Improving Support-Minors rank attacks: applications to G*e*MSS and Rainbow

John Baena[1], Pierre Briaud[2,3], Daniel Cabarcas[1], Ray Perlner[4], Daniel Smith-Tone[4,5] and Javier Verbel[6]

[1] Universidad Nacional de Colombia, Colombia
[2] Sorbonne Universités, UPMC Univ Paris 06
[3] Inria, Team COSMIQ, Paris, France
pierre.briaud@inria.fr
[4] National Institute of Standards and Technology, USA
[5] University of Louisville, USA
[6] Cryptography Research Centre, Technology Innovation Institute, Abu Dhabi, UAE

Abstract. The Support-Minors (SM) method has opened new routes to attack multivariate schemes with rank properties that were previously impossible to exploit, as shown by the recent attacks of [35] and [7] on the NIST candidates G*e*MSS and Rainbow respectively. In this paper, we study this SM approach more in depth, which allows us first to propose a greatly improved attack on G*e*MSS and also to define a more realistic cost model to evaluate the memory complexity of an XL strategy on the SM system using the Block-Wiedemann algorithm. Our new attack on G*e*MSS makes it completely unfeasible to repair the scheme by simply increasing the size of its parameters or even applying the projection technique from [31], as the signing time would be increased in a considerable way. Also, in our refined cost model, the rectangular MinRank attack from [7] does indeed reduce the security of all Round 3 Rainbow parameter sets below their targeted security strengths.

Keywords: Support-Minors, G*e*MSS, Rainbow, multivariate cryptography

1 Introduction

The MinRank problem 1 introduced in [9] has shown to be essential in establishing the security of several post-quantum cryptosystems, in particular multivariate schemes (MPKCs). Many MPKCs are indeed either directly based on the hardness of MinRank [16] or strongly related to it, such as [32,34,19].

Problem 1 (MinRank problem) *Given $d \in \mathbb{N}$, N matrices $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_N \in \mathbb{F}_q^{n_r \times n_c}$, find field elements $x_1, x_2, \ldots, x_N \in \mathbb{F}_q$, not all zero, such that*

$$\mathit{rank}\left(\sum_{i=1}^{N} x_i \boldsymbol{M}_i\right) \leq d.$$

The currently most high profile application of MinRank is the cryptanalysis of Rainbow [18], which was selected as a finalist to the NIST post-quantum standardization process. Rainbow is a multilayer variant of the well-known UOV signature scheme, and a key-recovery attack on the scheme can be performed by solving one of several particular MinRank instances [8,24,7]. This problem also shows up in the analysis of other types of MPKCs, namely those relying on the so-called *big-field* construction by using a field extension $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$. This is the case of the historical proposals C* [29] and HFE [32], but also more recently of the HFEv- schemes [33] GeMSS [10] and Gui [17]. In this context, a difference with the original formulation from Problem 1 is that the coefficients x_i's or the entries of the $\boldsymbol{M}_i$'s may belong to the extension field $\mathbb{F}_{q^n}$.

Support-Minors is a method proposed by Bardet et al. [3] to reduce the MinRank problem to the problem of solving a system of bilinear equations. This algebraic modeling is in particular the crux of the recent attacks on MPKCs and rank-based cryptosystems [3,7,2,35]. When the corresponding MinRank instance has a unique solution, which was the case in rank-based cryptography or Rainbow [3,7], this system can be solved using a variant of the XL algorithm [15]. In particular, this approach benefits from the extreme sparsity of the resulting linear system as one can use the Block-Wiedemann algorithm [14]. However, the situation is quite different for big-field schemes, since there are naturally n solutions coming from the big-field structure. In particular, using the XL algorithm proposed in [3] neither directly yields a solution nor reduces the problem to a simpler one. Of course, it is still possible to use a general purpose Gröbner basis algorithm, but this approach can be inefficient and one faces the challenging task of establishing the solving degree to precisely estimate its complexity. In particular, the authors of [35] conjectured from experiments that the *first degree fall* d_{ff} of their Support-Minors attack on GeMSS was equal to 3. Then, based on the common heuristic that the solving degree is close to d_{ff}, they derive the complexity given in Column "support minors modeling" from [35, Table 1]. However, one may wonder if such a small value for d_{ff} is not only due to the small scale of their experiments. Moreover, the assumption that d_{ff} coincides with the solving degree remains a conjecture. It is known this is not true in general, see for example [4], and if this solving degree were higher than 3 in the case of GeMSS, the complexity of the attack in [35] would dramatically change.

Also, even when there is justification for the time complexity of an attack, there remains the question of how to measure the complexity of memory intensive cryptanalytic attacks, an issue which has been a major point of discussion throughout the NIST PQC competition. In an effort to obtain more efficient parameters while still claiming high security, a number of submitters [5,1,12,28] have introduced cost models which treat memory intensive attacks as being more expensive than indicated by time complexity estimates using the more common Random Access Machine model. The question of the effect of memory access on the cost of MinRank attacks in particular, has been brought to the fore recently. In response to the rectangular MinRank attack [7], the Rainbow team put forward a statement [36] arguing that even though this attack reduces the

security of Rainbow relative to prior cryptanalysis, it does not bring any of the third round Rainbow parameters below their targeted security levels if memory costs are properly accounted for. This argument in particular states that, although the rectangular MinRank attack can use the Wiedemann algorithm and therefore does not require as much memory as attacks requiring Gröbner basis algorithms like F4 [22] and F5 [23], its complexity is dominated by a large number of random access queries to a memory, which is nonetheless fairly large.

Table 1. Time complexity of our attack (Improved SM, $\log_2(\#\text{gates})$) in comparison to [35]. pb: do we have to define "gates" ?

Scheme	Minors [35]	SM (conjectural) [35]	Improved SM
GeMSS128	139	118	72
BlueGeMSS128	119	99	65
RedGeMSS128	86	72	49
GeMSS192	154	120	75
BlueGeMSS192	132	101	67
RedGeMSS192	95	75	51
GeMSS256	166	121	75
BlueGeMSS256	141	103	68
RedGeMSS256	101	76	52

Contributions. As a first contribution, we provide solid ground to understand the Gröbner basis computation on the Support-Minors system for HFEv- and we significantly speed up the attack in [35] which used minors modeling [21]. We provide a necessary and sufficient condition for solving the SM system at degree 2, under mild assumptions. In the case of GeMSS, we show that it can always be solved at degree 2. This material allows us to give a precise complexity formula for the Support-Minors attack on GeMSS, which is also considerably smaller than the conjectured one in [35], which relied on the aforementioned degree fall assumption (see Column "SM (conjectural) [35]" in Table 1). In particular, with our attack we can also clearly break the proposed parameters for pHFEv-, which were an attempt by [31] to repair GeMSS in the aftermath of the attacks from [35]. Also, it makes it completely unfeasible to repair GeMSS by simply increasing the size of its parameters or even applying the projection technique without becoming impractical. These improvements come from some technical observations which are described more thoroughly throughout the paper. We show that by direct linearization on the Support-Minors equations, one can already obtain linear equations. Then, one can derive a quadratic system in only $n-1$ variables by substitution of these linear polynomials in the original system, and our attack is by solving this second system. For the sake of completeness, we also provide an estimate for the memory complexity. All in all, since the time complexity of

our attack on GeMSS is much reduced, the memory access cost also remains limited and it is not an obstacle to perform the attack.

As a second major contribution, we propose and analyse a strategy to obviate much of the memory access cost in implementing the Wiedemann algorithm as a subroutine for XL. We exemplify the strategy on the rectangular MinRank attack on Rainbow [7] and we determine the cost on average of memory accesses in this case. While the memory access cost of the Wiedemann algorithm when applied to a Macaulay matrix of size V and row weight w over $\mathbb{F}_q$ was estimated by [36] to require remotely accessing $3wV^2 \log_2 V$ bits within a memory of size V, we conjecture that by organizing memory locally, this figure can be reduced to the equivalent of $3V^2 \log_2 q$ bits worth of remote access to memory, saving a factor of $w \log_2 V / \log_2 q$. We also provide a concrete strategy for coming close to this figure, assuming the cost of memory access scales with either the square root or the cube root of the size of memory. Our concrete analysis shows that, even assuming the same cost for remote memory access as [36], the memory access cost can be reduced by a factor ranging from 2^{12} to 2^{16} relative to the costs estimated there, and the rectangular MinRank attack does indeed reduce the security of all round 3 Rainbow parameter sets below their targeted security strengths.

Along with this paper, we also provide a SageMath notebook [39], where the reader may verify our results for the GeMSS attack.

2 Preliminaries

2.1 Notation

Row vectors and matrices will be written in **bold**. We denote by v_i the i-th component of a vector $\boldsymbol{v}$, and the entries of a matrix $\boldsymbol{M}$ of size $n_r \times n_c$ will be denoted by $\boldsymbol{M}_{i,j}$, where i (resp. j) is an integer in $\{1..n_r\}$ (resp. $\{1..n_c\}$). The support $\mathrm{Supp}(\boldsymbol{v}) := \{i \mid v_i \neq 0\}$ of a vector $\boldsymbol{v}$ is the set of indices of its non-zero coordinates. For $I \subset \{1..n_r\}$ and $J \subset \{1..n_c\}$, we use the notation $\boldsymbol{M}_{I,J}$ for the submatrix of $\boldsymbol{M}$ formed by its rows (resp. columns) with indexes in I (resp. J), and we adopt the shorthand notation $\boldsymbol{M}_{*,J} = \boldsymbol{M}_{\{1..n_r\},J}$ and $\boldsymbol{M}_{I,*} = \boldsymbol{M}_{I,\{1..n_c\}}$. We use $\#I$ to denote the number of elements of the set I.

A field with q elements is denoted by $\mathbb{F}_q$. The big field schemes take their name from a field extension $\mathbb{F}_{q^n}$ of degree n over $\mathbb{F}_q$, and in the following we consider ϕ an isomorphism $\mathbb{F}_{q^n} \to \mathbb{F}_q^n$ between vector spaces. For $j \in \mathbb{Z}_{\geq 0}$ and $\boldsymbol{v} = (v_1, \ldots, v_k) \in \mathbb{F}_{q^n}^k$, we define

$$\boldsymbol{v}^{[j]} := (v_1^{q^j}, \ldots, v_k^{q^j}).$$

This corresponds to applying the Frobenius automorphism $x \mapsto x^q$ j times on each coordinate of $\boldsymbol{v}$. Note that this field automorphism is the identity on $\mathbb{F}_q$. We will adopt the same notation for matrices, namely the matrix $\boldsymbol{M}^{[j]}$ is the matrix obtained from $\boldsymbol{M}$ by raising all its entries to the power q^j.

Polynomial Systems and Coding Theory. We use $\boldsymbol{x} = (x_1, \ldots, x_N)$ to denote a vector of variables, and $\mathbb{F}_q[\boldsymbol{x}]$ denotes the ring of polynomials in the variables $\boldsymbol{x}$ and coefficients in $\mathbb{F}_q$. When q is an odd prime power, and g a quadratic form in $\mathbb{F}_q[\boldsymbol{x}]$, we denote by $\boldsymbol{G}$ the symmetric matrix defined by $g(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{G}\boldsymbol{x}^\mathsf{T}$ and g' the polar form associated to g. The evaluation of a polynomial system $\mathcal{P} = (p_1, \ldots, p_m)$ at $\boldsymbol{s} \in \mathbb{F}_q^n$ is the vector $\mathcal{P}(\boldsymbol{s}) := (p_1(\boldsymbol{s}), \ldots, p_m(\boldsymbol{s}))$, and we denote by $\mathcal{P}^h = (p_1^h, \ldots, p_m^h)$ the homogeneous sequence such that p_i^h is the homogeneous part of highest degree in p_i for $1 \leq i \leq m$. We also consider the Macaulay matrix $\boldsymbol{M}(\mathcal{P}) \in \mathbb{F}_q^{m \times n_\mathcal{M}}$ whose columns are indexed by the monomials in $\mathcal{P}$ and such that the entries in the i-th row correspond to the coefficients of p_i for $1 \leq i \leq m$. If this matrix is full rank, then the rowspace is an m-dimensional $\mathbb{F}_q$-subspace of $\mathbb{F}_q^{n_\mathcal{M}}$ which can be viewed as a linear code $\mathcal{M}$ of parameters $[n_\mathcal{M}, m]_q$. A *generating matrix* is precisely given by $\boldsymbol{M}(\mathcal{P})$, and the *dual* is the $[n_\mathcal{M}, n_\mathcal{M} - m]_q$-linear code $\mathcal{M}^\perp$ defined by

$$\mathcal{M}^\perp := \left\{ \boldsymbol{h} \in \mathbb{F}_q^{n_\mathcal{M}} \mid \forall \boldsymbol{c} \in \mathcal{M},\ \boldsymbol{c}\boldsymbol{h}^\mathsf{T} = 0 \right\},$$

which coincides with the right kernel of this matrix. Finally, the *puncturing* and *shortening* operations are classical ways to construct new linear codes from existing ones, and we use them in Section 5.1.

