- n, the degree of a field extension,
- v, the number of vinegar variables,
- Δ, the number of minus (the number of equations in the public-key is such that is $m = n - \Delta$).

In Section 8.6, we specify precisely these parameters to achieve a security level $\lambda \in \{128, 192, 256\}$.

2.1.2 Secret-key and public-key

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 [46], 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:

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

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 Inner.DualModeMS will be a function of D, n, v and m.

Algorithm 1 PK/SK generation in Inner.DualModeMS

```
1: procedure Inner.DualModeMS.KeyGen($1^\lambda$)
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 2.3.1.
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 2.3.2.
4:     sk ← $(F,\mathbf{S},\mathbf{T}) \in \mathbb{F}_2[X,v_1,\ldots,v_v] \times \mathrm{GL}_{n+v}(\mathbb{F}_2) \times \mathrm{GL}_n(\mathbb{F}_2)$
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 2.3.5 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:     pk ← $\mathbf{p} = (p_1,\ldots,p_m) \in \mathbb{F}_2[x_1,\ldots,x_{n+v}]^m$  ▷ Take the first $m = n - \Delta$ polynomials computed in Step 6
8:     return (sk, pk)
9: end procedure
```

2.1.3 Signing process

The main step of the signature process requires to solve:

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

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 [56, Algorithm 14.15].

Algorithm 2 Algorithm for finding the roots of an univariate polynomial

```
function FindRoots($F_{D'} \in \mathbb{F}_{2^n}[X]$)
    $X_n \leftarrow X^{2^n} - X \bmod F_{D'}$                ▷ This step is further detailed in Section 5.5
    $G \leftarrow \gcd(F_{D'}, X_n)$
    if degree($G$) $> 0$ then
        Roots $\leftarrow$ List of all roots of $G$, computed by the equal-degree factorization algorithm described in [56, Section 14.3]
        return (degree($G$), Roots)
    end if
    return (degree($G$), $\emptyset$)
end function
```

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

Theorem 1 (Corollary 14.16 from [56]). *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.:

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

The signature process in Inner.DualModeMS is then given in Algorithm 3:

Algorithm 3 Signing process in Inner.DualModeMS

```
 1: procedure Inner.DualModeMS.Sign($\mathbf{M} \in \{0,1\}^*, \mathsf{sk} \in \mathbb{F}_2[X, v_1, \ldots, v_v] \times \mathrm{GL}_{n+v}(\mathbb{F}_2) \times \mathrm{GL}_n(\mathbb{F}_2)$)
 2:     $\mathbf{d} \leftarrow$ first $m$ bits of SHA3($\mathbf{M}$)
 3:     repeat
 4:         $\mathbf{r} \in_R \mathbb{F}_2^{n-m}$        ▷ The notation $\in_R$ stands for randomly sampling.
 5:         $\mathbf{d}' \leftarrow (\mathbf{d}, \mathbf{r}) \in \mathbb{F}_2^n$
 6:         $D' \leftarrow \varphi^{-1}(\mathbf{d}' \times \mathbf{T}^{-1}) \in \mathbb{F}_{2^n}$
 7:         $\mathbf{v} \in_R \mathbb{F}_2^v$
 8:         $F_{D'}(X) \leftarrow F(X, \mathbf{v}) - D'$
 9:         $(\cdot, \mathrm{Roots}) \leftarrow \mathrm{FindRoots}(F_{D'})$
10:     until $\mathrm{Roots} = \emptyset$
11:     $Z \in_R \mathrm{Roots}$
12:     return $(\varphi(Z), \mathbf{v}) \times \mathbf{S}^{-1} \in \mathbb{F}_2^{n+v}$
13: end procedure
```

Remark 2. *We sample a roots at Step 11 always in the same way. First, we sort the elements of* Roots *in ascending order. We then compute* $\mathtt{SHA3}(D')$*, 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.

The signature process described here for Inner.DualModeMS is slightly different from the one used in G*e*MSS [18]. This is due to the fact that the size of a digest m considered here has to be bigger than in G*e*MSS. Thus, the iterative process of G*e*MSS is no longer required here. In fact, Inner.DualModeMS is equivalent to a G*e*MSS where the number of iterations is set to zero [18, Section 2].

2.1.4 Verification process

Let $\mathbf{d} \in \mathbb{F}_2^{n+v}$ and $\mathbf{s} \leftarrow \text{Inner.DualModeMS.Sign}\big(\mathbf{d}, \mathsf{sk} = (F, \mathbf{S}, \mathbf{T})\big) \in \mathbb{F}_2^{n+v}$. By construction, we have:

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

Thus, the verification of a signature (Algorithm 4) only requires to evaluate the public-key polynomials.

Algorithm 4 Verification process in Inner.DualModeMS

```
1: procedure Inner.DualModeMS.Verif($\mathbf{M} \in \{0,1\}^*, \mathbf{s} \in \mathbb{F}_2^{n+v}, \mathsf{pk} = \mathbf{p} \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^m$)
2:     $\mathbf{H} \leftarrow$ first $m$ bits of SHA3($\mathbf{M}$)
3:     $\mathbf{S}_0 \leftarrow \mathbf{p}(\mathbf{s}) \oplus \mathbf{d}$
4:     return VALID if $\mathbf{S}_0 = \mathbf{0}$ and INVALID otherwise.
5: end procedure
```

2.2 The SBP technique, Description of the outer layer and Specification of DualModeMS

2.2.1 Overview

The SBP technique [54], that we describe here, allows to transform the public-key origin.pk $= \mathbf{p} = (p_1, \ldots, p_m) \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^{n+v}$ of Inner.DualModeMS into a **new public-key** pk that is going to be the root of a binary tree (Algorithm 6). The **new secret-key** sk will include the original public-key of Inner.DualModeMS.

