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 Center, 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 [39] and [8] on the Round 3 NIST candidates G*e*MSS and Rainbow respectively. In this paper, we study this SM approach more in depth and we propose a greatly improved attack on G*e*MSS based on this Support-Minors method. Even though G*e*MSS was already affected by [39], our attack affects it even more and makes it completely unfeasible to repair the scheme by simply increasing the size of its parameters or even applying the recent projection technique from [35] whose purpose was to make G*e*MSS immune to [39]. For instance, our attack on the G*e*MSS128 parameter set has estimated time complexity 2^{72}, and repairing the scheme by applying [35] would result in a signature with slower signing time by an impractical factor of 2^{14}. Another contribution is to suggest optimizations that can reduce memory access costs for an XL strategy on a large SM system using the Block-Wiedemann algorithm as subroutine when these costs are a concern. In a memory cost model based on [6], we show that the rectangular MinRank attack from [8] may indeed reduce the security for all Round 3 Rainbow parameter sets below their targeted security strengths, contradicting the lower bound claimed by [40] using the same memory cost model.

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

1 Introduction

The MinRank problem 1 introduced in [11] 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 [19] or strongly related to it, such as [36,38,22].

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 [21], 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 [9,27,8]. 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* [32] and HFE [36], but also more recently of the HFEv- schemes [37] G*e*MSS [12] and Gui [20]. 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.* [4] 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 [4,8,3,39]. When the corresponding MinRank instance has a unique solution, which was the case in rank-based cryptography or Rainbow [4,8], this system can be solved using a variant of the XL algorithm [18]. In particular, this approach benefits from the extreme sparsity of the resulting linear system as one can use the Block-Wiedemann algorithm [16]. 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 [4] 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 [39] conjectured from experiments that the *first degree fall* d_{ff} of their Support-Minors attack on G*e*MSS 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 [39, 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 [5], and if this solving degree were higher than 3 in the case of G*e*MSS, the complexity of the attack in [39] 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 [6,1,14,31] 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 [8], the Rainbow team put forward a statement [40] 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 [25] and F5 [26], 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 [39].

Scheme	Minors [39]	SM (conjectural) [39]	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 [39] which used minors modeling [24]. 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 [39], which relied on the aforementioned degree fall assumption (see Column "SM (conjectural) [39]" in Table 1). In particular, with our attack we can also clearly break the proposed parameters for pHFEv-, which were an attempt by [35] to repair GeMSS in the aftermath of the attacks from [39]. 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 proceeds 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 G*e*MSS is much reduced, the memory access cost also remains limited and it is not an obstacle to perform the attack. In particular, we have been able to perform experiments in the Magma Computer Algebra System [10] for the main step of our attack —linearization on the SM equations— on parameters which are not too far from those of the smallest G*e*MSS instances. Apart from G*e*MSS which is the focus of this work in light of the current situation, the efficiency of our approach also suggests that it might be feasible to use rank attacks in order to solve the 24 year old HFE Challenge 2 [36]. Until now, note that previous unsuccessful attempts [17,33] are direct attacks and not MinRank attacks. More generally, we demonstrate that the Support-Minors method may be used to tackle any MinRank instance with multiple solutions belonging to an extension field as long as one can benefit from this extension field structure and from a quite specific parameter range.

As a second major contribution, we propose and analyze 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 [8] 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 [40] 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 $w \log_2 V / \log_2 q$. Our strategy precisely aims at coming close to this figure, still under the assumption that 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 [40], the rectangular MinRank attack does indeed reduce the security of all round 3 Rainbow parameter sets below their targeted security strengths. To evaluate this memory access cost, we provide a theoretical analysis of both the memory savings and the extra costs which are associated to our strategy. We also examined various costs which were suggested to be possibly significant by [40] such as parallelization costs, and the cost of generating Macaulay matrix coefficients "on the fly", finding that incorporating these costs in our cost model does not affect our conclusion at least in the case of Rainbow. Finally, we want to insist on the fact that our methodology is limited to theoretical arguments. Of course real benchmarks would be greatly appreciated to support our claims, but it is probable that software implementations would not be a good indicator for these memory costs. Indeed, the estimates from [6,40] aim to give relative costs in an ASIC implementation which may be used by an adversary with significant resources. In particular, we believe that providing such an involved implementation is far beyond the scope of this paper.