Definition 1 (Punctured code). *Let $\mathcal{C} \subset \mathbb{F}_q^n$ be a code of parameters $[n, K]_q$ and let $I \subset \{1..n\}$. The puncturing $\mathcal{P}_I(\mathcal{C}) \subset \mathbb{F}_q^{n-\#I}$ of $\mathcal{C}$ at I is the $[n-\#I, K' \leq K]_q$-code defined by:*

$$\mathcal{P}_I(\mathcal{C}) := \left\{ \boldsymbol{c}_{\{1..n\}\setminus I} \mid \boldsymbol{c} \in \mathcal{C} \right\}.$$

Definition 2 (Shortened code). *Let $\mathcal{C} \subset \mathbb{F}_q^n$ be a code of parameters $[n, K]_q$ and let $I \subset \{1..n\}$. The shortening $\mathcal{S}_I(\mathcal{C}) \subset \mathbb{F}_q^{n-\#I}$ of $\mathcal{C}$ at I is the $[n-\#I, K' \geq K - \#I]_q$-code defined by:*

$$\mathcal{S}_I(\mathcal{C}) := \left\{ \boldsymbol{c}_{\{1..n\}\setminus I} \mid \boldsymbol{c} \in \mathcal{C},\ \boldsymbol{c}_I = 0_I \right\}.$$

The shortening operation is in some sense dual to puncturing, namely one has $\mathcal{S}_I(\mathcal{C}^\perp) = \mathcal{P}_I(\mathcal{C})^\perp$ and $\mathcal{S}_I(\mathcal{C})^\perp = \mathcal{P}_I(\mathcal{C}^\perp)$.

2.2 Relevant Material for the Attack on GeMSS

GeMSS [10] is a specific instance of HFEv- which was selected as an alternative candidate in the third round of the NIST PQC standardization process.

HFEv-. The HFEv- signature scheme is a variant of HFE [32] that includes both the Minus and the Vinegar modifiers. The secret polynomial $f : \mathbb{F}_{q^n} \times \mathbb{F}_q^v \to \mathbb{F}_{q^n}$ is of the form

$$f(X, \boldsymbol{y}_v) = \sum_{\substack{i,j \in \mathbb{N} \\ q^i + q^j \leq D}} \alpha_{i,j} X^{q^i + q^j} + \sum_{\substack{i \in \mathbb{N} \\ q^i \leq D}} \beta_i(\boldsymbol{y}_v) X^{q^i} + \gamma(\boldsymbol{y}_v),$$

where $\boldsymbol{y}_v = (y_1, \ldots, y_v)$ are the vinegar variables, $\alpha_{i,j} \in \mathbb{F}_{q^n}$, the β_i's are linear maps $\mathbb{F}_q^v \to \mathbb{F}_{q^n}$ and γ is a quadratic map $\mathbb{F}_q^v \to \mathbb{F}_{q^n}$. The special shape of such an f gives rise to a quadratic central map over the base field $\mathcal{F} = \phi \circ f \circ \psi : \mathbb{F}_q^{n+v} \to \mathbb{F}_q^n$, where

$$\begin{aligned} \psi : \mathbb{F}_q^n \times \mathbb{F}_q^v &\longrightarrow \mathbb{F}_{q^n} \times \mathbb{F}_q^v \\ (x, y) &\longmapsto (\phi^{-1}(x), y). \end{aligned}$$

The public key is then given by a quadratic map $\mathcal{P} = \mathcal{T} \circ \mathcal{F} \circ \mathcal{S}$, where $\mathcal{S} : \mathbb{F}_q^{n+v} \to \mathbb{F}_q^{n+v}$ and $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^{n-a}$ are secret affine maps of maximal rank. For simplification, we assume in the rest of the paper that $\mathcal{S}$ (resp. $\mathcal{T}$) is a linear map described by a matrix $\boldsymbol{S} \in \mathbb{F}_q^{(n+v)\times(n+v)}$ (resp. $\boldsymbol{T} \in \mathbb{F}_q^{n\times(n-a)}$), so that the components of $\mathcal{P} = (p_1, \ldots, p_{n-a})$ are homogeneous polynomials in $N = n + v$ variables $\boldsymbol{x} = (x_1, \ldots, x_{n+v})$. When q is an odd prime power, we recall that $\boldsymbol{P}_i$ is the symmetric matrix associated to p_i by $p_i(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{P}_i\boldsymbol{x}^\mathsf{T}$ for $1 \le i \le n - a$.

MinRank Attack on HFEv- from [35]. Tao et al. recently proposed in [35] the most efficient key recovery attack on HFEv- so far. To describe this attack, we assume that q is an odd prime power, but the results can be extended to the even characteristic, see for instance Appendix A. Let $(\theta_1, \ldots, \theta_n)$ be a basis of the vector space $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$, let $\boldsymbol{H} \in \mathbb{F}_{q^n}^{n\times n}$ be the associated Moore matrix defined by $\boldsymbol{H} := [\theta_{i+1}^{q^j}]_{i,j=0}^{n-1}$ and let $\widetilde{\boldsymbol{H}} := \begin{pmatrix} \boldsymbol{H} & 0 \\ 0 & \boldsymbol{I}_v \end{pmatrix}$. The main step of the attack is by solving the following MinRank problem to recover the first n rows of the invertible matrix $\boldsymbol{U}$ defined by

$$\boldsymbol{U} := \widetilde{\boldsymbol{H}}^{-1}\boldsymbol{S}^{-1} \in \mathbb{F}_{q^n}^{(n+v)\times(n+v)}. \tag{1}$$

Problem 2 (Underlying MinRank problem) *Let $d := \lceil \log_q(D) \rceil$ and let $\boldsymbol{u} \in \mathbb{F}_{q^n}^{n+v}$ be the first row of $\boldsymbol{U}$. Let $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_{n-a} \in \mathbb{F}_q^{(n+v)\times(n+v)}$ denote the symmetric matrices associated with the HFEv- public key and let $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{n+v})$ be the canonical basis for $\mathbb{F}_q^{n+v}$. For $1 \le i \le n + v$, we define the matrix $\boldsymbol{M}_i \in \mathbb{F}_q^{(n-a)\times(n+v)}$ by*

$$\boldsymbol{M}_i := \boldsymbol{e}_i\boldsymbol{P}_* := \begin{pmatrix} \boldsymbol{e}_i\boldsymbol{P}_1 \\ \vdots \\ \boldsymbol{e}_i\boldsymbol{P}_{n-a} \end{pmatrix}.$$

Then, the vector $\boldsymbol{u} := (u_1, \ldots, u_{n+v})$ is a solution to the MinRank instance described by the $\boldsymbol{M}_i$'s with target rank d.

Indeed, the first n rows of $\boldsymbol{U}$ are the Frobenius iterates of $\boldsymbol{u}$, more precisely we have

$$\boldsymbol{U} = \begin{pmatrix} \boldsymbol{u} \\ \vdots \\ \boldsymbol{u}^{[n-1]} \\ \boldsymbol{R} \end{pmatrix},$$

where the block $\boldsymbol{R} \in \mathbb{F}_q^{v\times(n+v)}$ is full rank, see [35, Alg. 1, 4.]. Then, it is shown in [35, §4.3] how one can efficiently derive an equivalent key and finish the attack. Finally, to keep the same notation as in [35, Thm. 2], we set

$$\boldsymbol{Z} := \sum_{i=1}^{n+v} u_i \boldsymbol{M}_i \in \mathbb{F}_q[\boldsymbol{u}]^{(n-a)\times(n+v)}. \tag{2}$$

Fact 1 (On the number of solutions) *Let $\boldsymbol{u} \in \mathbb{F}_{q^n}^{n+v}$ be a solution to the MinRank problem 2. Then, for any $\lambda \in \mathbb{F}_{q^n}^*$, the vector $\lambda\boldsymbol{u} := (\lambda u_1, \ldots, \lambda u_{n+v})$ is another solution. Moreover, for any $0 \leq j \leq n-1$, the same goes for the vector $\boldsymbol{u}^{[j]} := (u_1^{q^j}, \ldots, u_{n+v}^{q^j})$ with corresponding rank d matrix $\boldsymbol{Z}^{[j]}$.*

This fact is inherent to the big-field structure used in HFEv- and was already observed in the previous rank attacks on big-field MPKC [27,25,6,37].

Projection Modifier. The projection modification was introduced in [11] in order to repair the previously broken SFLASH signature scheme [20] and devise the new PFLASH signature scheme. In reaction to the attack on GeMSS from [35], the authors of [31] also applied this modifier to HFEv-, leading to pHFEv-. The proposed parameters for the scheme are secure against this former attack, and the point of projecting is that it appears to be more efficient than simply increasing the degree D of f to obtain the same security. The projection modifier consists in replacing the map $S : \mathbb{F}_q^{n+v} \to \mathbb{F}_q^{n+v}$ by $S = \overline{L} \circ S' : \mathbb{F}_q^{n+v-p} \to \mathbb{F}_q^{n+v}$, where $S' : \mathbb{F}_q^{n+v-p} \to \mathbb{F}_q^{n+v-p}$ is full rank and $\overline{L} : \mathbb{F}_q^{n+v-p} \to \mathbb{F}_q^{n+v}$ is full rank represented by a matrix $\begin{pmatrix} \boldsymbol{\Lambda} & 0 \\ 0 & \boldsymbol{I}_v \end{pmatrix} \in \mathbb{F}_q^{(n+v-p)\times(n+v)}$. The authors of [31] have studied the effect of projection on the rank of the HFEv- central map. When $p > 0$, the rank of $\boldsymbol{Z}$ from Equation (2) is bounded by $d' := d + p$ instead of d (cf. [31, Prop. 2]), and this bound is believed to be tight from practical experiments. Moreover, the number of solutions to the corresponding MinRank problem is expected to be unchanged compared to plain HFEv-. In Table 2, we give the current GeMSS parameter sets as well as those of pHFEv- . In [31], a secure pHFEv- parameter set is constructed from a given GeMSS parameter set by choosing the least value of p such that the minors attack from [35] is just above the security level.

2.3 Relevant Material on Rainbow for Section 7.3

Rainbow is a third round finalist of the NIST PQC standardization process for digital signatures. In this paper, we are mainly interested in the recent rectangular MinRank attack from [7, §7] on this scheme.

Rainbow. For clarity, we adopt the simplified description from [7]. The version of Rainbow submitted to the NIST PQC project is a 2-layered variant of the

Table 2. GeMSS and pHFEv- parameter sets.

Scheme	q	n	v	D	a	p from [31]
GeMSS128	2	174	12	513	12	0
BlueGeMSS128	2	175	14	129	13	1
RedGeMSS128	2	177	15	17	15	4
GeMSS192	2	265	20	513	22	5
BlueGeMSS192	2	265	23	129	22	7
RedGeMSS192	2	266	25	17	23	10
GeMSS256	2	354	33	513	30	10
BlueGeMSS256	2	358	32	129	34	11
RedGeMSS256	2	358	35	17	34	14

well-know UOV signature scheme: the trapdoor consists of 3 $\mathbb{F}_q$-subspaces $O_2 \subset O_1 \subset \mathbb{F}_q^n$ and $W \subset \mathbb{F}_q^m$ of dimension o_2, m and o_2 respectively, and the public system $\mathcal{P}$ contains m quadratic equations in n variables such that $\mathcal{P}(\boldsymbol{z}) \in W$ for all $\boldsymbol{z} \in O_1$ and $\mathcal{P}'(\boldsymbol{x}, \boldsymbol{y}) \in W$ for all $\boldsymbol{x} \in \mathbb{F}_q^n$ and $\boldsymbol{y} \in O_2$, where $\mathcal{P}'$ is the system of *polar forms* associated to $\mathcal{P}$. To perform a key-recovery on Rainbow, it had already been noted that the hardest part is to recover the space O_2: once O_2 is found, it is then easy to recover both W and O_1. Thus, the rectangular MinRank attack from [7] targets secret vectors $\boldsymbol{y} \in O_2$.