The **new signature** process (Algorithm 7) will require to generate signatures from Inner.DualModeMS. A (new) signature from DualModeMS will include random linear combinations $\mathbf{h} = (h_1, \ldots, h_\alpha) \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^\alpha$, with $1 \leq \alpha \leq m$, from the public-key origin.pk of Inner.DualModeMS together with a set of nodes allowing to check that $\mathbf{h}$ has been correctly derived from origin.pk (Algorithm 8).

2.2.2 Parameter space

The main parameters involved in the outer layer of DualModeMS are:

- N, number of square-free monomials in $n+v$ variables of degree ≤ 2
- σ, number of signatures of Inner.DualModeMS included in the final signature of DualModeMS
- α, number of polynomials included in a signature of DualModeMS
- $\tau \geq 1$, size of an evaluation set; must be a power of two
- k, degree of the extension field for MAC polynomials (Algorithm 5); must be such that $2^k \geq \tau$
- ν, number of Merkle paths to open

2.2.3 Secret-key and public-key

We describe the generation of the secret-key and the public-key in DualModeMS. This process uses the function Inner.DualModeMS.KeyGen that returns the public-key and secret-key of the inner multivariate scheme Inner.DualModeMS. That is $(\text{origin.sk}, \text{origin.pk}) \leftarrow$ Inner.DualModeMS.KeyGen(1^λ) with:

$$\begin{aligned} \text{origin.sk} &\leftarrow (F, \mathbf{S}, \mathbf{T}) \in \mathbb{F}_2[X, v_1, \ldots, v_v] \times \mathrm{GL}_{n+v}(\mathbb{F}_2) \times \mathrm{GL}_n(\mathbb{F}_2) \\ \text{origin.pk} &\leftarrow \mathbf{p} = (p_1, \ldots, p_m) \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^m \end{aligned}$$

The signature process in DualModeMS requires to generate signatures from Inner.DualModeMS. In addition, a signature from DualModeMS will also include a set of random linear combination

$$\mathbf{h} = (h_1, \ldots, h_\alpha) \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^\alpha \quad (7)$$

of Inner.DualModeMS's public-key. A key point in the constuction of [54], and so in DualModeMS, is a mechanism allowing to check that $\mathbf{h} \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^\alpha$ has been indeed correctly derived from origin.pk.

To do so, SBP introduced the concept of MAC polynomial. This is a univariate polynomial defined over $\mathbb{F}_{2^k}$ constructed from any multivariate polynomial (Algorithm 5).

Algorithm 5 Construction of the MAC polynomial

```
1: function MacPoly( $f \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]$)
2:     Coefficients ← Sorted list of all coefficients of $f$
3:     $j \leftarrow 1$
4:     for $i$ from 0 to $\lceil \frac{N}{k} \rceil - 1$ do
5:         $c_i \leftarrow$ Cast Coefficients$[j, \ldots, j + (k-1)]$ as an element of $\mathbb{F}_{2^k}$
6:         $j \leftarrow j + k$
7:     end for
8:     $\hat{f} \leftarrow \sum_{j=0}^{\lceil \frac{N}{k} \rceil - 1} c_j z^j$
9:     return $\tilde{f}$
10: end function
```

The MAC polynomials corresponding to origin.pk and $\mathbf{h}$ from (7) has to coincide. This is verified by evaluating the MAC polynomials on a set $Z \subseteq \mathbb{F}_{2^k}$ of τ points. This set is stored and compressed in a *Merkle tree* [45]. The function Merkle.generate_tree (Section 2.3.7, Algorithm 9) takes as input a set of τ points in $\mathbb{F}_{2^k}^m$ and construct the corresponding Merkle tree. The leafs of the Merkle tree are elements in $\mathbb{F}_{2^k}^m$ whilst the inner nodes are hash values of size λ.

We have now all the tools to describe the public-key/secret-key generation process in DualModeMS (Algorithm (1)).

The public-key is then just a hash value of λ bits. On the other hand, the secret is big since it will include – in particular – the public-key of Inner.DualModeMS.

Algorithm 6 PK/SK generation in DualModeMS

```
1: procedure DualModeMS.KeyGen($1^\lambda$)
2:     $(\mathsf{origin.sk}, \mathsf{origin.pk}) \leftarrow$ Inner.DualModeMS.KeyGen($1^\lambda$)
3:     $\mathsf{origin.pk} \leftarrow (p_1, \ldots, p_m) \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^m$
4:     for $i$ from 1 to $m$ do
5:         $\hat{p}_i \leftarrow \mathrm{MacPoly}(p_i)$
6:     end for
7:     $\hat{\mathbf{p}} \leftarrow (\hat{p}_1, \ldots, \hat{p}_m)$
8:     $Z \leftarrow$ Randomly choose a set of $\tau$ points $Z_1, \ldots, Z_\tau \in \mathbb{F}_{2^k}$.
9:     $\mathsf{mt} \leftarrow \mathrm{Merkle.generate\_tree}(\{\hat{\mathbf{p}}(z)_{z \in Z}\})$
10:    $\mathsf{pk} \leftarrow \mathrm{Merkle.root}(\mathsf{mt})$    ▷ The (new) public-key is the root of a Merkle tree
11:    $\mathsf{sk} \leftarrow (\mathsf{origin.sk}, \mathbf{p}, Z)$
12:    return $(\mathsf{sk}, \mathsf{pk})$
13: end procedure
```

2.2.4 Signing process

The signature process of DualModeMS is derived from Inner.DualModeMS. The novelty in DualModeMS.KeyGen is on the use of a Merkle tree for adding an *authentication tag*. In particular, the function Merkle.path takes as input the root of a Merkle tree and a leaf of this tree and return a list of nodes allowing to re-compute the public-key root from tree from the leaf (Algorithm 10, Section 2.3.9).