Along with this paper, we also provide a SageMath notebook [2], 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 also denote by $|\boldsymbol{M}|$ (resp. $|\boldsymbol{M}|_{*,J}$) the determinant of $\boldsymbol{M}$ (resp. $\boldsymbol{M}_{*,J}$). Finally, we use $\#I$ to denote the number of elements of a 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 $g'(\boldsymbol{x}, \boldsymbol{y}) = g(\boldsymbol{x} + \boldsymbol{y}) - g(\boldsymbol{x}) - g(\boldsymbol{y}) + g(0)$ 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 G*e*MSS

G*e*MSS [12] 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 [36] that includes both the Minus and the Vinegar modifiers. In this description we consider that q is an odd prime power. 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 $1 \leq i \leq n - a$.

MinRank Attack on HFEv- from [39]. Tao et al. recently proposed in [39] 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.1. 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.

We refer to [39, Theorem 2] for extra details. Also, note that 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 [39, Alg. 1, 4.]. Then, it is shown in [39, §4.3] how one can efficiently derive an equivalent key and finish the attack. Finally, to keep the same notation as in [39, 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 $0 \le j \le n-1$, 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 [30,28,7,41].

Projection Modifier. The projection modification was introduced in [13] in order to repair the previously broken SFLASH signature scheme [23] and devise the new PFLASH signature scheme. In reaction to the attack on GeMSS from [39], the authors of [35] 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 [35] 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. [35, 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 [35], 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 [39] is just above the security level.

Table 2. GeMSS and pHFEv- parameter sets.

Scheme	q	n	v	D	a	p from [35]
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

2.3 Relevant Material on Rainbow for Section 8.3

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 [8, §7] on this scheme.

Rainbow. For clarity, we adopt the simplified description from [8]. The version of Rainbow submitted to the NIST PQC project is a 2-layered variant of the

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 [8] targets secret vectors $\boldsymbol{y} \in O_2$.

Rectangular MinRank attack. The rectangular MinRank attack by [8] 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 [8], 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 [4], see Section 3. Moreover, [8] 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 [8, Table 6].

3 Support-Minors Modeling (SM)

Support-Minors is an efficient method to model and solve the MinRank problem [4]. It has been used to cryptanalyze MPKC and rank-based cryptosystems [3,4,8]. 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 the support matrix $\boldsymbol{C} \in \mathbb{K}^{d \times n_c}$ 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 $1 \le j \le n_r$. 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 $1 \le j \le n_r$, 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 $1 \le j \le n_r$. 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

$$A_J := \left\{u_i c_{J\setminus j_u} :\ 1 \le u \le d+1,\ 1 \le i \le N\right\}.$$

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, [4] proposes a dedicated XL approach by multiplying the SM equations by monomials in the linear variables. This is typically the case for Rainbow [8] or rank-based cryptography [4,3]. 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 $n' \le n_c$ columns in $\boldsymbol{M}$ (for example the first n' ones), which yields a SM system with $n_r\binom{n'}{d+1}$ equations and $N\binom{n'}{d}$ 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 [4]. 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 [4], 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 $n' \in [d+1, n-a]$ 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 [39]. 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 G*e*MSS 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*

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

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*

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

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

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

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

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

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

The linear equations from Fact 3 are often referred to in the literature as *degree falls* from degree 2 to degree 1, and we denote by $\boldsymbol{M}(\mathcal{L})$ the Macaulay matrix of this linear system $\mathcal{L}$. By considering an ordering on the columns such that $c_T > u_{n+v-1} > \cdots > u_1 > u_{n+v} = 1$, we choose to eliminate first and foremost all the $n_{c_T} := \tbinom{n'}{d} - 1$ minor variables.