Rectangular MinRank attack. The rectangular MinRank attack by [7] is currently the best key-recovery attack on Rainbow so far. For $\boldsymbol{y} \in \mathbb{F}_q^n$, let

$$\boldsymbol{L}_{\boldsymbol{y}} := \begin{pmatrix} \mathcal{P}'(\boldsymbol{e}_1, \boldsymbol{y}) \\ \vdots \\ \mathcal{P}'(\boldsymbol{e}_n, \boldsymbol{y}) \end{pmatrix},$$

where $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n)$ is the canonical basis of $\mathbb{F}_q^n$. The attack heavily exploits the fact that $\mathcal{P}'(\boldsymbol{x}, \boldsymbol{y}) \in W$ for any $\boldsymbol{x} \in \mathbb{F}_q^n$ and $\boldsymbol{y} \in O_2$. Indeed, when $\boldsymbol{y} \in O_2$, the rows of $\boldsymbol{L}_{\boldsymbol{y}}$ lie in W, so that the rank of this matrix is at most $\dim W := o_2$. Also $\boldsymbol{L}_{\boldsymbol{y}} = \sum_{i=1}^{n} y_i \boldsymbol{L}_{\boldsymbol{e}_i}$ by linearity, and therefore a solution to the MinRank instance described by the $\boldsymbol{L}_{\boldsymbol{e}_i}$'s with target rank o_2 is very likely to reveal a vector $\boldsymbol{y}$ in O_2. Finally, as noted in [7], it is possible to fix $o_2 - 1$ entries in $\boldsymbol{y}$ at random in order to obtain a 1-dimensional solution space. The resulting MinRank instance is then solved by relying on the recent Support-Minors modeling [3], see Section 3. Moreover, [7] suggests to also use the fact that $\mathcal{P}(\boldsymbol{y}) = 0$, which allows to consider a system with more equations while keeping the same variables as in the Support-Minors system. The concrete improvement of this trick compared to the plain MinRank attack remains modest, see [7, Table 6].

3 Support-Minors Modeling (SM)

Support-Minors is an efficient method to model and solve the MinRank problem [3]. It has been used to cryptanalyze MPKC and rank-based cryptosystems [2,3,7]. The idea is to factor the secret matrix $\boldsymbol{M} \in \mathbb{K}^{n_r \times n_c}$ of rank $\leq d$ as

$$\boldsymbol{M} := \sum_{i=1}^{N} u_i \boldsymbol{M}_i := \boldsymbol{D}\boldsymbol{C}, \tag{3}$$

where $\boldsymbol{D} \in \mathbb{K}^{n_r \times d}$ and $\boldsymbol{C} \in \mathbb{K}^{d \times n_c}$ are unknown matrices. For $1 \leq j \leq n_r$, one then considers the matrix

$$\boldsymbol{C}_j := \begin{pmatrix} \boldsymbol{r}_j \\ \boldsymbol{C} \end{pmatrix},$$

where $\boldsymbol{r}_j := \boldsymbol{M}_{\{j\},*}$ is the j-th row of $\boldsymbol{M}$ whose components are linear forms in the so-called *linear* variables u_i's. The rank of $\boldsymbol{C}_j$ is at most d, and equations are obtained by setting all $(d+1)\times(d+1)$ minors of this matrix to zero, namely $Q_{j,J} := |\boldsymbol{C}_j|_{*,J}$ for $J \subset \{1..n_c\},\ \#J = d+1$. The Support-Minors system then contains a total of $n_r\binom{n_c}{d+1}$ polynomials by considering $1 \leq j \leq n_r$. Moreover, by using Laplace expansion along the first row of $\boldsymbol{C}_j$, one notices that these equations are bilinear in the u_i variables and in the so-called *minor* variables $c_T := |\boldsymbol{C}|_{*,T}$, where $T \subset \{1..n_c\},\ \#T = d$. The following fact will be used several times in the paper.

Fact 2 (Structure of the SM system) *Each SM equation contains at most $N(d+1)$ bilinear monomials. More precisely, given $J \subset \{1..n_c\},\ \#J = d+1$ and $1 \leq j \leq n_r$, the monomials of $Q_{j,J}$ belong to a set of $N(d+1)$ elements which only depends on J.*

Proof. Let $J := \{j_1 < \cdots < j_{d+1}\}$ and $1 \leq j \leq n_r$. By Laplace expansion along the first row of $(\boldsymbol{C}_j)_{*,J}$, one has that the monomials in $Q_{j,J}$ are in the set

$$A_J := \left\{ u_i \boldsymbol{c}_{J\setminus j_u} :\ 1 \leq u \leq d+1,\ 1 \leq i \leq N \right\}.$$

This set contains $N(d+1)$ elements which are independent from j. □

Solving the SM System. When the corresponding MinRank problem has a unique solution, [3] proposes a dedicated XL approach by multiplying the SM equations by monomials in the linear variables. This is typically the case for Rainbow [7] or rank-based cryptography [3,2]. The attack constructs the Macaulay matrix $\boldsymbol{M}(\mathcal{Q}_b)$, where $\mathcal{Q}_b$ is the system of all degree $b+1$ polynomials of the form $\mu_{\boldsymbol{u}} f$, where $\mu_{\boldsymbol{u}}$ is a monomial of degree $b-1$ in the linear variables and f is a SM equation. Note that direct linearization corresponds to $b = 1$ with $\mathcal{Q}_1 = \mathcal{Q}$. The value of b is chosen such that the rank of $\boldsymbol{M}(\mathcal{Q}_b)$ is equal to the number of columns minus one. In this case, the linear system $\boldsymbol{M}(\mathcal{Q}_b)\boldsymbol{x}^{\mathsf{T}} = 0$ has a non-trivial solution, and this solution easily yields a solution to the initial MinRank problem. The situation is quite different when there are $N' > 1$ solutions to

this original MinRank instance, *e.g.* HFE, see Fact 1, but the approach can be adapted. There still exists a value of b for which the kernel of $\boldsymbol{M}(\mathcal{Q}_b)$ is non-trivial and can be computed, but the dimension N'' of this kernel is expected to be > 1. In particular, the second step to solve the initial MinRank problem from arbitrary kernel vectors is no longer straightforward. By finding a basis of that kernel one can at least reduce the initial MinRank problem to a new one with N'' matrices with the same dimensions and the same target rank d, but this secondary MinRank instance has no reason to be much easier to solve.

The linear system $\boldsymbol{M}(\mathcal{Q}_b)\boldsymbol{x}^\mathsf{T} = 0$ is usually sparse, especially when $b > 1$, and in this case it is often advantageous to use the Wiedemann algorithm. Another idea to reduce the cost of linear algebra is to start from a Macaulay matrix of smaller size by selecting only $n' \leq n_c$ columns in $\boldsymbol{M}$ (for example the first n' ones), which yields a SM system with $n_r\binom{n'}{d+1}$ equations and $N\binom{n'}{d+1}$ monomials $u_i c_T$, where this time $T \subset \{1..n'\}$.

4 Improved Attack on G*e*MSS Using Support-Minors

In this section, we describe our approach to solve the MinRank instance 2 arising from HFEv- using Support-Minors. As noted in Fact 1, this problem is expected to have several solutions which are triggered by the big-field structure, hence we cannot directly apply the XL techniques from [3]. Two remarks are in order before we describe the attack. From the definition of $\boldsymbol{Z}$ in Equation (2) and the fact that the $\boldsymbol{M}_i$'s are over the small field, it is important to notice that the coefficients of the SM system are in $\mathbb{F}_q$, whereas the solutions may belong to the extension field $\mathbb{F}_{q^n}$. Also, as discussed in [3], we will consider a subset of the SM equations coming from a submatrix of $\boldsymbol{Z}^\mathsf{T} \in \mathbb{F}_q[\boldsymbol{u}]^{(n+v)\times(n-a)}$ obtained by selecting a subset J of $n' \in [d', n-a]$ columns.

4.1 Fixing Variables in the Support-Minors System

Up to relabelling of the linear variables, one can fix $u_{n+v} = 1$ as in [35]. In this case, one expects to obtain n solutions which correspond to the first n rows of $\boldsymbol{U}$, namely $\boldsymbol{u}, \boldsymbol{u}^{[1]}, \ldots, \boldsymbol{u}^{[n-1]}$. Also, since we can choose an arbitrary submatrix $\boldsymbol{Z}^\mathsf{T}_{*,J}$ of $\boldsymbol{Z}^\mathsf{T}$ with $\#J = n'$, we can make sure that this submatrix is full rank on its first d columns. Therefore, we will fix the minor variable $c_{\{1\ldots d\}}$ to 1.

Modeling 1 (Support-Minors modeling on $\boldsymbol{Z}^\mathsf{T}$) *Let $\boldsymbol{Z}$ be as defined in Equation (2). We consider the SM equations obtained by choosing $n' \leq n-a$ columns in $\boldsymbol{Z}^\mathsf{T}$, with coefficients in $\mathbb{F}_q$ and solutions in $\mathbb{F}_{q^n}$. Moreover, we fix $u_{n+v} = 1$ and $c_{\{1\ldots d\}} = 1$.*

The system from Modeling 1 contains $(n+v)\binom{n'}{d+1}$ affine bilinear equations in $(n+v)\binom{n'}{d}$ monomials, and $(n+v-1)(\binom{n'}{d}-1)$ of them are bilinear monomials. Also, one can choose a number of columns $n' \leq n-a$ that yields a sub-system with more equations than monomials. Indeed, this will be the case when $(n+$

$v)\binom{n'}{d+1} \geq (n+v)\binom{n'}{d}$, and this condition is equivalent to $n' \geq 2d+1$. Finally, in GeMSS the value of $n-a$ is much higher than $2d+1$, which allows to choose $n' \in [2d+1, n-a]$.

4.2 Solving via Gröbner Bases when $n' \geq 2d+1$

In the case when $n' \geq 2d+1$, there are more equations than monomials in the SM system, but once again it is not possible to solve by direct linearization because the resulting linear system has a large kernel. More precisely, since we expect the system to have n solutions and since these solutions correspond to n linearly independent vectors $\{\boldsymbol{v}, \boldsymbol{v}^{[1]}, \ldots, \boldsymbol{v}^{[n-1]}\}$ such that the first $n+v-1$ components of $\boldsymbol{v}$ are $u_1, \ldots, u_{n+v-1}$, its dimension should be at least n. For large enough n, in every single instance we have tested, the linearization process triggers no spurious solutions, thus the dimension of the solution space is equal to n. Therefore, we adopt the following Assumption 1 in the rest of the analysis.

Assumption 1 *Let $n' \geq 2d+1$. Then, the number of linearly independent equations in Modeling 1 is equal to*

$$\mathcal{N}_1 := (n+v)\binom{n'}{d} - n.$$

Our attack works in two steps. First, by forming linear combinations between the equations from Modeling 1, we are able to produce a system $\mathcal{L}$ of degree 1 polynomials (Step 1). Then, using $\mathcal{L}$ to substitute some of the variables, we get a quadratic system in $n_u = n-1$ of the linear variables. Finally (Step 2) we solve this second system.

Step 1: Linear Polynomials Produced at $b=1$. Here we explain how the system $\mathcal{L}$ is obtained at Step 1. We start by proving

Fact 3 *Under Assumption 1, by linear algebra on the affine SM equations, one can generate $\mathcal{N}_L$ linearly independent degree 1 polynomials, where*

$$\mathcal{N}_L \geq \binom{n'}{d} + v - 1. \tag{4}$$

Proof. By Assumption 1, the system given in Modeling 1 contains $\mathcal{N}_1 := (n+v)\binom{n'}{d} - n$ linearly independent equations. Moreover, one has

$$\mathcal{N}_1 \geq (n+v-1)\left(\binom{n'}{d} - 1\right),$$

so that the number of linearly independent *affine bilinear* equations is greater than the number of *bilinear* monomials. In particular, there are non-trivial linear combinations between the bilinear parts of the equations that are zero. This means that by performing linear algebra operations on the equations in Modeling 1, one can generate at least

$$\underbrace{\left((n+v)\binom{n'}{d} - n\right)}_{\mathcal{N}_1} - \underbrace{(n+v-1)\left(\binom{n'}{d} - 1\right)}_{\#\text{bilinear monomials}} = \binom{n'}{d} + v - 1$$

linearly independent affine degree 1 polynomials in the u_i's and in the c_T variables. □

These linear equations are often referred to in the literature as *degree falls* from degree 2 to degree 1. Since producing a higher number of independent degree 1 equations would be even more beneficial for our attack, we assume the worst case scenario, namely

Assumption 2 *There are exactly*

$$\mathcal{N}_L = \binom{n'}{d} + v - 1.$$

linearly independent degree 1 *polynomials in the span of Modeling 1.*

We consider the Macaulay matrix $\boldsymbol{M}(L)$ of these $\mathcal{N}_L$ linear polynomials, and here we choose to eliminate all the $n_{c_T} := \binom{n'}{d} - 1$ minor variables first by considering an ordering on the columns such that $c_T > u_{n+v-1} > \cdots > u_1 > u_{n+v} = 1$. We also denote by $\boldsymbol{M}(L^h)$ the Macaulay matrix of the system which consists of the homogeneous degree 1 parts, which is obtained from $\boldsymbol{M}(L)$ by removing the last column.