The signature process in DualModeMS is described in Algorithm 7.

Algorithm 7 Signing process in DualModeMS

```
1: procedure Verif.Sign($\mathbf{M} \in \{0,1\}^*, \mathsf{sk} = (\mathsf{origin.sk}, \mathsf{origin.pk}, Z) \in \mathbb{F}_2[X, v_1, \ldots, v_v] \times$
   $\mathrm{GL}_{n+v}(\mathbb{F}_2) \times \mathrm{GL}_n(\mathbb{F}_2) \times \mathbb{F}_2[x_1, \ldots, x_{n+v}]^m \times \mathbb{F}_{2^k}$)
2:     $\mathbf{p} \leftarrow \mathsf{origin.pk} = (p_1, \ldots, p_m) \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^m$
3:     for $i$ from 1 to $\sigma$ do
4:         $\mathbf{s}_i \leftarrow$ Inner.DualModeMS($\mathbf{M} \| i, \mathsf{sk}$)    ▷ Get $\sigma$ signatures from Inner.DualModeMS
5:     end for
6:     $\mathbf{t} \leftarrow$ Cast $m \times \alpha$ bits of $\mathtt{PRNG}(M \| s_1 \| \cdots \| s_\sigma)$ into a matrix $\mathcal{M}_{m \times \alpha}(\mathbb{F}_2)$
7:     $\mathbf{h} = (h_1, \ldots, h_\alpha) \leftarrow \mathbf{p} \times \mathbf{t} \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^\alpha$
8:     $i_1, \ldots, i_\nu \leftarrow$ Cast $\nu \times \log_2(\tau)$ bits of $\mathtt{PRNG}(M \| s_1 \| \cdots \| s_\sigma \| \mathbf{h})$ into a set $\nu$ integers, each
   $\in [1, \tau]$
9:     $O \leftarrow \{Z_{i_1}, \ldots, Z_{i_\nu}\} \subset \mathbb{F}_{2^k}$    ▷ A random subset of $Z \in \mathbb{F}_{2^k}$ of size $\nu$
10:    $\hat{\mathbf{p}} \leftarrow \big(\mathrm{MacPoly}(p_1), \ldots, \mathrm{MacPoly}(p_m)\big)$
11:    for $i$ from 1 to $\nu$ do
12:        $\mathtt{openpath}[i] \leftarrow \mathrm{Merkle.path}\big(\mathsf{mt}, \hat{\mathbf{p}}(O_i)\big)$
13:    end for
14:    $\mathtt{openpaths} \leftarrow [\mathtt{openpath}[i] \mid 1 \le i \le \tau]$
15:    Return $(\mathbf{s}_1, \ldots, \mathbf{s}_\sigma, \mathbf{h}, \mathtt{openpaths})$
16: end procedure
```

The size of a signature is then :

$$(n+v)\cdot\sigma+\alpha\cdot N+\nu\cdot m\cdot k+\nu(\log_2(\tau)-2)\lambda. \tag{8}$$

2.2.5 Verification process

The verification process in DualModeMS is described in Algorithm 8. We need to verify the validity of signatures from Inner.DualModeMS as well as the validity of polynomials included in a signature from VERIF.SIGN. To so so, we use the function Merkle.verify (Algorithm 11, Section 2.3.9) that takes as input the root of a Merkle tree and a set of nodes and verify that the root can be indeed generated from the nodes.

Algorithm 8 Verification process in DualModeMS

```
1: procedure VERIF.DualModeMS($\mathbf{M}\in\{0,1\}^*, \mathbf{s}\in\mathbb{F}_2^{\sigma\cdot m}\times\mathbb{F}_2[x_1,\ldots,x_{n+v}]^\alpha\times\{0,1\}^*$, pk $=$ mt $\in\{0,1\}^\lambda$)
2:     $(\mathbf{s}_1,\ldots,\mathbf{s}_\sigma,\mathbf{h},$ openpaths$) \leftarrow \mathbf{s}$
3:     $\mathbf{t}\leftarrow$ Cast $m\times\alpha$ bits of PRNG$(M\|s_1\|\cdots\|s_\sigma)$ in a matrix $\mathcal{M}_{m\times\alpha}(\mathbb{F}_2)$
4:     for $i$ from 1 to $\sigma$ do
5:         $\mathbf{H}_i\leftarrow$ first $m$ bits of SHA3$(\mathbf{M}\|i)$
6:         if $\mathbf{h}(\mathbf{s}_i)\neq\mathbf{H}_i\times\mathbf{t}$ then
7:             return INVALID
8:         end if
               ▷ We use the verification process of Inner.DualModeMS with $\mathbf{h}$ as a public-key
9:     end for
10:    $\hat{\mathbf{h}}\leftarrow\big(\mathrm{MacPoly}(\mathbf{h}_1),\ldots,\mathrm{MacPoly}(\mathbf{h}_\alpha)\big)$
11:    $i_1,\ldots,i_\nu\leftarrow$ Cast $\nu\times\log_2(\tau)$ bits of PRNG$(M\|s_1\|\cdots\|s_\sigma\|\mathbf{h})$ into a set $\nu$ integers $\in[1,\ldots,\tau]$
12:    $O\leftarrow\{Z_{i_1},\ldots,Z_{i_\nu}\}\subset\mathbb{F}_{2^k}$
13:    for $i$ from 1 to $\tau$ do
14:        $\hat{\mathbf{p}}(O_i)$, mp $\leftarrow$ openpaths$[i]$
15:        if Merkle.verify(pk, mp) $=$ False or $\hat{\mathbf{p}}(O_i)\neq\hat{\mathbf{h}}(O_i)\times\mathbf{t}$ then
16:            return INVALID
17:        end if
18:    end for
19:    return VALID
20: end procedure
```

We can now explain why G*e*MSS [18] can not be directly used in DualModeMS. In Step 6, we use the fact that:

$$\mathbf{h}(\mathbf{s}_i)=\mathbf{p}(\mathbf{s}_i)\times\mathbf{t}=\mathbf{H}_i\times\mathbf{t}.$$

Thus, we can use $\mathbf{h}$ instead of $\mathbf{p}$ in the verification process of Inner.DualModeMS. Due to the iterative process used in G*e*MSS, this trick is no longer possible. This is why we have to use a multivariate-based signature without any iterative signature process.