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

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

where $\boldsymbol{K} \in \mathbb{F}_q^{(\mathcal{N}_L - n_{c_T}) \times (n+v)}$ is row reduced. Moreover, we have that $\mathcal{N}_L = \tbinom{n'}{d} + v - 1$.

Proof. Let $\boldsymbol{L}$ denote the echelon form of $\boldsymbol{M}(\mathcal{L})$, namely

$$\boldsymbol{L} := \begin{pmatrix} \boldsymbol{N} & * \\ 0 & \boldsymbol{K} \end{pmatrix}, \text{where } \boldsymbol{N} \in \mathbb{F}_q^{n_{c_T} \times n_{c_T}} \text{ and } \boldsymbol{K} \in \mathbb{F}_q^{(\mathcal{N}_L - n_{c_T}) \times (n+v)}.$$

Assume that this matrix is not systematic on its first n_{c_T} rows. On that hypothesis, there is a set of $v_0 \geq \mathcal{N}_L - n_{c_T} + 1 \geq v + 1$ 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, $\boldsymbol{u}\boldsymbol{h}_i^{\mathsf{T}} = 0$, where $\boldsymbol{u} \in \mathbb{F}_{q^n}^{n+v}$ denotes the first row of the matrix $\boldsymbol{U}$ defined in Equation (1). 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. This gives $\boldsymbol{N} = \boldsymbol{I}_{n_{c_T}}$.

For the second part of the proof, the number of rows $\mathcal{N}_L - n_{c_T}$ in $\boldsymbol{K}$ is at least v by Fact 3. Since $\boldsymbol{u}$ is a solution to the MinRank problem, there exists a vector $\boldsymbol{v} \in \mathbb{F}_{q^n}^{n_{c_T}}$ corresponding to the minor variables such that

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

Since the matrix $\boldsymbol{M}(\mathcal{L})$ has its entries in $\mathbb{F}_q$ we obtain n linearly independent vectors in the right kernel, namely

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

This shows that the rank of $\boldsymbol{K}$ is at most $(n+v-n) = v$, so that $\mathcal{N}_L - n_{c_T} = v$. □

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 corresponding to the homogeneous degree 1 parts (by dropping the last column of $\boldsymbol{L}$) is of the form

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

where $\boldsymbol{Y} \in \mathbb{F}_q^{n_{c_T} \times n_u}$, $\boldsymbol{W} \in \mathbb{F}_q^{v \times n_u}$ and $n_u := n - 1$.

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 prove at which degree the computation terminates as long as $n_{c_T} \geq n_u$. The proof relies on Assumption 1 and the following Assumption 2 on the echelon form $\boldsymbol{L}$ from Equation (6).

Assumption 2 *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 and 2. 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 the 2 assumptions are satisfied almost 100% of the times. The reader might find helpful to experimentally explore these assumptions using the SageMath notebook [2].

Proposition 1. *Under Assumptions 1 and 2, 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.

Proof. By Assumption 1 and the first part of Lemma 1, 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

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

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 $\boldsymbol{c}$ the row vector of length n_{c_T} whose components are the minor variables and $(u_1, \ldots, u_{n+v-1}) := (\boldsymbol{u}_+, \boldsymbol{u}_-)$, where $\boldsymbol{u}_+$ is of length n_u (remaining linear variables) and $\boldsymbol{u}_-$ is of length v (removed linear variables). Then, there is a 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 2, 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. □

When $n_{c_T} < n_u$, we conjecture that the Gröbner basis algorithm terminates in degree 3, and heuristic arguments to support this conjecture can be found in Appendix A.2. 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 GeMSS. 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 GeMSS 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 $\omega \approx 2.81$ is the linear algebra constant.