Lemma 1 *Under Assumption 1, the reduced row echelon form of* $\boldsymbol{M}(L^h)$ *is of the form*

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{I}_{n_{c_T}} & * \\ 0 & \boldsymbol{K} \end{pmatrix} \in \mathbb{F}_q^{\mathcal{N}_L \times (n_{c_T}+n+v-1)}, \tag{5}$$

where $\boldsymbol{K} \in \mathbb{F}_q^{v \times (n+v-1)}$ *is row reduced.*

Proof. Let $\boldsymbol{U} \in \mathbb{F}_{q^n}^{(n+v)\times(n+v)}$ be the invertible matrix defined in Equation (1), and let $\boldsymbol{u} := (u_1, \ldots, u_{n+v}) \in \mathbb{F}_{q^n}^{n+v}$ be its first row. This vector is a solution to the MinRank problem, so that there exists $\boldsymbol{v} \in \mathbb{F}_{q^n}^{n_{c_T}}$ such that

$$\boldsymbol{M}(L) \cdot (\boldsymbol{v}, u_{n+v-1}, \ldots, u_1, u_{n+v})^{\mathsf{T}} = \boldsymbol{0}.$$

By Assumption 1, the matrix $\boldsymbol{M}(L^h)$ is full rank. We proceed by contradiction by assuming that the echelon form $\boldsymbol{L}$ of $\boldsymbol{M}(L^h)$ is not in systematic form on its first n_{c_T} rows. On that hypothesis, there is a set of $v_0 \geq v+1$ linearly independent vectors in the row space of $\boldsymbol{L}$ which have zero in their leftmost n_{c_T} entries. This yields v_0 linearly independent vectors $\boldsymbol{h}_1, \ldots \boldsymbol{h}_{v_0} \in \mathbb{F}_q^{n+v}$ such that for all i, $\boldsymbol{u}\boldsymbol{h}_i^{\mathsf{T}} = 0$. Then, by applying the Frobenius isomorphism and using the fact that it is the identity on $\mathbb{F}_q$, it follows that $\boldsymbol{u}^{[j]}\boldsymbol{h}_i^{\mathsf{T}} = 0$ for all i and $0 \leq j \leq n-1$. Therefore, the matrix

$$\boldsymbol{U}_{\{1..n\},*} = \begin{pmatrix} \boldsymbol{u} \\ \vdots \\ \boldsymbol{u}^{[n-1]} \end{pmatrix} \in \mathbb{F}_{q^n}^{n \times (n+v)},$$

is not full rank, which is a contradiction since $\boldsymbol{U}$ is invertible. □

By Lemma 1, it is possible to express all the minor variables as well as v linear variables in terms of the remaining $n-1$ linear variables. Moreover, by reordering the linear variables if necessary, we may further assume that the remaining ones are $u_1, \ldots, u_{n-1}$. In this case, the matrix $\boldsymbol{L}$ of Lemma 1 is of the form

$$\boldsymbol{L} := \begin{pmatrix} \boldsymbol{I}_{n_{c_T}} & 0 & \boldsymbol{Y} \\ 0 & \boldsymbol{I}_v & \boldsymbol{W} \end{pmatrix} \in \mathbb{F}_q^{\mathcal{N}_L \times (n_{c_T}+n+v-1)}, \tag{6}$$

where $\boldsymbol{Y} \in \mathbb{F}_q^{n_{c_T} \times n_u}$, $\boldsymbol{W} \in \mathbb{F}_q^{v \times n_u}$ and $n_u := n-1$. Finally, let $\mathcal{L}$ be the affine system obtained by performing the same linear operations on the initial $\mathcal{N}_L$ equations as the ones to produce the echelon form $\boldsymbol{L}$ starting from $\boldsymbol{M}(L^h)$.

Step 2: Solving the Resulting Quadratic System. By using the linear equations from $\mathcal{L}$ to substitute variables in Modeling 1, we obtain the following

Modeling 2 (Quadratic system) *We consider the quadratic system in $n_u = n-1$ linear variables $u_1, \ldots, u_{n-1}$ obtained by plugging the linear polynomials of $\mathcal{L}$ into the equations from Modeling 1.*

We now focus on the task of solving this quadratic system using Gröbner bases, and in Proposition 1 we rigorously prove at which degree the computation terminates. The proof relies on Assumption 1, Assumption 2 and the following Assumption 3 on the echelon form $\boldsymbol{L}$ from Equation (6).

Assumption 3 *The matrix $\boldsymbol{Y} \in \mathbb{F}_q^{n_{c_T} \times n_u}$ in Equation (6) is full rank.*

Note that this assumption should hold with high probability if $\boldsymbol{Y}$ behaves as a random matrix. Also, we have performed different simulations to experimentally verify Assumptions 1, 2 and 3. According to the results obtained for different sets of parameters (q, n, v, D, a), it seems that if n' is chosen such that $n' \geq 2d+1$ and $n_{c_T} \geq n_u$, then all 3 assumptions are satisfied almost 100% of the times.

jb: The reader might find helpful to experimentally explore these 3 assumptions using the SageMath notebook [39].

Proposition 1. *Under Assumptions 1, 2 and 3, if $n_{c_T} \geq n_u$, a Gröbner basis of the system from Modeling 2 can be obtained by Gaussian elimination on the initial equations,* i.e. *it is found at degree* 2. *When $n_{c_T} < n_u$, this Gröbner basis is found at degree* 3.

Proof. By Assumption 1 and Assumption 2, the number of degree 2 affine equations which remain after the linear algebra step in Modeling 1 is equal to $\mathcal{N}_1 - \mathcal{N}_L = (n+v-1)\left(\binom{n'}{d} - 1\right)$. As we cannot construct extra degree falls between them, this implies that the linear span of these equations contains an equation with leading monomial $u_i c_T$ for any T, $\#T = d$, $T \neq \{1..d\}$ and any $1 \leq i \leq n_u + v$. Let

$$\boldsymbol{L} := \begin{pmatrix} \boldsymbol{I}_{n_{c_T}} & 0 & \boldsymbol{Y} \\ 0 & \boldsymbol{I}_v & \boldsymbol{W} \end{pmatrix} \in \mathbb{F}_q^{\mathcal{N}_L \times (n_{c_T}+n+v-1)},$$

where $\boldsymbol{Y} \in \mathbb{F}_q^{n_{c_T} \times n_u}$, $\boldsymbol{W} \in \mathbb{F}_q^{v \times n_u}$ and $n_u := n - 1$ as defined in Equation (6). We also denote by $\boldsymbol{c} \in \mathbb{F}_{q^n}^{n_{c_T}}$ the row vector whose components are the minor variables and $(u_1, \ldots, u_{n+v-1}) := (\boldsymbol{u}_+, \boldsymbol{u}_-)$, where $\boldsymbol{u}_+ \in \mathbb{F}_{q^n}^{n_u}$ (remaining linear variables) and $\boldsymbol{u}_- \in \mathbb{F}_{q^n}^{v}$ (removed linear variables). Then, there is vector of constants $\alpha \in \mathbb{F}_q^{n_{c_T}}$ such that

$$\boldsymbol{c}^\mathsf{T} = -\boldsymbol{Y}\boldsymbol{u}_+^\mathsf{T} - \alpha^\mathsf{T}. \tag{7}$$

Since $\boldsymbol{Y}$ is full rank by Assumption 3, the linear system given by Equation (7) can be inverted when $n_{c_T} \geq n_u$, and therefore all the $\binom{n_u+1}{2}$ quadratic leading monomials will be found in the span of Modeling 2. The second part of the proof, which discusses the case $n_{c_T} < n_u$, can be found in Appendix B. □

Finally, note that the content of the current Section 4 also applies to pHFEv- with rank equal to $d' = d + p$, since what really matters in the analysis is the number of solutions to the MinRank problem. We simply have to replace the condition $n' \geq 2d + 1$ by $n' \geq 2d' + 1$.

5 Complexity of the Attack

This section analyses the cost of our attack on G*e*MSS. In Sections 5.1 and 5.2, we estimate the time complexity. This complexity comes down to two major steps, first generating Modeling 2 from Modeling 1 (Step 1) and then solving Modeling 2 via Gröbner bases (Step 2). Then, in Section 5.3, we evaluate the corresponding memory complexity.

First, note that choosing $n' = 2d + 1$ already ensures $n_{c_T} \geq n_u$ for all the G*e*MSS and pHFEv- parameters, see Table 2. In particular, Proposition 1 implies that the system in Modeling 2 will be solved at degree 2. In the following, we then adopt $n' = 2d + 1$ and we will also consider that $v = o(n)$.

5.1 Time Complexity of Step 1

This first step can be performed by echelonizing the equations from Modeling 1 using Strassen's algorithm. The complexity in this case is

$$\mathcal{O}\left((n+v)\binom{2d+1}{d}\left((n+v)\binom{2d+1}{d}\right)^{\omega-1}\right) = \mathcal{O}\left(n_{c_T}^{\omega} n_u^{\omega}\right) \tag{8}$$

$\mathbb{F}_q$-operations, where $n_u = n - 1$, $n_{c_T} = \binom{2d+1}{d} - 1$ and ω is the linear algebra constant.

An alternative path is to use Coppersmith's Block-Wiedemann algorithm (BW). Let $\mathcal{M}$ be the row space of the Macaulay matrix $\boldsymbol{M}(\mathcal{Q})$ of the SM system. By Assumption 1, it can be seen as a linear code of length $(n+v)\binom{2d+1}{d}$ and dimension $\mathcal{N}_1 = (n+v)\binom{2d+1}{d} - n$, so that we expect the right kernel of $\boldsymbol{M}(\mathcal{Q})$ to be of dimension n. In particular, by running BW roughly n times, we hope to

obtain a basis for this kernel which corresponds to the dual code $\mathcal{C} := \mathcal{M}^{\perp}$. Let I be the subset of positions of $\mathcal{M}$ corresponding to the bilinear monomials. We then puncture $\mathcal{C}$ at I to obtain $\mathcal{P}_I(\mathcal{C})$. Since the dual of the punctured code is the shortening of the dual, we have that $\mathcal{P}_I(\mathcal{C})^{\perp} = \mathcal{S}_I(\mathcal{M})$, and the dimension of this code corresponds to the number of independent linear equations $\mathcal{N}_{\mathcal{L}}$ given by Fact 3. Moreover, by Assumption 2 we have that $\mathcal{N}_{\mathcal{L}} = \binom{2d+1}{d} + v - 1$. Also, the cost of obtaining the shortened code $\mathcal{S}_I(\mathcal{M})$ from $\mathcal{P}_I(\mathcal{C})$ is negligible compared to the BW step to obtain $\mathcal{P}_I(\mathcal{C})$. Finally, by Fact 2, there are at most $(d+1)(n+v)$ monomials in one SM equation, so that the overall complexity using the Wiedemann algorithm n times to find a basis of $\mathcal{C}$ is

$$\mathcal{O}\left(n \times (n+v)(d+1)\left((n+v)\binom{2d+1}{d}\right)^2\right) = \mathcal{O}\left(dn_{c_T}^2 n_u^4\right). \tag{9}$$

5.2 Time Complexity of Step 2

As the choice $n' = 2d+1$ ensures $n_u \leq n_{c_T}$ for all the parameters of GeMSS and pHFEv-, the system given by Modeling 2 can be solved at degree 2 by Proposition 1. Thus, the cost of this second step is simply the cost of row reducing the Macaulay matrix of this quadratic system. The number of columns is the number of initial monomials which is equal to $1 + n_u + \binom{n_u+1}{2}$ and there are more equations than monomials, so that the complexity of the second step is

$$\mathcal{O}\left(n_{c_T}(n+v-1) \times \left(1 + n_u + \binom{n_u+1}{2}\right)^{\omega-1}\right) = \mathcal{O}\left(n_{c_T} n_u^{2\omega-1}\right) \tag{10}$$

$\mathbb{F}_q$-operations. Note that Step 1 is expected to be more costly since $n_u \leq n_{c_T}$.

5.3 Memory Cost

In this section, we estimate the space complexity of the attack on GeMSS, which is dominated by the space complexity of Step 1 as the system from Modeling 2 is much smaller. We choose $q = 2$ to be in accordance with the GeMSS parameters, so that one element in $\mathbb{F}_q$ occupies one bit in memory. We start by describing two approaches to store the Macaulay matrix $\boldsymbol{M}(\mathcal{Q})$ associated with the system $\mathcal{Q}$ from Modeling 1 when used within the Block-Wiedemann algorithm.