Due to this constraint, we have to consider $m = 2\lambda$ in order to avoid a simple birthday paradox.

2.3 Implementation

2.3.1 Generating invertible matrices

This part is similar to [18, Section 2.5.1] of the G*e*MSS submission.

2.3.2 Generating HFEv polynomials

Similar to [18, Section 2.5.2] of the G*e*MSS submission.

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

Similar to [18, Section 2.5.3] of the G*e*MSS submission.

2.3.4 Generating the components of a HFEv polynomial

Similar to [18, Section 2.5.4] of the G*e*MSS submission.

2.3.5 Generating the components of a HFEv polynomial

Similar to [18, Section 2.5.5] of the G*e*MSS submission.

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

Similar to [18, Section 2.5.6] of the G*e*MSS submission.

Algorithm 9 Generation of the Merkle tree

```
1: function Merkle.generate_tree($\{z_0, \ldots, z_{\tau-1}\} \subset \mathbb{F}_{2^k}^m$)
2:     for $i$ from 0 to $\tau - 1$ do
3:         $\mathsf{mt}[i] \leftarrow$ SHA3($z_i$)
4:     end for
5:     $j \leftarrow 0$
6:     for $i$ from $\tau$ to $2\tau - 2$ do    ▷ Each node is the SHA3 of the concatenation of these leaves
7:         $\mathsf{mt}[i] \leftarrow$ SHA3($\mathsf{mt}[j] \| \| mt[j+1]$)
8:         $j \leftarrow j + 2$
9:     end for
10: end function
```

Algorithm 10 Authentification path

```
1: function Merkle.path($\mathsf{mt}$, $\mathsf{leaf}$)
2:     $index \leftarrow$ index of the leaf in $mt$
3:     $fl \leftarrow 0$    ▷ Index of the begin of the $i$th floor
4:     for $i$ from 0 to $\log_2(\tau) - 1$ do    ▷ For each floor of the Merkle tree (excepted the root)
5:         $\mp[i] \leftarrow mt[fl + (index \,\mathbf{xor}\, 1)]$
6:         $fl + = \frac{\tau}{2^i}$    ▷ Add the size of the current floor
7:         $index / = 2$    ▷ Integer division
8:     end for
9:     return $\mp$
10: end function
```

Algorithm 11 Authentification path

```
1: function Merkle.verify($\mathsf{pk}, \mp, \hat{\mathbf{p}}(O_i)$)
2:     $H_{-1} \leftarrow$ SHA3($\hat{\mathbf{p}}(O_i)$)
3:     $index \leftarrow$ index of $H_{-1}$ in $mt$
4:     for $i$ from 0 to $\log_2(\tau) - 1$ do    ▷ For each floor of the Merkle tree (excepted the root)
5:         if ($index \mod 2) = 0$ then
6:             $H_i \leftarrow$ SHA3($H_{i-1} || \mp [i]$)
7:         else
8:             $H_i \leftarrow$ SHA3($mp[i] || H_{i-1}$)
9:         end if
10:        $index / = 2$    ▷ Integer division
11:    end for
12:    if $H_{\log_2(\tau)-1} == pk$ then
13:        return VALID
14:    else
15:        return INVALID
16:    end if
17: end function
```

2.3.7 Generation of the Merle tree

2.3.8 Computing the authenfication path

2.3.9 Verification of the authenfication path

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

3.1 Parameter set `sign/DualModeMS128`

We choose $\Delta = 10, v = 11$ and $m = 256$. This gives $n = 266, n + v = 277$ $D = 129$ and $K = 128$. The extension field is defined as $\mathbb{F}_{2^n} = \frac{\mathbb{F}_2[X]}{X^n+X^{47}+1}$. For Inner.DualModeMS, the public-key size is then 1139.06 KBytes and the signature size is 277 bits.

For the outer layer, we choose $\alpha = 2$, $\sigma = 64$, $\tau = 2^{18}$, $k = 21$, PKI$v = 18$ and $\delta = 4$. The extension field is defined as $\mathbb{F}_{2^k} = \frac{\mathbb{F}_2[X]}{X^k+X^2+1}$. For DualModeMS, this gives a public-key of 528 bytes and a signature of 32.002 KBytes.

3.2 Parameter set `sign/DualModeMS192`

We choose $\Delta = 18, v = 18$ and $m = 384$. This gives $n = 402, n + v = 420$, $D = 129$ and $K = 192$. The extension field is defined as $\mathbb{F}_{2^n} = \frac{\mathbb{F}_2[X]}{X^n+X^{171}+1}$. For Inner.DualModeMS, this gives a public-key of 4243.73 KBytes and a signature of 420 bits.

For the outer layer, we choose $\alpha = 2$, $\sigma = 96$, $\tau = 2^{18}$, $k = 20$, PKI$v = 33$ and $\delta = 5$. The extension field is defined as $\mathbb{F}_{2^k} = \frac{\mathbb{F}_2[X]}{X^k+X^3+1}$. For DualModeMS, this gives a public-key of 1560 bytes and a signature of 79.415 KBytes.

3.3 Parameter set `sign/DualModeMS256`

We choose $\Delta = 32, v = 32$ and $m = 512$. This gives $n = 544, n + v = 576, D = 129$ and $K = 256$. The extension field is defined as $\mathbb{F}_{2^n} = \frac{\mathbb{F}_2[X]}{X^n+X^8+X^3+X+1}$. For Inner.DualModeMS, this gives a public-key of 10635.32 KBytes and a signature of 576 bits.