An alternative path is to use Coppersmith's Block-Wiedemann algorithm (BW). Let $\mathcal{M}$ be the rowspace 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 $\mathcal{N}_L$ given by Fact 3. By Lemma 1, we have that $\mathcal{N}_L = \binom{2d+1}{d}+v-1$. 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

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

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

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

Optimized Approach. Here we adapt to the SM equations the strategy used by Niederhagen for a generic Macaulay matrix [34, §4.5.3]. By Fact 2, recall that for a given subset $J \subset \{1..n_c\}$, $\#J = d+1$, all SM equations of the form $Q_{i,J}$ for $1 \leq i \leq n_r$ have the same set of potential nonzero monomials. Hence, the set of columns in $\boldsymbol{M}(\mathcal{Q})$ potentially allocating nonzero entries are the same for each row which correspond to one of these equations.

To store the system $\mathcal{Q}$ we use four arrays, namely V_1, V_2, V_3, and V_4. The array V_1 is implemented as a 2-dimensional array of size $n_r \times (Nn_c)$ in which we store the coordinates of the MinRank input matrices $\boldsymbol{M}_i$'s. The array V_2, instead, stores the monomials of all SM equations. More precisely, for each subset $J \subset \{1..n_c\}$, $\#J = d+1$, we store in V_2 the coordinates corresponding to potential nonzero monomials in SM equations associated to J. Finally, it remains

to store the information about how to read from V_1 the coefficients of a given $Q_{i,J}$ equation. This information is given a list of $N(d+1)$ coordinates in V_1 that belong to the same row. The set of column indexes is stored in V_3, while the row index is stored in V_4. A more detailed description of this storage can be found in Appendix A.3, and overall we obtain a space complexity of

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

which saves a factor of order $n+v$ compared to the Standard approach. This Optimized approach also has better memory access than the Standard approach. Indeed, both approaches require to retrieve the same amount of information, but in the Standard approach the size of the memory is larger. For instance, if one uses the 2-dimensional model of [6,40] stating that retrieving b consecutive bits from a memory of M bits costs

$$2^{-5}(b+\log N)\sqrt{M},$$

where N is the length of the array we are reading from. In the Standard approach, one can check that both $(b+\log N)$ and $\sqrt{M}$ factors are larger in each vector-vector multiplication of the Block-Wiedemann algorithm.

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 BW 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 GeMSS, 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 GeMSS 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 GeMSS and pHFEv- signature schemes. In Table 4, we give the time complexity of our attack on the current GeMSS 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 [42], while a constant of 3 for the concrete complexity of BW [29, Theorem 7]. One can check that for the specific parameters proposed by the GeMSS 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 [35, Table 2] using $\omega = 2.81$. In this paper, the value of p was chosen such that the minors attack from [39] 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 [39] for the GeMSS parameters.

Scheme	Minors [39]	SM [39]	SM Step 1 (Strassen/BW)	SM Step 2 (Strassen)	n'
GeMSS128	139	118	76/72	54	21
BlueGeMSS128	119	99	65/65	51	17
RedGeMSS128	86	72	49/53	45	11
GeMSS192	154	120	78/75	57	21
BlueGeMSS192	132	101	67/67	53	17
RedGeMSS192	95	75	51/55	48	11
GeMSS256	166	121	79/77	59	21
BlueGeMSS256	141	103	68/69	55	17
RedGeMSS256	101	76	52/57	50	11

The nature of our approach, although in theory similar to the one used in [39], allows us to reduce significantly the complexity of the Support-Minors attack performed by Tao et al. against GeMSS. This is important since this improvement makes it completely infeasible to repair GeMSS 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 [39] 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 [39]". Another noticeable difference between our work and the one in [39] 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 [35]. 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 [39] 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 [39] is just above x.