Standard Approach. This approach uses the sparsity of the matrix $\boldsymbol{M}(\mathcal{Q})$ in a naive way. Recall from Fact 2 that every SM equation contains at most $(n+v)(d+1)$ nonzero monomials. Thus, one way to store a single row of $\boldsymbol{M}(\mathcal{Q})$ is by storing the indexes corresponding to nonzero positions. Hence we must store at most $(n+v)(d+1)$ column indexes per row. Since the Macaulay matrix has $(n+v)\binom{2d+1}{d}$ columns and assuming that several rows can be dropped to get a square matrix, the space complexity is given by

$$\binom{2d+1}{d}(d+1)(n+v)^2 \log_2\left(\binom{2d+1}{d}(n+v)\right) = \mathcal{O}\left(dn_u^2 n_{c_T} \log_2(n_{c_T})\right). \tag{11}$$

Optimized Approach. Here we adapt to the SM equations the strategy used by Niederhagen for a generic Macaulay matrix [30, §4.5.3]. Contrary to [30], we cannot exploit the fact that several rows of our Macaulay matrix correspond to the same equation multiplied by distinct monomials as we consider the plain SM system at $b = 1$. Instead, we take advantage of the structure of the SM equations as described in Fact 2. Our approach is detailed in Appendix C, and we obtain a space complexity of

$$\binom{2d+1}{d}(n+v)(d+1)\log_2\left(\binom{2d+1}{d}(n+v)\right) = \mathcal{O}\left(dn_u n_{c_T}\log_2(n_{c_T})\right). \tag{12}$$

Note that this approach saves a factor of order $n + v$ compared to the naive approach, see Equation (11).

Table 3. Memory ($\log_2$(#bytes)) needed to store the Macaulay matrix $\boldsymbol{M}(\mathcal{Q})$ from Step 1 to be used in BW or Strassen's algorithm.

Scheme	BW Standard	BW Optimized	Strassen
GeMSS128	38.665	34.553	48.935
BlueGeMSS128	34.332	30.258	41.263
RedGeMSS128	27.645	23.729	29.873
GeMSS192	39.930	35.213	50.166
BlueGeMSS192	35.586	30.917	42.478
RedGeMSS192	28.897	24.410	31.073
GeMSS256	40.836	35.686	51.049
BlueGeMSS256	36.488	31.389	43.353
RedGeMSS256	29.800	24.905	31.940

Table 3 shows the space complexity of the first step of our attack. Keep in mind that the memory demand for the Block-Wiedemann algorithm will not be much more than the one to fully store the Macaulay matrix. It can even be significantly lower, if rows are generated on-demand, but this would increase the time complexity. In contrast, the space complexity of Strassen's algorithm is dominated by the memory demand to store a square dense matrix of size $\binom{2d+1}{d}(n+v)$, see Column "Strassen".

As we can see in Table 3, the Optimized storage requires only a few GigaBytes of shared memory to execute Step 1 with BW on any of the proposed parameters for G*e*MSS, whereas for the Standard approach requires up to a few TeraBytes. To perform this step with Strassen's algorithm, one would need up to more than two Petabytes. To sum up, the amount of memory required by BW is small enough to be allocated even in a shared memory device, especially if one uses the Optimized storage.

6 Application to G*e*MSS and pHFEv- Parameter Sets.

In this section, we use the results developed in Section 5 to determine the effect of our attack on the security of the G*e*MSS and pHFEv- signature schemes.

In Table 4, we give the time complexity of our attack on the current G*e*MSS parameters. We use Equation (8) or Equation (9) for Step 1 (Strassen or BW) and Equation (10) for Step 2. We use $\omega = 2.81$ and a conservative constant of 7 for the concrete complexity of Strassen's algorithm [38], while a constant of 3 for the concrete complexity of BW [26, Theorem 2]. One can check that for the specific parameters proposed by the G*e*MSS team, the value $n' = 2d + 1$ is high enough to ensure to solve at degree 2 in Step 2, *i.e.* $n_u \leq n_{c_T}$. Similarly, the behavior of our attack on pHFEv- is given in Table 5. We adopt the parameters from [31, Table 2] using $\omega = 2.81$. In this paper, the value of p was chosen such that the minors attack from [35] is just above the security level. On these parameters, one notices that our attack always succeeds in solving at degree 2 with $n' = 2d' + 1 = 2(d + p) + 1$. As before, for those parameters the values of d' are indeed high enough to guarantee $n_u \leq n_{c_T}$.

Table 4. Complexity of our attack ($\log_2$(#gates)) versus known attacks from [35] for the G*e*MSS parameters.

Scheme	Minors [35]	SM [35]	SM Step 1 (Strassen/BW)	SM Step 2 (Strassen)	n'
G*e*MSS128	139	118	76/72	54	21
BlueG*e*MSS128	119	99	65/65	51	17
RedG*e*MSS128	86	72	49/53	45	11
G*e*MSS192	154	120	78/75	57	21
BlueG*e*MSS192	132	101	67/67	53	17
RedG*e*MSS192	95	75	51/55	48	11
G*e*MSS256	166	121	79/77	59	21
BlueG*e*MSS256	141	103	68/69	55	17
RedG*e*MSS256	101	76	52/57	50	11

The nature of our approach, although in theory similar to the one used in [35], allows us to reduce significantly the complexity of the Support-Minors attack performed by Tao et al. against G*e*MSS. This is important since this improvement makes it completely infeasible to repair G*e*MSS by simply increasing the size of its parameters without turning it into an impractical scheme. The dominant cost of our attack is the initial linear algebra step (dense or sparse) on the Support-Minors equations, whereas in [35] an attacker needs to multiply these equations by linear and/or minor variables to solve the system. This explains why we obtain a much smaller cost than the one presented in column "SM [35]". Another noticeable difference between our work and the one in [35] is that their complexity estimate is conjectural, whereas ours is proven under

mild assumptions in comparison. The results from Table 5 also suggest that the projection modifier on HFEv- will not be sufficient to repair the scheme as we have significantly broken the parameters given in [31]. To meet the new security levels, the value of p should be increased by a consequential amount, making the scheme inefficient. For example, to achieve security level 128 with the former GeMSS128 parameters, one should take $p = 14$, increasing the signing time by a factor q^{14}, which is considerable.

Table 5. Complexity of our attack ($\log_2$(#gates)) versus known attacks from [35] for pHFEv-. The pHFEv- parameter set for level x consists of (q, n, v, D, a, p), where (q, n, v, D, a) is taken from GeMSSx and $p \geq 0$ is the smallest value such that the cost of the minors attack [35] is just above x.

Scheme	p	Minors [35,31]	SM Step 1 (Strassen/BW)	SM Step 2 (Strassen)	n'
GeMSS128	0	139	76/72	54	21
BlueGeMSS128	1	128	71/69	53	19
RedGeMSS128	4	128	71/69	53	19
GeMSS192	5	201	105/95	67	31
BlueGeMSS192	7	201	105/95	67	31
RedGeMSS192	10	205	105/95	67	31
GeMSS256	10	256	134/117	79	41
BlueGeMSS256	11	256	129/113	77	39
RedGeMSS256	14	263	129/113	77	39

7 Memory Management Strategy for the Support-Minors Equations within Block Wiedemann

It is generally acknowledged that the cost of the Wiedemann algorithm is dominated by the combined cost of a large number of matrix-vector multiplications, where the matrix is a fixed, full-rank, square submatrix $\boldsymbol{M}(\mathcal{P})'$ of the initial Macaulay matrix $\boldsymbol{M}(\mathcal{P})$. This matrix-vector product occurs approximately $3V$ times, where V is the dimension of the vector $\boldsymbol{v}$ being multiplied. While the cost of these multiplications is often expressed in terms of the number of field operations involved, it is likely that for cryptographically-interesting instances, this cost is dominated instead by random-access queries to a large memory. In [36], the cost of a random access memory query is estimated by the formula

$$C_2 \log_2 V \sqrt{V \log_2 q}, \tag{13}$$

where $C_2 > 0$ is a constant, and it is asserted that such a random access must occur every time a field multiplication is performed in the Wiedemann algorithm. Here, [36] follows [5] in estimating the cost of moving a bit in a memory

of size $V \log_2 q$ – the size of $\boldsymbol{v}$ – as $C_2\sqrt{V \log_2 q}$. The additional factor of $\log_2 V$ in Equation (13) appears to be included on the assumption that each field multiplication requires not just the transmission of a field element but a $\log_2 V$-bit address for that field element. Here, note that it is not necessary to store the Macaulay matrix, since, at least for cryptographically large systems, the necessary matrix elements can be generated on-the-fly algorithmically with negligible cost compared to the cost of random access queries to memory. However, we do not see any way to avoid storing $\boldsymbol{v}$ while the value of the matrix-vector product is being written to memory. In our analysis, we will therefore take the size of the memory to be $2V \log_2 q$ instead of $V \log_2 q$.

More importantly, we challenge the assertion of [36] that $\log_2 V$ bits must be moved long distance in memory for every field multiplication for the Support-Minors system, namely when $\mathcal{P} := \mathcal{Q}_b$ for some $b \geq 1$. To this end, we propose a simple hashing strategy to obviate much of this memory access cost. In our methodology, the cost of the matrix-vector products that dominate the cost of the Wiedemann algorithm approaches one long distance memory access to a field element per active row of $\boldsymbol{M}(\mathcal{Q}_b)'$ per matrix-vector multiplication, and moreover memory accesses are blocked so that the cost of transmitting memory addresses is negligible. Assuming the same cost formula for generic RAM access as [36], this implies that the cost of the Wiedemann algorithm should be quite close to

$$3V^2 \log_2 q \cdot C_2\sqrt{2V \log_2 q}. \tag{14}$$

As the matrix-vector product involves w field multiplications for each row of $\boldsymbol{M}(\mathcal{Q}_b)'$, where w is the number of nonzero elements in the row, this idealized complexity value would represent a memory-access cost savings factor of $w \log_2 V / \log_2 q$ in comparison to the model of [36]. It should be noted that the memory cost formulae of [5] and [36] are based on an assumed 2-dimensional arrangement of memory units. If we instead assume a 3-dimensional memory model, we closely approximate a similar formula for the cost with $(2V \log_2 q)^{1/3}$ substituted for $\sqrt{2V \log_2 q}$ and a different constant. For brevity, we will derive our formulae in terms of the 2-dimensional memory model with the understanding that it is a trivial matter to adjust them to the 3-dimensional model.

7.1 Hashing on the main memory

Each coordinate of a matrix-vector product performed within the Wiedemann algorithm is computed by multiplying a vector-vector product of the form

$$\boldsymbol{r}\boldsymbol{v}^{\mathsf{T}} = \sum_{i \in \mathrm{Supp}(\boldsymbol{r})} r_i v_i,$$

where $\boldsymbol{r}$ is a row of the Macaulay matrix with support $\mathrm{Supp}(\boldsymbol{r})$ of size w, whose elements can be cheaply computed on the fly. The cost estimates in [36] effectively assign a cost of w random access queries in a memory of size V to perform this vector-vector product. This would be accurate if the corresponding sum is computed by a central processor which first computes the nonzero elements r_i,

then fetches v_i for $i \in \mathrm{Supp}(\boldsymbol{r})$ from memory, and finally multiplies each r_i by the corresponding v_i and sums the products.

The strategy we propose, however, will partition the main memory in which the v_i's are stored, so that for each row $\boldsymbol{r}$, the v_i's with $i \in \mathrm{Supp}(\boldsymbol{r})$ will be clustered into a small number of groups such that the v_i's in each group are all in the same memory partition. This allows a distributed approach where a processor assigned to each memory partition Π computes the nonzero r_i's for $v_i \in \Pi$ and then computes the partial sum $\sum_{v_i \in \Pi} r_i v_i$. This partial sum is then transmitted by each such processing cluster to that cluster among them located in the section of memory where the total sum is to be written. Thus, most of the arithmetic is performed locally, within each partition, with "remote" communication only between the small number of relevant processing clusters.

To establish this partition, we observe that each pair of memory addresses—a read address for a coordinate of the vector $\boldsymbol{v}$ and a write address for the same coordinate of the product $\boldsymbol{M}(\mathcal{Q}_b)'\boldsymbol{v}^\mathsf{T}$— corresponds to a fixed bi-degree $(b, 1)$ monomial. Also, any row $\boldsymbol{r}$ is associated to an equation of $\mathcal{Q}_b$ of the form $\mu Q_{i,J}$, where J is a collection of $d+1$ columns of the support matrix $\boldsymbol{M}$. The thing each monomial in such an equation has in common is that the minor variable present corresponds to a subset of J of size d; that is, it belongs to the set $\{c_{J\setminus\{j\}}, j \in J\}$. We may thus define an h-bit hash H for each monomial, whose value represents the inclusion or exclusion of the first h columns of the support matrix from the minor corresponding to this minor variable. Since there is significant overlap in which columns are present in a minor corresponding to a minor variable within each equation, we expect each row $\boldsymbol{r}$ to involve relatively few possible hash values, thereby minimizing "remote" communication.