For the outer layer, we choose $\alpha = 1$, $\sigma = 256$, $\tau = 2^{18}$, $k = 20$, PKI$v = 52$ and $\delta = 5$. The extension field is defined as $\mathbb{F}_{2^k} = \frac{\mathbb{F}_2[X]}{X^k+X^3+1}$. For DualModeMS, this gives a public-key of 2112 bytes and a signature of 149.029 KBytes.

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

The main design rationale of DualModeMS is to propose SBP [54] as a *mode of operations* for multivariate schemes. In order to demonstrate the drastic effect of SBP on public-key sizes, we tailored a specific *inner* multivariate-based scheme. Inner.DualModeMS is a HFEv- scheme [46, 41, 48] since [54] identified that such family is well suited in the context for SBP.

The design of Inner.DualModeMS follows from the analysis performed for G*e*MSS, a HFEv-based scheme, proposed in [18]. The SBP imposes to have a scheme without the iterative procedure proposed in G*e*MSS. It is then rather natural to take the number of equations m equals to the security parameter Inner.DualModeMS. We then use the methodology proposed in G*e*MSS [18, Section 8] to derive secure parameters. A signature of SBP requires to generate many signatures from Inner.DualModeMS. This leads toward the choice of a small $D = 129$ to make the signature process of Inner.DualModeMS efficient. We detail these choices in Section 8.

5 Detailed performance analysis (2.B.2)

We consider the parameters of DualModeMS128.

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

For the optimized implementation, the DualModeMS.KeyGen takes 797 seconds, the time to sign is 2.31 seconds, and the verification takes 2.69 ms.

For the additional implementation, the DualModeMS.KeyGen takes 552 seconds, the time to sign is 2.05 seconds, and the verification takes 2.84 ms.

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 theoretical sizes.

`DualModeMS`:
Public key is 528 bytes. Secret key is 18038184 bytes. Signatures are 32640 bytes.

5.5 Optimizations

The optimizations used for Inner.DualModeMS are similar to the ones used by G*e*MSS [18, Section 5.6].

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

We review in this part known results on the provable security of DualModeMS in the sense of the existential unforgeability against adaptive chosen-message attack (EUF-CMA security). The `SBP` technique [54] allows to reduce, for a certain range of parameters, the security of

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

DualModeMS to the one of Inner.DualModeMS. We then first discuss the EUF-CMA security (Section 6.1) of Inner.DualModeMS and then continue with the (EUF-CMA) security (Section 6.2) of Inner.DualModeMS.

6.1 EUF-CMA security of Inner.DualModeMS

This part is essentially similar to [18, Section 6.2] of the G*e*MSS submission. However, there is a slight difference between Inner.DualModeMS and G*e*MSS. Inner.DualModeMS is a version of G*e*MSS without any iteration.

EUF-CMA security of HFEv-, over which Inner.DualModeMS is designed, has been mainly investigated in [53]. The authors demonstrated that a minor, but costly, modification of Inner.DualModeMS.Sign (Algorithm 3) permits to achieve EUF-CMA security for Inner.DualModeMS.

We first formalize the security of Inner.DualModeMS against chosen message attacks.

Definition 1 ([53]). *The* Inner.DualModeMS *signature scheme*

$$(\text{Inner.DualModeMS.KeyGen}, \text{Inner.DualModeMS.Sign}, \text{Inner.DualModeMS.Verif})$$

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* $(\cdot, \mathsf{pk}_{\text{Inner.DualModeMS}}) \leftarrow \text{Inner.DualModeMS.KeyGen}()$ *and outputs a valid signature after* $t(\lambda)$ *steps with a probability at least* $\epsilon(\lambda)$ *with at most* $q_h(\lambda)$ *queries to the random oracle and* $q_s(\lambda)$ *queries to the signature oracle,*

We want to provably reduce EUF-CMA security of Inner.DualModeMS to the the hardness of inverting the public-key of Inner.DualModeMS. Formally:

Definition 2 ([53]). *We shall say that the* Inner.DualModeMS *function generator* Inner.DualModeMS.KeyGen *is* $\big(\epsilon(\lambda), t(\lambda)\big)$ *secure, if there is no inverting algorithm that takes* $\mathsf{pk}_{\text{Inner.DualModeMS}} = \mathbf{p}_{\text{Inner.DualModeMS}}$ *generated via* $(\cdot, \mathsf{pk}_{\text{Inner.DualModeMS}}) \leftarrow$ $\text{Inner.DualModeMS.KeyGen}(1^\lambda)$*, 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*

$$\mathbf{p}_{\text{Inner.DualModeMS}}(\mathbf{s}) = \mathbf{d}.$$

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

Following [53], we explain now how to modify Inner.DualModeMS for proving EUF-CMA security. Recall that D is degree of the secret polynomial with HFEv-shape in Inner.DualModeMS. The main modification proposed by [53] is roughly to repeat D times the signature process described in Algorithm 3.