Scheme	p	Minors [39,35]	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 Experiments for Step 1

We have performed experiments in Magma-2.23-8 in order to explore the feasibility of the attack on GeMSS. We focus on Step 1, because its cost dominates the total cost as discussed in Section 5. We measure the running time of this step for larger parameters so that a trend can be observed. For these experiments, we selected $a = v \approx n/10$, a small prime $q > 2$ and $d = \lceil \log_q(D) \rceil \geq 3$. We chose the number of columns n' to be the smallest integer such that $n_{cT} \geq n_u$, *i.e.* $\binom{n'}{d} \geq n$, so the system from Step 2 is solved at degree 2.

Fig. 1 summarizes the results of these experiments. In the graph, the theoretical value is the logarithm in base two of the time complexity given in Equation (8) with $n_u = n - 1$, $n_{c_T} = \binom{n'}{d} - 1$, $\omega = 2.81$ and a hidden constant from the Strassen's algorithm taken equal to 7. The experimental complexity is measured in terms of clock cycles of the CPU given by the Magma command ClockCycles(). The matrix reduction was done via the Magma command GroebnerBasis($\mathcal{Q}$, 2), which is equivalent to Reduce($\mathcal{Q}$) in this context[7], yet more efficient.

Our goal here is to discuss how feasible an attack on GeMSS is. For example, the level I parameter set RedGeMSS128 is $(q, n, v, D, a) = (2, 177, 15, 17, 15)$, so that $d = 5$. According to our estimates its complexity is upper bounded by 2^{49}, as shown in Table 4. For this value of d, we have been able to run experiments up to $n = 160$, which is quite close to the goal of 177. Fig. 1 also shows that the estimated complexity is a good upper bound for the computation's complexity. Note that the jump in the $d = 4$ curves corresponds to a change in the value of n'. Indeed, one can solve the system from Step 2 at degree 2 with $n' = 2d+1 = 9$ as long as $n \leq 126$, and otherwise one has to consider $n' > 2d + 1$, for instance $n' = 2d + 2$ for the rest of the data points in these curves.

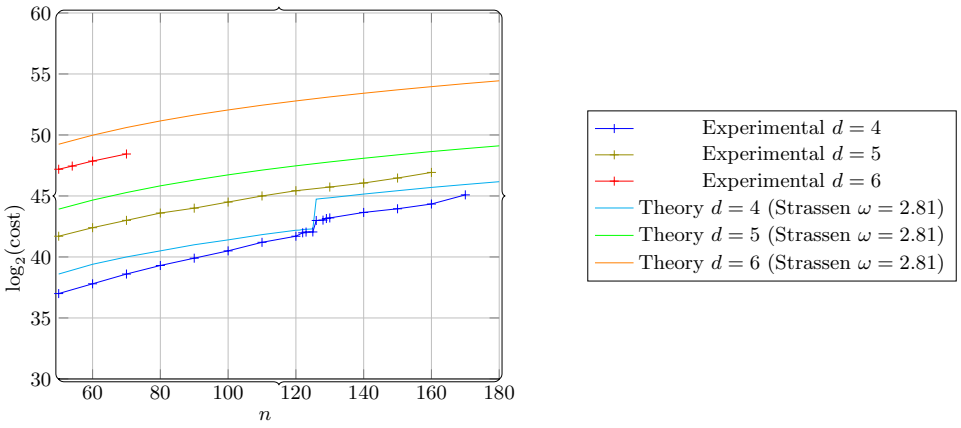

Fig. 1. Experimental vs Theoretical value of the complexity of Step 1.

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

This section is dedicated to the study of the memory complexity associated to the XL strategy on a very large Support-Minors system $\mathcal{Q}_b$ with possibly $b > 1$

[7] The two procedures are equivalent because the system is bilinear, hence quadratic, and Gröbner bases are automatically reduced in Magma.

such as in attacks on rank metric code-based cryptosystems [4] or in the recent rectangular MinRank attack on Rainbow [8]. The core operation here is the use of Block-Wiedemann, whose cost is dominated by the combined cost of a large number of matrix-vector multiplications, where the matrix is a fixed, full-rank, square submatrix $\boldsymbol{M}(\mathcal{Q}_b)'$ of the initial Macaulay matrix $\boldsymbol{M}(\mathcal{Q}_b)$. Note that 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 queries to a large memory. In [40], 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, [40] follows [6] 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 constant $C_2 = 2^{-5}$ is used in [6], [40] and, where we provide concrete numbers, in our paper.