We may assume, as in [36] and [13], that the cost of distributing the MinRank instance and a seed for a PRNG to generate the same square submatrix of the Macaulay matrix to the 2^h processing clusters arising from our hashing strategy is insignificant in comparison to the cost of running the Wiedemann algorithm. Thus, the hashing strategy has the potential to produce a significant savings in memory access cost by making the vast majority of the multiplications in the Wiedemann algorithm local.

7.2 Memory savings from our approach

The rationale behind our strategy is the specific structure of the SM system, see Fact 2. First, note that rows of the full Macaulay matrix $\boldsymbol{M}(\mathcal{Q}_b)$ can be grouped in blocks of size n_r of the form $\{\mu Q_{i,J},\ 1 \le i \le n_r\}$, where J is a collection of $d+1$ columns of the support matrix $\boldsymbol{M}$ and where μ is a fixed monomial of degree $b-1$ in the linear variables. While not all of these n_r rows of $\boldsymbol{M}(\mathcal{Q}_b)$ are present in the square submatrix $\boldsymbol{M}(\mathcal{Q}_b)'$ input into the Wiedemann algorithm, on average

$$n_r' = \frac{\mathrm{rank}(\boldsymbol{M}(\mathcal{Q}_b))}{\#\mathrm{blocks}} \approx \frac{\#\mathrm{cols}(\boldsymbol{M}(\mathcal{Q}_b))}{\#\mathrm{blocks}} = \frac{\binom{N+b-1}{b}\binom{n_c}{d}}{\binom{N+b-2}{b-1}\binom{n_c}{d+1}}$$

such equations are included from each block. Fact 2 states that these equations have potential nonzero coefficients for $N(d+1)$ monomials all involving the same set of minor variables, and thus memory access patterns arising from vector-vector products involving these rows will be the same.

In our approach, each of these equations is considered by a given processing cluster in function of the presence or absence of columns of $\boldsymbol{M}$ in the calculation of that equation. Note that the total number of choices of $d+1$ of the $n_c \geq h$ columns can be written

$$\underbrace{\binom{n_c}{d+1}}_{\#\text{All patterns}} = \sum_{i=d+h+1-n_c}^{h} \binom{h}{i}\binom{n_c-h}{d+1-i},$$

where we have partitioned the choices by the hash value, and where binomial coefficients with a negative second argument, if they occur, are interpreted as zero. Therefore, for each hash value of Hamming weight i, there are $\binom{n_c-h}{d+1-i}$ choices of $d+1$ among the n_c columns of $\boldsymbol{M}$ including exactly that hash specified choice of i of the first h columns.

Memory access cost of one field element within a partition. The portion of memory required by the processing cluster corresponding to a hash H of Hamming weight i is of size $2V_i := 2\binom{n_c-h}{d+1-i}\binom{N+b-1}{b}$. This quantity includes all V_i memory locations associated with bidegree $(b,1)$ monomials $\mu u_k c_T$, where $u_k c_T$ is a bilinear monomial from the initial SM system and such that $H_j = 1$ if and only if $j \in \{1..h\} \cap T$, as well as an equal amount of memory for writing output values. The total cost of reading every value of $\boldsymbol{v}$ within such a partition, that is, exactly half of the partition's values, is the product of the number of such values, the square root of memory size in bits and the communication cost for transmitting a field element and an address. This product is $(2\log_2 q)^{1/2}(\log_2 q + \log_2(2V_i \log_2 q))V_i^{3/2}$. Thus, summing this quantity across all $\binom{h}{i}$ hashes H of Hamming weight i for all values of i and dividing by the total size, $2V = 2\binom{N+b-1}{b}\binom{n_c}{d}$, of memory, we find that the average memory access cost of a field element within some memory partition is given by

$$\psi_1 = \frac{C_2(\log_2 q)^{1/2}}{2^{1/2}V} \sum_{i=d+h+1-n_c}^{h} \binom{h}{i}(\log_2 q + \log_2(2V_i \log_2 q))V_i^{3/2}. \tag{15}$$

Additional costs due to the hashing strategy. Still, this hash-based management scheme creates overhead that must be taken into account in the final cost. For any given row of the Macaulay matrix, there is a $(d+1)/n_c$ probability that the i-th column of $\boldsymbol{M}$ is used. Therefore, we expect the average vector-vector product to require $h(d+1)/n_c$ processing clusters to locally add products and then transmit to the designated accumulator. We further note that all of the processing clusters can transmit all of their partial sums of size $\log_2 q$ for every

equation to the designated accumulator in a canonical order, removing the need for the transmission of an address of size $\log_2(2V)$ between the processing cluster and the accumulator for every equation. Therefore, dividing by $N(d+1)$, which is the number of monomials in any equation, we compute the average overhead incurred by using the hash strategy per multiplication to be

$$\psi_2 = C_2 \frac{h \log_2 q}{n_c N} \sqrt{2V \log_2 q}, \tag{16}$$

where C_2 is the constant from the two-dimensional model as in Equation (13).

Total cost per $\mathbb{F}_q$-multiplication. Putting these pieces together, we compute the memory access cost per multiplication for solving a generic SM system with the Wiedemann algorithm as follows. Since each equation belongs to a set of, on average, n'_r equations having the exact same $N(d+1)$ monomials, and noting that even for very large SM systems the quantity $N(d+1)\log_2 q$ is small, each processing cluster may retrieve these values only once and store them in its local cache while computing each of the n'_r partial sums. Thus, each field element is accessed $\rho := 1/n'_r$ times on average per multiplication by a processing cluster. Multiplying this average number of accesses by the average access cost ψ_1 within that partition and adding the above computed overhead ψ_2, we obtain

$$\text{Total Memory Cost Per Multiplication} = \rho\psi_1 + \psi_2. \tag{17}$$

Recall that the validity of Equations (15) and (16) depends on the acceptance of a two-dimensional nearest-neighbor topology being optimal for large scale memory. If we prefer a three-dimensional nearest-neighbor topology of a similar nature, the above formulae still work when each exponent of $1/2$ is replaced by $1/3$, $3/2$ is replaced by $4/3$ and C_2 is replaced by a new constant C_3.

Computing Addresses in Canonical Order pb: I was wondering if this should be put in Appendix as well In order to avoid sending an address between the processing clusters and the designated accumulator with each partial sum, note that the cluster and accumulator must locally compute a canonical order for the rows of $\boldsymbol{M}(\mathcal{Q}_b)'$, which may represent an additional cost. However, this cost is not taken into account in Equation (17) as we consider it negligible for all parameter sets from GeMSS or Rainbow. Still, there are some subtleties worth discussing.

An easy and naive way of canonically ordering the rows of $\boldsymbol{M}(\mathcal{Q}_b)$ is to associate to any equation $\mu Q_{i,J}$ the triple (μ, i, J) and then to adopt a lexical order on these triples. The challenging part is then to deterministically compute a pseudorandom subset of these equations to be included in $\boldsymbol{M}(\mathcal{Q}_b)'$. A method which almost solves this problem is to have each processing cluster step through, in canonical order all the rows of $\boldsymbol{M}(\mathcal{Q}_b)$ which have support in its partition and use a PRF like AES CMAC to generate a pseudorandom number from a random seed fixed at the beginning of the attack and (μ, i, J), and use that

number to deterministically decide whether to include the row (μ, i, J). The same computation would be duplicated at the accumulator to determine the same subset of rows of $\boldsymbol{M}(\mathcal{Q}_b)$. However, if the probability of keeping a row is simply fixed to the ratio of V to the number R_b of rows in $\boldsymbol{M}(\mathcal{Q}_b)$, then the exact number of rows in $\boldsymbol{M}(\mathcal{Q}_b)'$ can seen as the outcome of a binomial distribution of parameters R_b and $p := V/R_b$. Therefore, this number will typically be too low or two high compared to V by a term of the order of $\sqrt{V}$.

A way to guarantee that $\boldsymbol{M}(\mathcal{Q}_b)'$ has exactly V rows is to use a slightly higher probability p of computing each partial sum at the processing clusters. The designated accumulator would then repeat the same computation to determine which rows to expect partial sums for, however it would pseudorandomly reject a small fraction of these rows in order to adjust the number of rows in $\boldsymbol{M}(\mathcal{P})'$ down to V. pb: The following text should explain clearly the choice of p right ? Please take your time to make it rigorous before sending it to us. It seems that you are always adding new elements ... ray: Update: I think I know how to prove that overall success probability is $O(1)$ if the acceptance probability at the processing clusters for row s out of r exceeds V/r by $O\left(\frac{\sqrt{log(r)(r-s)V(r-V)/r^2}}{r-s}\right)$. I think the $log(r)$ can be replaced by a constant, which would make the total number of extra partial sums computed $O(\sqrt{V})$ as advertised. Even with the $\sqrt{r}$, it's still $O(\sqrt{V log r})$, which isn't very different for the parameters we care about.

The process by which the designated accumulator adjusts its probability of rejection requires a small amount of state. In particular it must keep track of the value of (μ, i, J) for the most recently processed row, and how many rows up to that point have been included in $\boldsymbol{M}(\mathcal{P})'$. For no parameter set under consideration does this amount of state exceed a few hundred bits. Assuming the designated accumulator does not change very often the cost of transmitting this state will be negligible. This can be arranged, for example if the processing clusters are given a canonical order, and the first cluster is designated for as many rows of $\boldsymbol{M}(\mathcal{P})'$ as there are addresses to be written in its partition, and then the second cluster is designated for the next consecutive block of rows, and so on.

The cost of this process for each transmitted partial sum is negligibly more than the cost of the PRF multiplied by the number of rows in $\boldsymbol{M}(\mathcal{P})$ divided by V. By our calculations, if the PRF costs something like the 2^{15} bit operations required to compute an AES encryption, the cost of finding the next row in the canonical order is at least 1000 times smaller than the costs we analyze for all parameter sets of Rainbow we consider.

pb: I would prefer to take a unique parameter set and detail the computation for this one even if it is very simple. It is more convincing.

For GeMSS, we use $b = 1$ in our analysis, and therefore the costs of canonical ordering are much smaller than we analyze here – this is because, when $b = 1$ almost all the rows of $\boldsymbol{M}(\mathcal{P})$, so the processing clusters do not need to do any rejection sampling and can simply send partial sums for all rows.

7.3 Memory cost for GeMSS and Rainbow

It was already clear from Section 5.3 that memory is not much of a concern in attacking GeMSS; however, the hashing strategy still offers a significant improvement in the two-dimensional nearest neighbor model. The formula of [36] would imply that every multiplication for GeMSS128 should have a cost of $2^{12.68}$, whereas with our hash methodology, calculation shows that the average memory access cost for a multiplication is only $2^{0.58}$; thus, memory costs for GeMSS are insignificant.

The cost of memory access for Rainbow, however, is much more significant. Recall that the "MinRank + $\mathcal{P} = 0$" version of the rectangular MinRank attack of [7] considers the hybrid system $\mathcal{H}$ which combines the SM equations $\mathcal{Q}$ with the m public equations $\mathcal{P}(\boldsymbol{y}) = 0$ which are quadratic in the linear variables present in $\mathcal{Q}$. In this case, the Macaulay matrix $\boldsymbol{M}(\mathcal{H}_b)$ in bi-degree $(b, 1)$ is obtained by taking the rows of $\boldsymbol{M}(\mathcal{Q}_b)$ together with these bi-degree $(2, 0)$ public equations p_i for $1 \leq i \leq m$ multiplied by all degree $(b-2, 1)$ monomials ν. The effect of adding $\mathcal{P} = 0$ to SM is that for a fixed number of columns n_c of $\boldsymbol{M}$, the resulting hybrid system may be solved at a smaller degree b than the initial Support-Minors one. To be convinced of this, note that since the MinRank instance described in [7] and Section 2.3 has a unique solution, the system $\mathcal{H}$ can be solved in degree $(b, 1)$ with any subset of the SM and $\mathcal{P} = 0$ equations of rank $V = \binom{N+b-1}{b}\binom{n_c}{d} - 1$. In particular, since both b and n_c are parameters of the SM equations, it is possible to construct an augmented SM system s.t. $\operatorname{rank}(\boldsymbol{M}(\mathcal{Q}_b)) < V$ while $\operatorname{rank}(\boldsymbol{M}(\mathcal{H}_b)) = V$. Indeed, the rank $R_{SM,b}$ of $\boldsymbol{M}(\mathcal{Q}_b)$ is given by [3, Eq. (20)] when this quantity is smaller than V, and therefore $\boldsymbol{M}(\mathcal{H}_b)$ is of full rank if there exist at least $V - R_{SM,b}$ linearly independent equations in bi-degree $(b, 1)$ derived from $\mathcal{P} = 0$. In practice, as is found in [7, Table 6], optimization of this attack often occurs at a lower value of n_c and a higher value of b than when considering the SM system alone.