Let ℓ be the length of a random salt. The modified inversion process is given in Algorithm 12:

We then have:

Algorithm 12 Modified inversion for Inner.DualModeMS

```
1: procedure Inner.DualModeMS.Sign*($\mathbf{M} \in \{0,1\}^*, \mathsf{sk} \in \mathbb{F}_2[X, v_1, \ldots, v_v] \times \mathrm{GL}_{n+v}(\mathbb{F}_2) \times \mathrm{GL}_n(\mathbb{F}_2)$)
2:     $\mathbf{d} \leftarrow$ first $m$ bits of SHA3($\mathbf{M}$)
3:     $\mathbf{v} \in_R \mathbb{F}_2^v$
4:     repeat
5:         $\mathbf{salt} \in_R \{0,1\}^\ell$
6:         $\mathbf{r} \leftarrow$ first $n-m$ bits of SHA3($\mathbf{d}||\mathbf{salt}$)
7:         $\mathbf{d}' \leftarrow (\mathbf{d}, \mathbf{r}) \in \mathbb{F}_2^n$
8:         $D' \leftarrow \varphi^{-1}(\mathbf{d}' \times \mathbf{T}^{-1}) \in \mathbb{F}_{2^n}$
9:         $F_{D'}(X) \leftarrow F(X, \mathbf{v}) - D'$
10:        $(\cdot, \mathrm{Roots}) \leftarrow \mathrm{FindRoots}(F_{D'})$
11:        $u \in_R \{1, \ldots, D\}$
12:    until $1 \leq u \leq \#\mathrm{Roots}$
13:    $Z \in_R \mathrm{Roots}$
14:    return $(\varphi(Z), \mathbf{v}) \times \mathbf{S}^{-1} \in \mathbb{F}_2^{n+v}$
15: end procedure
```

Theorem 2 ([53]). *Let* Inner.DualModeMS* *be the signature scheme defined by* (Inner.DualModeMS.KeyGen, Inner.DualModeMS.Sign*, Inner.DualModeMS.Verif). *If the* Inner.DualModeMS *function generator* Inner.DualModeMS.KeyGen *is* (ϵ', t') *secure, then* Inner.DualModeMS* *is* (ϵ, t, q_H, q_S) *secure, with:*

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

where $t_{\mathsf{Inner.DualModeMS}}$ *is the time required to evaluate the public-key of* Inner.DualModeMS.

The modification of the signature process renders Inner.DualModeMS* less efficient than Inner.DualModeMS. The expected number of calls to the roots finding (Step 10) in Inner.DualModeMS.Sign* is $\frac{1}{1-1/e} D \approx 1.58 \times D$. In Inner.DualModeMS.Sign, the average number of calls to the roots finding (Step 9) is $\frac{1}{1-1/e} \approx 1.58$. For efficiency reasons, we did not incorporated this modification in our implementation.

Remark 3. *The threshold D in Step 11 corresponds to a bound on the number of roots of the univariate polynomial F at Step 10. 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 [35], 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 [53], the threshold D in Step 11 can be theoretically much decreased without compromising the proof. The authors of [53] mentioned a value around* ≈ 30 *for the threshold.*

However, we emphasize that the provable security result mentioned up to know only require minor modifications of the signature process. We don't need to change the underlying trapdoor. As a consequence, the security of Inner.DualModeMS has to be mainly investigated with respect to the hardness of inverting the public-key. This question is investigated in Section 8.

6.2 EUF-CMA security of DualModeMS

We consider now the EUF-CMA security of DualModeMS. The next fundamental theorem is directly derived from [54].

Theorem 3 (SBP, [54])**.** *If there exists a forger* A *that breaks the* EUF-CMA *security against* DualModeMS *in time t with q random oracle queries and with success probability ϵ, then there is an adversary B^{A} than breaks* EUF-CMA *security of* Inner.DualModeMS *in time $O(t)$ and with success probability al least:*

$$\epsilon - (q+1)\left(\frac{\left\lceil \frac{N}{k} \right\rceil - 1}{\tau}\right)^{\nu} - 2\tau\frac{(q+1)}{2^{\lambda}} - (q+1)\frac{1}{2^{\alpha}}. \tag{9}$$

Theorem 3 provides a guidance for choosing the various parameters involved in DualModeMS. The term $\left(\frac{\left\lceil \frac{N}{k} \right\rceil - 1}{\tau}\right)^{\nu}$ in (9) is the probability that an invalid set of polynomials $\mathbf{h}' \in\in \mathbb{F}_2[x_1,\ldots,x_{n+v}]^{\alpha}$ passes the test of VERIF.DualModeMS (Algorithm 8). Thus, we need to choose the parameters such that :

$$\left(\frac{\left\lceil \frac{N}{k} \right\rceil - 1}{\tau}\right)^{\nu} < \frac{1}{2^{\lambda}}.$$

When $\alpha = O(1)$, Theorem 3 is meaningless. However, this is the situation that we will consider for DualModeMS. Indeed, α denotes the number of polynomials included in the signature. Thus, taking small α allows to obtain smaller signatures.

In this situation, we have however no security reduction [54]. The security of the SBP transform relies then on a new hard problem, so called *Approximate Multivariate Quadratic* (AMQ) problem, that is defined below:
Input. a set of polynomial $\mathbf{p} = (p_1,\ldots,p_m) \in \mathbb{F}_2[x_1,\ldots,x_{n+v}]^m$, σ vectors $\mathbf{y}_1,\ldots,\mathbf{y}_\sigma \in \mathbb{F}_2^m$ and $r > 0$.
Question. Find $\mathbf{x}_1,\ldots,\mathbf{x}_\sigma \in \mathbb{F}_2^{n+v}$ such that

$$\dim\Bigg(\mathrm{Vec}\big(\mathbf{p}(\mathbf{x}_1) - \mathbf{y}_1,\ldots,\mathbf{p}(\mathbf{x}_\sigma) - \mathbf{y}_\sigma\big)\Bigg) \leq r.$$

According to [54], it is necessary to have AMQ hard for small r, i.e:

$$r < \frac{\lambda}{\alpha}.$$

This is discussed in Section in Section 8.

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

7.1 Parameter set `sign/DualModeMS128`

Category 1.

7.2 Parameter set `sign/DualModeMS192`

Category 3.

7.3 Parameter set `sign/DualModeMS256`

Category 5.

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

This part provides a summary of the main attacks against DualModeMS. In Section 8.1, we consider direct signature forgery attacks Inner.DualModeMS. 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 against Inner.DualModeMS.