In Section 8.1, we propose a strategy to obviate much of the memory access cost per multiplication of Formula (13). In this 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. First, note that as we have not seen any obvious way to avoid storing $\boldsymbol{v}$ while the value of the matrix-vector product is being written to memory, we assume without harm a memory of size $2V \log_2 q$ instead of $V \log_2 q$. With this choice and assuming the same cost formula for generic RAM access as [40], this implies that the cost of the Wiedemann algorithm should be quite close to

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

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. In Section 8.2, we analyze the memory costs associated to our strategy with this 2-dimensional memory model in mind with the understanding that it is a trivial matter to adjust them to the 3-dimensional model. Finally, we apply our formulae in Section 8.3 to the rectangular MinRank attack [8] to show that at least in this case our results are indeed close to Formula (14).

8.1 Hashing strategy on the main memory

Each coordinate of a matrix-vector product performed within the Wiedemann algorithm is obtained from 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 [40] 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 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, where bit $i \in \{1, \ldots, h\}$ of $H(\mu c_{J\setminus\{j\}})$ is 1 exactly when $i \in J \setminus \{j\}$. 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 [40] and [15], 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.

8.2 Memory savings from our approach

In this section, we analyze the memory savings of our approach compared to a naive XL implementation which does not take advantage of structure within 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 $J \subset \{1..n_c\}$, $\#J = d+1$ 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

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

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 the first h 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 these 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, $V = \binom{N+b-1}{b}\binom{n_c}{d}$, of read memory, we find that the average memory access cost of a field element within some memory partition is given by

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

Additional costs due to the hashing strategy. Still, our 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

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

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.

Neglected Costs There are many costs to consider that arise from the hashing strategy that are either slight or negligible, hence they are not included in Equation (17). Similarly to [40], the first of these costs is the cost of computing the Macaulay matrix $\boldsymbol{M}(\mathcal{Q}_b)$ on the fly. While it is common knowledge that this represents a negligible cost for current parallel implementations of XL, see [15], we must be sure that this task contributes insignificantly to the total complexity with the level of parallelization required by the hash strategy. We verify that the total impact to the complexity is negligible with this strategy in Appendix B.1.

Second, 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 included in Equation (17). Indeed, by using a sufficiently efficient canonicalization method

one can ensure that it is negligible for all current parameter sets from GeMSS and Rainbow. An example of such canonicalization is given in Appendix B.2.

Finally, the above hashing strategy dramatically decreases the memory access cost of running XL on SM systems but requires a significant degree of parallelization. In fact, in order for a cryptographically relevant calculation to finish in a reasonable amount of time, further parallelization is likely desirable. Though the cost of idle processes for attacks at cryptographic scales may not translate to the same monetary cost they do for feasible calculations, it is standard to consider the cost of idle processors in parallelization. Thus, we analyze the extra cost of the hashing strategy incurred by these idle processors. In doing so, we show that our approach to run XL on such SM systems can be seen as "embarrassingly parallel", see Appendix B.3.