Adapting the approach to the $\mathcal{P} = 0$ equations. To take these augmented $\mathcal{P} = 0$ equations into account in our hashing methodology, a first remark is that they trivially come in groups of size m the form $\{\nu p_i,\ 1 \leq i \leq m\}$ with the same monomial content, a set of $\binom{N+1}{2}$ monomials all involving the unique minor variable which divides ν, where we set $N := n - o_2 + 1$ and $d := o_2$ to stick to the notation from Section 3. This structure implies that any vector-vector product $\boldsymbol{r}\boldsymbol{v}^\mathsf{T}$ where $\boldsymbol{r}$ corresponds to a $\mathcal{P} = 0$ equation can be computed by a single processing cluster under the strategy we outlined in Section 7.1. Having at most one processing cluster required to compute the vector-vector product and having at most one long distance transmission of the sum to the designated processor that writes the value in memory, the $\mathcal{P} = 0$ equations are much more efficient than the SM equations, even if they contain many more monomials per equation, as $\binom{N+1}{2} > N(d+1)$ for most parameters.

Another important remark is that any basis of the rowspace of $\boldsymbol{M}(\mathcal{H}_b)$ can be made to contain all of the $\mathcal{P} = 0$ equations. An informal justification for this claim is that for all practical parameters, the first fall degree of the polynomial

system $\mathcal{P}$— which was extensively tested in direct attacks on UOV/Rainbow — is significantly higher than the solving degree of the SM system using the $(b, 1)$ XL strategy. Thus, the collection of $R_{P,b} = m\binom{N+b-3}{b-2}\binom{n_c}{d}$ equations derived from $\mathcal{P} = 0$ are already linearly independent. Therefore, we may take advantage of the efficiency of the $\mathcal{P} = 0$ equations by including them all in $\boldsymbol{M}(\mathcal{H}_b)'$ and randomly adding a subset of size $V - R_{P,b}$ among the SM equations to produce the square submatrix given as input to Wiedemann.

Overall costs. Naturally, using fewer of the SM equations requires a recalculation of the average number n_r' of equations included in the system from among each block of equations with the exact same monomial content. With the above strategy, we have

$$n_r' = \frac{V - R_{P,b}}{\#\text{blocks}},$$

and the value of ψ_1 from Equation (15) is adjusted accordingly. Thus we may compute the total cost of the hybrid attack against Rainbow. Let σ_{SM} denote the ratio $(V - R_{P,b})/V$ of SM equations to total equations in the hybrid system corresponding to $\boldsymbol{M}(\mathcal{H}_b)'$ and let $\sigma_P = R_{P,b}/V$ represent the ratio of $\mathcal{P} = 0$ equations. Then, the total cost under the same assumptions on memory cost of [36] and using the method described in Section 7.1 is given by

$$3(\rho\psi_1 + \psi_2)V^2\sigma_{\text{SM}}N(d+1) + 3\left(\frac{\psi_1}{m} + C_2 \log_2 q(2V \log_2 q)^{1/2}\right) V^2\sigma_P\binom{N+1}{2},$$

where we recall that $N = n - o_2 + 1$ and where ψ_1 is defined in Equation (15). Since $R_{P,b}$ is significantly smaller than $V - R_{P,b}$ and the $\mathcal{P} = 0$ equations are much more memory efficient than the SM equations, we find that the contribution of the $\mathcal{P} = 0$ equations in complexity ends up being a negligible fraction of the total cost for all the parameters we consider. Finally, we present the total cost of the rectangular MinRank attack using the hash method in the 2-dimensional case in Table 6, and we compare it to the conjectured formula given by (14).

8 Conclusion

The Support-Minors modeling of the MinRank problem [3] has changed our perspective of the applicability of rank methods in cryptanalysis. This new technique has changed the complexity of many MinRank instances by a significant amount *in the exponent*. In addition, this advance has opened up new avenues for cryptanalysis by making newly discovered attacks that exploit rank viable, e.g. [7,35].

Some recent work has suggested that it may be possible to avoid the wrath of Support-Minors. In particular, [31] and [36] claim to support protection from the Support-Minors method, the former by a modification of G*e*MSS, and the latter by considering memory costs.

Table 6. Optimal hash size (h) and total attack cost for the MinRank (SM) and "MinRank + $\mathcal{P} = 0$" (SM+$\mathcal{P}$) attacks in the 2D nearest-neighbor topology model for Rainbow variants compared with the conjectured bound (2D Conj.) of Formula (14) and the required security level. The modeling is done with the constant $C = 2^{-5}$.

Scheme (q, n, m, d)		2D SM	2D SM+$\mathcal{P}$	2D Conj.	Security Level
Rainbow-I $(16, 100, 64, 32)$	cost (hash)	$2^{145.39}$ $(h = 18)$	$2^{138.89}$ $(h = 11)$	$2^{135.76}$	2^{143}
Rainbow-III $(256, 148, 80, 48)$	cost (hash)	$2^{206.11}$ $(h = 23)$	$2^{201.09}$ $(h = 19)$	$2^{197.39}$	2^{207}
Rainbow-V $(256, 196, 100, 64)$	cost (hash)	$2^{272.78}$ $(h = 30)$	$2^{260.76}$ $(h = 22)$	$2^{256.89}$	2^{272}

In this work, we have shown that Support-Minors still renders these schemes insecure. By demonstrating a technique for solving a Support-Minors MinRank instance with solutions in an extension field, we have shown that both GeMSS and pHFEv- remain insecure for all practical parameters. Even considering the cost of memory access, we have shown that these schemes as well as the Round 3 parameters of Rainbow fail to achieve the proper security level.

References

1. Albrecht, M.R., Bernstein, D.J., Chou, T., Cid, C., Gilcher, J., Lange, T., Maram, V., Maurich, I.v., Misoczki, R., Niederhagen, R., Paterson, K.G., Persichetti, E., Peters, C., Schwabe, P., Sendrier, N., Szefer, J., Tjhai, C.J., Tomlinson, M., Wang, W.: Classic McEliece: Round 3 (2020), https://classic.mceliece.org/nist/mceliece-20201010.pdf, last accessed on Sep. 10, 2021.
2. Bardet, M., Briaud, P.: An algebraic approach to the rank support learning problem. In: Cheon, J.H., Tillich, J.P. (eds.) Post-Quantum Cryptography. pp. 442–462. Springer International Publishing, Cham (2021)
3. Bardet, M., Bros, M., Cabarcas, D., Gaborit, P., Perlner, R., Smith-Tone, D., Tillich, J.P., Verbel, J.: Improvements of algebraic attacks for solving the rank decoding and MinRank problems. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020. pp. 507–536. Springer International Publishing, Cham (2020)
4. Bardet, M., Mora, R., Tillich, J.P.: Decoding Reed-Solomon codes by solving a bilinear system with a Gröbner basis approach. In: 2021 IEEE International Symposium on Information Theory (ISIT). pp. 872–877 (2021)
5. Bernstein, D.J., Brumley, B.B., Chen, M.S., Chuengsatiansup, C., Lange, T., Marotzke, A., Peng, B.Y., Tuveri, N., Vredendaal, C.v., Yang, B.Y.: NTRU Prime: Round 3 (2020), https://ntruprime.cr.yp.to/nist/ntruprime-20201007.pdf, last accessed on Sep. 26, 2021.

6. Bettale, L., Faugère, J.C., Perret, L.: Cryptanalysis of HFE, multi-HFE and variants for odd and even characteristic. Designs, Codes and Cryptography **69**(1), 1–52 (2013)
7. Beullens, W.: Improved cryptanalysis of UOV and Rainbow. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021. pp. 348–373. Springer International Publishing, Cham (2021)
8. Billet, O., Gilbert, H.: Cryptanalysis of Rainbow. In: De Prisco, R., Yung, M. (eds.) Security and Cryptography for Networks. pp. 336–347. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
9. Buss, J.F., Frandsen, G.S., Shallit, J.O.: The computational complexity of some problems of linear algebra. J. Comput. System Sci. **58**(3), 572–596 (Jun 1999)
10. Casanova, A., Faugère, J.C., Macario-Rat, G., Patarin, J., Perret, L., Ryckeghem, J.: GeMSS: A great multivariate short signature. NIST CSRC (2020), https://www-polsys.lip6.fr/Links/NIST/GeMSS_specification_round2.pdf
11. Chen, M.S., Yang, B.Y., Smith-Tone, D.: PFLASH - secure asymmetric signatures on smart cards. Lightweight Cryptography Workshop 2015 (2015), http://csrc.nist.gov/groups/ST/lwc-workshop2015/papers/session3-smith-tone-paper.pdf
12. Chen, Cong and Danba, Oussama and Hoffstein, Jeffrey and Hülsing, Andreas and Rijneveld, Joost and Schanck, John M. and Schwabe, Peter and Whyte, William and Zhang, Zhenfei: NTRU: Round 3 (2019), https://ntru.org/f/ntru-20190330.pdf
13. Cheng, C.M., Chou, T., Niederhagen, R., Yang, B.Y.: Solving quadratic equations with XL on parallel architectures. In: Prouff, E., Schaumont, P. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2012. pp. 356–373. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
14. Coppersmith, D.: Solving homogeneous linear equations over GF(2) via block wiedemann algorithm. Mathematics of Computation **62**(205), 333–350 (1994)
15. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: EUROCRYPT (2000)
16. Courtois, N.T.: Efficient zero-knowledge authentication based on a linear algebra problem minrank. In: Boyd, C. (ed.) Advances in Cryptology — ASIACRYPT 2001. pp. 402–421. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
17. Ding, J., Chen, M.S., Petzoldt, A., Schmidt, D.: Gui. NIST CSRC (2017), https://csrc.nist.gov/Projects/post-quantum-cryptography/Round-1-Submissions
18. Ding, J., Chen, M.S., Petzoldt, A., Schmidt, D., Yang, B.Y.: Rainbow. NIST CSRC (2020), https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions
19. Ding, J., Petzoldt, A., Wang, L.c.: The cubic simple matrix encryption scheme. In: Mosca, M. (ed.) Post-Quantum Cryptography. pp. 76–87. Springer International Publishing, Cham (2014)
20. Dubois, V., Fouque, P.A., Shamir, A., Stern, J.: Practical cryptanalysis of SFLASH. In: Menezes, A. (ed.) Advances in Cryptology - CRYPTO 2007. pp. 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
21. Faugère, J.C., Safey El Din, M., Spaenlehauer, P.J.: Computing loci of rank defects of linear matrices using Gröbner bases and applications to cryptology. In: ISSAC 2010 - 35th International Symposium on Symbolic and Algebraic Computation. pp. 257–264. ACM, Munich, Germany (Jul 2010), https://hal.archives-ouvertes.fr/hal-01057840
22. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra **139**, 61–88 (1999)

23. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). ISSAC 2002, ACM Press pp. 75–83 (2002)
24. Goubin, L., Courtois, N.T.: Cryptanalysis of the TTM cryptosystem, pp. 44–57. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
25. Jiang, X., Ding, J., Hu, L.: Kipnis-Shamir attack on HFE revisited. In: Pei, D., Yung, M., Lin, D., Wu, C. (eds.) Information Security and Cryptology. pp. 399–411. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
26. Kaltofen, E.: Analysis of Coppersmith's Block Wiedemann Algorithm for the Parallel Solution of Sparse Linear Systems. Mathematics of Computation **64**(210), 777–806 (1995), http://www.jstor.org/stable/2153451
27. Kipnis, A., Shamir, A.: Cryptanalysis of the HFE public key cryptosystem by relinearization. In: Wiener, M. (ed.) Advances in Cryptology – CRYPTO 99. pp. 19–30. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
28. Longa, P., Wang, W., Szefer, J.: The cost to break SIKE: A comparative hardware-based analysis with AES and SHA-3. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021. pp. 402–431. Springer International Publishing, Cham (2021)
29. Matsumoto, T., Imai, H.: Public quadratic polynominal-tuples for efficient signature-verification and Message-Encryption. In: EUROCRYPT. pp. 419–453 (1988)
30. Niederhagen, R.: Parallel Cryptanalysis. Ph.D. thesis, Eindhoven University of Technology (2012), http://polycephaly.org/thesis/index.shtml
31. Øygarden, M., Smith-Tone, D., Verbel, J.: On the effect of projection on rank attacks in multivariate cryptography. In: Cheon, J.H., Tillich, J.P. (eds.) Post-Quantum Cryptography. pp. 98–113. Springer International Publishing, Cham (2021)
32. Patarin, J.: Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In: EUROCRYPT. pp. 33–48 (1996)
33. Petzoldt, A., Chen, M.S., Yang, B.Y., Tao, C., Ding, J.: Design principles for HFEv-based multivariate signature schemes. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology – ASIACRYPT 2015. pp. 311–334. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
34. Porras, J., Baena, J., Ding, J.: ZHFE, a new multivariate public key encryption scheme. In: Mosca, M. (ed.) Post-Quantum Cryptography. pp. 229–245. Springer International Publishing, Cham (2014)
35. Tao, C., Petzoldt, A., Ding, J.: Efficient key recovery for all HFE signature variants. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021. pp. 70–93. Springer International Publishing, Cham (2021)
36. The Rainbow Team: Response to recent paper by Ward Beullens. https://troll.iis.sinica.edu.tw/by-publ/recent/response-ward.pdf (2020)
37. Vates, J., Smith-Tone, D.: Key recovery attack for all parameters of HFE-. In: Lange, T., Takagi, T. (eds.) Post-Quantum Cryptography. pp. 272–288. Springer International Publishing, Cham (2017)
38. Volker, S.: Gaussian elimination is not optimal. Numerische Mathematik **13**, 354–356 (1969)
39. xxxx xxxx, xxxx xxxx: Sage tool for the GeMSS attack (2021), https://github.com/jbbaena/Attack_on_GeMSS

A Even characteristic.