8.1 Direct signature forgery attacks

The public-key of Inner.DualModeMS 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:

$$p_1(x_1, \ldots, x_{n+v}) - d_1 = 0, \ldots, p_m(x_1, \ldots, x_{n+v}) - d_m = 0, x_1^2 - x_1, \ldots, x_{n+v}^2 - x_{n+v} = 0. \quad (10)$$

Stated differently, the task is to invert Inner.DualModeMS.Sign (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 (10) 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 Inner.DualModeMS.Sign (Steps 4 and 7 of Algorithm 3).

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

$$p_1(x_1,\ldots,x_m,\mathbf{r})-d_1=0,\ldots,p_m(x_1,\ldots,x_m,\mathbf{r})-d_m=0,x_1^2-x_1,\ldots,x_m^2-x_m=0. \quad (11)$$

8.1.1 Exhaustive search

In [10], 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 (11), the approach from [10] requires:

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

For the parameters of Inner.DualModeMS, we have:

m	Fast exhaustive search ([10])
256	2^{261}
384	$2^{389.1}$
512	$2^{517.16}$

8.1.2 Quantum exhaustive search

In [16], the authors proposed simple quantum algorithms for solving systems of quadratic boolean equations. The principle of [16] is to perform a fast quantum exhaustive search by using Grover's algorithm. [16] 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 (11) for the parameters of Inner.DualModeMS. The quantum attacks from [16] require then:

m	#qbits	#quantum gates
256	516	$2^{153.52}$
256	268	$\approx 2^{154.52}$
384	772	$2^{219.27}$
384	396	$\approx 2^{220.27}$
512	1028	$2^{284.51}$
512	525	$\approx 2^{289.51}$

8.2 Approximation algorithm

Recently, the authors of [43] proposed a new algorithm for solving systems of non linear equations that is faster than a direct exhaustive search. The techniques from [43] allow for the approximation of a non-linear system, as (11), 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 (11) 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 Inner.DualModeMS, we have then:

m	Lower bound on the complexity of [43]
256	$2^{224.38}$
384	$2^{336.57}$
512	$2^{448.76}$

8.3 Gröbner bases

To date, the best methods for solving non-linear equations, including the attack system (11), utilize Gröbner bases [14, 13]. The historical method for computing such bases – known as Buchberger's algorithm – has been introduced by Buchberger in his PhD thesis [14, 13]. 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 [28, 29]. The F4 algorithm, for example, is the default algorithm for computing Gröbner bases in the computer algebra software Magma [9]. The F5 algorithm, which is available through the FGb [31] 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` [12] which is a general framework to compute Gröbner basis in $\mathbb{F}_2[x_1, \ldots, x_n]/\langle x_i^2 - x_i\rangle_{1\leq i\leq n}$. 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 proposed in cryptography is the `XL` algorithm [20]. 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` [37], `GVW` [38], ..., flourished building on the core ideas of F4 and F5. This literature is vast and we refer to [27] for a recent survey of these algorithms.

[5] http://www-polsys.lip6.fr/~jcf/FGb/index.html
[6] http://polybori.sourceforge.net

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 [50] 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 Inner.DualModeMS against `BooleanSolve` (probabilistic version) for the three parameters proposed.

m	Lower bound on the cost of `BooleanSolve` ($2^{0.792 \cdot m}$)
256	$2^{202.75}$
384	$2^{304.12}$
512	$2^{405.50}$

Table 3: Security of Inner.DualModeMS against `BooleanSolve`.

`QuantumBooleanSolve`. In a recent paper [33], the authors present a quantum version of `BooleanSolve` that takes advantages of Grover's quantum algorithm [40]. `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 Inner.DualModeMS against `QuantumBooleanSolve` (probabilistic version) for the three parameters proposed.

8.3.2 Practically fast algorithms

The direct attack described in [30, 34] provides reference tools for evaluating the security of `HFE` and `HFEv-` against a direct message-recovery attack. This attack uses the F5 algorithm

m	Lower bound on the # quantum gates for QuantumBooleanSolve ($2^{0.462 \cdot m}$)
256	$2^{118.27}$
384	$2^{177.4}$
512	$2^{236.54}$

Table 4: Security of Inner.DualModeMS against QuantumBooleanSolve.

[29, 4] and has a complexity of the following general form:

$$O\big(\mathrm{poly}(m,n)^{\omega \cdot D_{\mathrm{reg}}}\big), \tag{12}$$

with $2 \leq \omega < 3$ being the so-called *linear algebra constant* [56], 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 $\omega < 2.3728639$ [36]. In this part, we will always use $\omega = 2$ to evaluate the cost of Gröbner bases attacks.

The complexity (12) 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 (11) has been experimentally shown to be smaller than $\log_2(D)$ [30, 34]. This behavior has been further demonstrated in [39, 26].

In particular, [39] claims that the degree of regularity reached in HFE is asymptotically upper bounded by:

$$(2+\epsilon)(1-\sqrt{3/4})\cdot\min\bigl(m,\log_2(D)\bigr),\ \text{for all } \epsilon>0. \tag{13}$$

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 (13) is not necessarily tight for specified values of the parameters. Thus, (13) 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 [34], we lower bound the complexity of F5 against HFE, i.e. for solving the attack system (11). 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:

$$O\left(\binom{m}{D_{\text{reg}}}^2\right). \tag{14}$$

This is a conservative estimate on the cost of solving (11). 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 (14) 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.

m	minimal D_{reg} required	Lower bound on the cost of a Gröbner basis as given in (14)
256	12	$2^{133.57}$
384	17	$2^{194.17}$
512	22	$2^{263.64}$

Table 7: Smallest degree of regularity required.