8.3 Application to the Rainbow rectangular MinRank attack [8]

In this section, we estimate the memory access cost of our hashing scheme applied to the rectangular MinRank attack [8]. In comparison to our GeMSS attack, the sizes of the SM systems encountered there can be significant. In particular, [40] pointed out that it may be vital to consider memory access costs in this context. Recall from Section 2.3 that the "MinRank + $\mathcal{P} = 0$" version of this attack considers the hybrid system $\mathcal{H}$ combining 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. In general, the system $\mathcal{H}$ can be solved in degree $(b, 1)$ with any subset of the SM and $\mathcal{P} = 0$ equations of rank $V = \binom{N+b-1}{b}\binom{n_c}{d} - 1$. As derived in [8], under standard genericity assumptions such a subset exists at degree $(b, 1)$ when the coefficient of t^b in $(1-t^2)^N G'(t)$ is non-positive, where $G'(t)$ is the generating function for the quotient of the polynomial ring by $\mathcal{Q}$. 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 $\operatorname{rank}(\boldsymbol{M}(\mathcal{H}_b)) = V$. Note that the rank $R_{SM,b}$ of $\boldsymbol{M}(\mathcal{Q}_b)$ is given by [4, Heuristic 2] when this quantity is smaller than V, and therefore $\boldsymbol{M}(\mathcal{H}_b)$ will be 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 [8, 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 of 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 8.1. 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 as many of the $\mathcal{P} = 0$ equations as are linearly independent. First, note that 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. Also, the assumptions which were proposed and empirically verified on pages 22-23 of [8] for the hybrid system are actually stronger than assuming that merely the $\mathcal{P} = 0$ quadratic system is generic. Thus, there seems to be no harm in assuming that with high probability, the number $R_{P,b}$ of linearly independent $\mathcal{P} = 0$ equations can be calculated as $R_{P,b} = \binom{n_c}{d}[t^b]H(t)$, where $H(t) = \frac{1-(1-t^2)^m}{(1-t)^N}$ and where $[t^b]H(t)$ represents the coefficient of t^b in the power series expansion of H. Finally, recall that we have to add as many SM equations as possible to these $\mathcal{P} = 0$ equations in order to reach the final rank V. Since SM equations occur in the span of the augmented $\mathcal{P} = 0$ polynomials, the rank of the Macaulay will not always increase by 1 each time we add a random SM equation. Under the standard heuristic that these equations behave as random vectors in a space of the appropriate dimension, any random subset of $R_{P,b}$ of these $\mathcal{P} = 0$ equations should be linearly independent with probability around $1 - q^{-1}$, and a similar argument under the same heuristic can be used again to verify that this system can be extended to a full rank system with randomly chosen SM equations with a similarly high probability. For clarity, we do not add the factor corresponding to this probability in our estimations as we can treat it as a constant.

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 $\rho := 1/n'_r$ 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 [40] and using the

method described in Section 8.1 is given by

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

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 estimated cost of applying XL using the hash method to the rectangular MinRank attack on Rainbow in the 2-dimensional case in Table 6, and we compare it to the conjectured formula given by (14).

Table 6. Optimal hash size (h) and total attack cost including idle costs 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 using the constant $C_2 = 2^{-5}$.

Scheme (q, n, m, d)		2D SM	2D SM+$\mathcal{P}$	2D Conj.	Security Level
Rainbow-I $(16, 100, 64, 32)$	cost (hash)	$2^{146.9}$ $(h = 12)$	$2^{139.5}$ $(h = 14)$	$2^{135.8}$	2^{143}
Rainbow-III $(256, 148, 80, 48)$	cost (hash)	$2^{205.9}$ $(h = 16)$	$2^{201.2}$ $(h = 16)$	$2^{197.4}$	2^{207}
Rainbow-V $(256, 196, 100, 64)$	cost (hash)	$2^{272.4}$ $(h = 18)$	$2^{260.9}$ $(h = 19)$	$2^{256.9}$	2^{272}

9 Conclusion

The Support-Minors modeling of the MinRank problem [4] 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. [8,39]. This new MinRank algorithm has inspired recent efforts to repair broken schemes, see [35], and work to estimate the real-world complexity of implementing Support-Minors via XL, see [40]. In particular, [35] claims to offer protection from the Support-Minors method by way of a modification of GeMSS

called pHFEv- while [40] offers a first approximation of a memory cost analysis for solving Support-Minors.