In this section, we deal with the case when q is a power of 2, and we briefly discuss why our attack is not affected, as the one from [35]. Note that the authors from [35] did not really make this part very explicit. For a quadratic form $Q : V \to \mathbb{K}$, we use the *polar form* $B(\boldsymbol{x}, \boldsymbol{y}) = Q(\boldsymbol{x} + \boldsymbol{y}) - Q(\boldsymbol{x}) - Q(\boldsymbol{y})$ instead of

$$B'(\boldsymbol{x}, \boldsymbol{y}) = \frac{Q(\boldsymbol{x} + \boldsymbol{y}) - Q(\boldsymbol{x}) - Q(\boldsymbol{y})}{2},$$

which does not make sense in characteristic 2. For simplicity, we consider plain HFE and the central map

$$f(X) = \sum_{\substack{i,j \in \mathbb{N} \\ q^i + q^j \leq D}} \alpha_{i,j} X^{q^i + q^j}, \ \alpha_{i,j} \in \mathbb{F}_{q^n},$$

which is seen as a quadratic form in $(X, \dots, X^{q^{n-1}})$. For $0 \leq j \leq n-1$, let $\boldsymbol{F}^{*(j)}$ be the matrix representing the polar form of f^{q^j} and let $\boldsymbol{F} := \boldsymbol{F}^{*(0)}$.

Former MinRank attacks. The attack from [6] looks for a rank $\leq d$ linear combination between the public matrices $\boldsymbol{P}_i$ over $\mathbb{F}_{q^n}$. Keeping the notation from their paper, a solution is given by

$$\sum_{k=1}^{n} \gamma_k \boldsymbol{P}_k = \boldsymbol{W} \boldsymbol{F}^{*(0)} \boldsymbol{W}^{\mathsf{T}} = \boldsymbol{W} \boldsymbol{F} \boldsymbol{W}^{\mathsf{T}},$$

where $\boldsymbol{W} := \boldsymbol{S}\boldsymbol{M}$ and where $(\gamma_1, \dots, \gamma_n)$ is the first column of $\boldsymbol{T}^{-1}\boldsymbol{M} \in \mathbb{F}_{q^n}^{n \times n}$. Another solution is clearly given by $(\gamma_1^q, \dots, \gamma_n^q)$, since

$$\sum_{k=1}^{n} \gamma_k^q \boldsymbol{P}_k = \boldsymbol{W} \boldsymbol{F}^{*(1)} \boldsymbol{W}^{\mathsf{T}}.$$

When the characteristic is equal to 2 and $(\lambda, \mu) \in \mathbb{F}_{q^n}^2$, the only non-zero block in the matrix $\lambda \boldsymbol{F}^{*(0)} + \mu \boldsymbol{F}^{*(1)}$ is skew-symmetric of size $(d+1) \times (d+1)$ with zeroes on the diagonal. If d is even, then $d+1$ is odd and the rank of this block is at most d, so that $\lambda(\gamma_1, \dots, \gamma_n) + \mu(\gamma_1^q, \dots, \gamma_n^q)$ gives another solution to MinRank which does not occur in odd characteristic. This type of solution does not lead to an equivalent key, and in [6, §6.3] it is explained how to deal with this issue. If d is odd however, we still have n solutions to the MinRank problem. Finally, note that the discussion is similar for the folklore MinRank attack from [27].

MinRank by Tao et al. However, the situation is not the same for the new MinRank attack from [35] and a fortiori for our attack. A noticeable difference compared to [6,27] is that the matrices $\boldsymbol{M}_i$'s from the MinRank problem 2

are not skew-symmetric anymore, even if the $\boldsymbol{P}_i$'s are. In particular, using the notation from Equation (2), one has that $\boldsymbol{Z}$ and $\boldsymbol{Z}^{[1]}$ have the same rank $\leq d$, but there is no reason why it should be the case for the sum $\boldsymbol{Z} + \boldsymbol{Z}^{[1]}$ or any linear combination of this kind, even when d is even. Here we see that in a way the low rank matrix $\boldsymbol{Z}$ "involves" all the Frobenius iterates of the central map f, see for instance the proof of [35, Prop. 4], whereas a low rank matrix in [6,27] is typically of the form $\boldsymbol{W}\boldsymbol{F}^{*(i)}\boldsymbol{W}^{\mathsf{T}}$.

B Solving Modeling 2 at degree 3 when $n_{c_T} < n_u$.

In this section, we show how a Gröbner basis for Modeling 2 can be found in degree 3 when $n_{c_T} < n_u$. As already mentioned, we can mostly ensure that $n_{c_T} \geq n_u$ with the parameters of GeMSS, so the proof is essentially for the sake of completeness.

We keep the notation from the proof of Proposition 1. Recall that the linear system defined in Equation (7) expresses the minor variables c_T in terms of the remaining $n_u = n - 1$ linear variables $u_1, \ldots, u_{n-1}$, and it is full rank by Assumption 3. When $n_{c_T} < n_u$, there exists a set $(\gamma_i)_{i=1}^{n_{c_T}}$ of linear variables which can be expressed in terms of these minor variables, and the $n_u - n_{c_T}$ remaining ones are denoted by $(\delta_j)_j$. This means that $\binom{n_u - n_{c_T}+1}{2}$ quadratic monomials will be missing at degree 2, namely the ones of the form $\delta_i\delta_j$. Moreover, as Modeling 2 initially contains a lot more equations than $\binom{n_u+1}{2} - \binom{n_u - n_{c_T}+1}{2}$ which is the possible number of leading monomials of the form $\gamma_i\gamma_j$ or $\gamma_i\delta_j$, we hope to construct an independent set of equations $(f_\mu)_\mu$, where for each possible leading monomial μ we have

$$f_\mu = \mu + \ell_\mu,$$

and where ℓ_μ is a degree 1 polynomial. Now let us show how the missing quadratic monomials $\delta_i\delta_j$ are found at degree 3, which will conclude the proof. For the sake of clarity, we do the reasoning for δ_1^2. For $1 \leq i \leq c_T$, let $\mu_{i,1} := \gamma_i\delta_1$ and let $\mu_{i,2} := \gamma_i\delta_2$. Then, the S-polynomial

$$S(f_{\mu_{i,1}}, f_{\mu_{i,2}}) = \delta_2\ell_{\mu_{i,1}} - \delta_1\ell_{\mu_{i,2}}$$

is a polynomial of degree 2 found in degree 3 during the Gröbner basis computation. It will typically contain δ_1^2 for at least one $1 \leq i \leq c_T$.

C Optimized approach to store the Macaulay matrix of Modeling 1.

In the Optimized approach, we aim at storing the Macaulay matrix in a more efficient way by taking advantage of the structure of the SM system. To present the approach, we consider a MinRank instance with N matrices in $\mathbb{F}_2^{n_r \times n_c}$ and target rank $\leq d$ matrix

$$\boldsymbol{Z} := \sum_{i=1}^{N} u_i \boldsymbol{M}_i \in \mathbb{F}_2[\boldsymbol{u}]^{n_r \times n_c}.$$

The core idea is to divide the Macaulay matrix into $\binom{n_c}{d+1}$ blocks $\mathcal{S}_J$ labelled by the subsets $J \subset \{1..n_c\},\ \#J = d+1$ such that $\mathcal{S}_J$ contains the n_r SM equations $Q_{i,J}$ for $1 \leq i \leq n_r$. We have seen in Fact 2 that all these equations have the same monomials, so that the set of columns potentially allocating nonzero entries are the same for each row in the block. This is the key fact to get a more efficient storage of $\boldsymbol{M}(\mathcal{Q})$. This approach then splits the storage of the $\boldsymbol{M}(\mathcal{Q})$ into four arrays, named V_1, V_2, V_3, and V_4:

V_1: This stores the coefficients of the linear forms which are the entries of $\boldsymbol{Z} \in \mathbb{F}_2[\boldsymbol{u}]^{n_r \times n_c}$. For this we require $N n_r n_c$ bits of memory, and for simplicity we assume that these coefficients are stored as a 2-dimensional array of dimensions $n_r \times (N n_c)$, where the entry in V_1 in position (i, j) stores the coefficient of $x_{(j \bmod N)+1}$ in the linear form $\boldsymbol{Z}_{i,\lceil (j-1)/N \rceil + 1}$.

V_2: This stores the indexes of the nonzero values of $\boldsymbol{M}(\mathcal{Q})$ for each block $\mathcal{S}_J,\ J = \{j_1, \ldots, j_{d+1}\}$. As seen in Fact 2, the potential nonzero coefficients of a given SM equation $Q_{j,J}$ correspond to the monomials $x_i \boldsymbol{c}_{J \setminus j_u}$ for $1 \leq i \leq N$ and $1 \leq u \leq d+1$, and in particular they only depend on J. Thus, V_2 can be implemented as an array of length $\binom{n_c}{d+1}$, where each coordinate is enumerated by a set J and stores the $N(d+1)$ potential nonzero indexes. Hence, we need

$$\binom{n_c}{d+1} \cdot N(d+1) \cdot \log_2 \left(\binom{n_c}{d} N \right) \tag{18}$$

bits of memory to store V_2.

V_3: This indicates the columns of V_1 from which the nonzero coefficients of a given SM equation should be taken. Notice that these indexes are the same for all the equations in one block $\mathcal{S}_J$ since they correspond to the elements of J. Thus, to store such indexes we would need $\binom{n_c}{d+1} \cdot N(d+1) \cdot \log_2 (N n_c))$ bits of memory. So far, the only information missing to be able to read the values of the nonzero coefficients of a given SM equation is the index of the row of V_1 from which they must be read. This is stored in V_4.

V_4: Since we drop several rows of the initial Macaulay matrix so that we end up with a square matrix, we have to keep track of the row of $\mathbf{Z}$ from which a given SM equation comes from. Therefore, V_4 stores the indexes of the corresponding row in $\boldsymbol{Z}$ for the $N\binom{n_c}{d}$ SM equations chosen to construct this square Macaulay matrix. This requires $\binom{n_c}{d} N \cdot \log_2(n_r)$ bits of memory.

Now we explain how the allocations of the vectors $V_1, \ldots, V_4$ fully store the Macaulay matrix. Basically, for a given row of the Macaulay matrix, we show how to get the coordinates and values of the potential nonzero entries by just accessing the memory allocated in V_1, V_2, V_3, and V_4. For the seek of clarity, let

us assume that the coordinates of the vector V_4 are enumerated by elements of the set

$$\left\{(a,b) : 0 \leq a \leq \binom{n_c}{d+1} \text{ and } 1 \leq b \leq n_r\right\}.$$

Then, for a given row (a_0, b_0) we know:

1. The indexes of the coordinates containing the potential nonzero positions by reading the bits in $V_2[a_0]$.
2. The values (which are either 0 or 1) corresponding to indexes given in the previous item by reading the positions given by $V_3[a_0]$ in the vector $V_1[b_0]$.

Finally, we apply this approach to Modeling 1 with $N = n + v$, $n_r = n + v$ and $n_c = 2d + 1$. In this case, one notices that the dominant cost is provided by Equation (18), which reads

$$\tbinom{2d+1}{d+1}(n+v)(d+1)\log_2\left(\tbinom{2d+1}{d}(n+v)\right) = \mathcal{O}\left(dn_u n_{c_T}\log_2(n_{c_T})\right),$$

where $n_u = n - 1 \leq n_{c_T} = \binom{2d+1}{d}$. This leads to the memory storage claimed in Equation (12).