In particular, the practical effect of the minus and vinegar modifiers have been considered in [30, 34]. This has been further investigated in [21, 24] who presented a theoretical upper bound on the degree 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

$$\frac{R+v+\Delta-1}{2}+2, \quad \text{when } R+\Delta \text{ is odd,} \tag{15}$$

$$\frac{R+v+\Delta}{2}+2, \quad \text{otherwise.} \tag{16}$$

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, [50] derived an experimental *lower bound* on the degree of regularity in `HFEv-`. The authors [50] obtained that the degree of regularity for `HFEv-` should be at least :

$$\left\lceil \frac{R+\Delta+v+7}{3} \right\rceil. \tag{17}$$

8.3.3 Experimental results for `HFEv-`

The main question in the design of Inner.DualModeMS 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 (11). We mention that similar experiments were performed in [51].

We first consider $v = 0$, and denote by Δ the number of equation removed, i.e. $m = n - r$. According to the upper bounds (15) and (16), 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 [51] where the authors

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.

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.

We also performed experimental results with a combination of vinegar and minus. Similarly

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.

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.

to [51], we observed that the behaviour obtained seems similar for HFEv- with $\Delta = 0$ and v vinegar 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* [42] (Section 8.4.1) and differential attacks (Section 8.4.2).

8.4.1 Kipnis-Shamir attack

In [42], 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 [15, 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:

$$\sum_{\substack{0\leqslant i<j<n \\ 2^i+2^j\leqslant D}} A_{i,j}\, X^{2^i+2^j} \in \mathbb{F}_{2^n}[X], \text{ with } A_{i,j} \in \mathbb{F}_{2^n}. \tag{18}$$

We can then write (18) 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 [42, 7].

In [7], the authors evaluated the cost of the Kipnis-Shamir key-recovery attack with the best known tools for solving the MinRank [32] instance that occurs in HFE. Following [7], the cost of the Kipnis-Shamir attack against HFE can be estimated to:

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

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 [42, 7] to HFE- when $n - m \geq 2$. In [55], 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 [51]) 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:

$$O\left(n^{\omega(\lceil \log_2(D)\rceil + v + \Delta + 1)}\right), \qquad (19)$$

where D is the degree of the secret univariate polynomial.

For all the parameters proposed for scheme, assuming $\omega = 2$, the cost (19) is always much bigger than the cost of the best direct attack (Section 8.1).

Remark 4. *Recently, [22] 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* Inner.DualModeMS *below the security parameter.*

8.4.2 Differential attack

We finally consider so-called *differential attacks*, introduced [25], are structural attacks that can be used to attack multivariate cryptosystems. Differential attacks turned to be very efficient, e.g. [25, 11] against `SFLASH` [47]; a popular multivariate-based signature based on the Mastsumoto and Imai [44].

`HFE` is the successor, and a generalization, of [44]. 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 [44] variants such as `SFLASH` [47]. In [17], the authors proved that variants of `HFE`, such as Inner.DualModeMS, are immune against known differential attacks.

8.5 `AMQ`

For the parameters chosen, we need to assume the hardness of `AMQ` for

$$r < \frac{\lambda}{\alpha}.$$

For our parameters, $\alpha = 1$ or 2. Precisely, r is bounded by 64, 96 and 256 for the 3 parameters proposed respectively in Section 8.6. In fact, we have chosen the parameters, so that $m - r = \lambda$ so that the best attack proposed has complexity 2^λ. It can be also mentioned that `AMQ` is related to the so-called `Generalized MinRank Problem` [32]. Given a matrix whose coefficients are multivariate polynomials, the goal is to find a assignment of the variables that makes the rank of the matrix smaller than a given rank. Thus, we have a problem which is in some sense harder than the Kipnis-Shamir attack described Section 8.4.

`AMQ` is `Generalized MinRank Problem` with $n + v$ variables, polynomials of degree 2 and matrix of size $m \times \nu$. [32] provides then the degree of regularity for solving the corresponding determinantal system.

8.6 Deriving number of variables for Inner.DualModeMS

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 Inner.DualModeMS, we consider a secret univariate polynomial of degree $D = 129$. This is less than the degree used in G*e*MSS [18] that considers $D = 512$. We make this choice because we need to repeat many times the signature process Inner.DualModeMS.Sign in DualModeMS.

This $D = 129$ corresponds to a degree of regularity of 5 for a nude `HFE`, i.e. without any modifier. We consider that 3 modifiers allow to increase the degree of regularity by one.

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 (5 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	$v + \Delta$
Inner.DualModeMS128	256	$12 - 5 = 7$	21
Inner.DualModeMS192	384	$17 - 5 = 12$	36
Inner.DualModeMS256	512	$22 - 5 = 22$	64

Table 13: Numbers of modifiers required in Inner.DualModeMS.

9 Advantages and limitations (2.B.6)

Will be addressed in the full version.

The construction allows to greatly decreases the size of Inner.DualModeMS. However, the secret is much bigger and the time to sign verify. However, this provides very interesting tradeoffs for multivariate schemes.

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] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] A. Casanova, J.-C. Faugère, Orange G. Macario-Rat, J. Patarin, L. Perret, and J. Ryckeghem. G*e*MSS : A gr*e*at multivariate short signature. Submission to NIST.

[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, 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.

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

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

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

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

[29] 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.

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

[31] 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.

[32] 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.

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

[34] 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.

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

[36] 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.

[37] 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.

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

[39] 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.

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

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] Ralph C. Merkle. A certified digital signature. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer, 1989.

[46] 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.

[47] 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.

[48] 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.

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

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

[51] 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.

[52] 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.

[53] 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.

[54] Alan Szepieniec, Ward Beullens, and Bart Preneel. MQ signatures for PKI. In *PQCrypto*, volume 10346 of *Lecture Notes in Computer Science*, pages 224–240. Springer, 2017.

[55] 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.

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

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