In this work, we provide a technique for solving a Support-Minors MinRank instance with solutions in an extension field, verifying that both GeMSS and pHFEv- remain insecure for all practical parameters. Indeed, it turns out that the advantage of using Support-Minors in this scenario is significant and the complexity of the attack is much smaller than that of [39]. The attack is efficient enough so that with more effort it may finally be feasible to practically solve HFE Challenge 2 [17].

Also, with our hashing strategy, we give theoretical arguments of the same level as in [40] to show that much of the memory access cost described there may be obviated when solving large Support-Minors systems using XL. Moreover, while this hash strategy depends intimately on the structure of the Support-Minors system, it does not seem to depend strongly on the solving degree in the XL algorithm. This fact suggests that it may even be possible to fully parallelize XL generically. This task remains an important direction for future work.

Acknowledgements. The Magma experiments of Section 7 were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr). This work was supported by a grant from the Simons Foundation (712530, DCST).

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. Baena, J., Verbel, J.: Sage tool for the GeMSS attack (2021), https://github.com/jbbaena/Attack_on_GeMSS/blob/main/Attack_on_GeMSS.ipynb
3. 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)
4. 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)
5. 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)
6. 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.

7. 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)
8. 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)
9. 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)
10. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. J. Symbolic Comput. **24**(3-4), 235–265 (1997). https://doi.org/10.1006/jsco.1996.0125, computational algebra and number theory (London, 1993)
11. 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)
12. 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
13. 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
14. 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
15. 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)
16. Coppersmith, D.: Solving Homogeneous Linear Equations Over GF(2) via Block Wiedemann Algorithm. Mathematics of Computation **62**(205), 333–350 (1994)
17. Courtois, N.: Algebraic Attacks over GF(2k), Application to HFE Challenge 2 and Sflash-v2. pp. 201–217 (02 2004)
18. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In: EUROCRYPT (2000)
19. 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)
20. Ding, J., Chen, M.S., Petzoldt, A., Schmidt, D.: Gui. NIST CSRC (2017), https://csrc.nist.gov/Projects/post-quantum-cryptography/Round-1-Submissions
21. 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
22. 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)
23. 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)

24. 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
25. Faugère, J.C.: A New Efficient Algorithm for Computing Gröbner bases (F4). Journal of Pure and Applied Algebra **139**, 61–88 (1999)
26. Faugère, J.C.: A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5). ISSAC 2002, ACM Press pp. 75–83 (2002)
27. Goubin, L., Courtois, N.T.: Cryptanalysis of the TTM cryptosystem, pp. 44–57. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
28. 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)
29. 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
30. 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)
31. 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)
32. Matsumoto, T., Imai, H.: Public Quadratic Polynominal-Tuples for Efficient Signature-Verification and Message-Encryption. In: EUROCRYPT. pp. 419–453 (1988)
33. Mohamed, M., Ding, J., Buchmann, J.: Towards Algebraic Cryptanalysis of HFE Challenge 2. vol. 6, pp. 123–131 (08 2011)
34. Niederhagen, R.: Parallel Cryptanalysis. Ph.D. thesis, Eindhoven University of Technology (2012), http://polycephaly.org/thesis/index.shtml
35. 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)
36. Patarin, J.: Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In: EUROCRYPT. pp. 33–48 (1996)
37. 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)
38. 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)
39. 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)
40. The Rainbow Team: Response to recent paper by Ward Beullens. https://troll.iis.sinica.edu.tw/by-publ/recent/response-ward.pdf (2020)

41. 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)
42. Volker, S.: Gaussian Elimination is not Optimal. Numerische Mathematik **13**, 354–356 (1969)

A Supplementary material: additional content on the G*e*MSS attack

A.1 Even characteristic

In this section, we deal with the case when q is a power of 2, and we briefly discuss why both our attack and the one from [39] are not affected. Recall that these two attacks consider the very same MinRank instance (Problem 2). For simplicity, we consider plain HFE and the central map

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

and we further assume that this polynomial f can be seen as a quadratic form in $(X,\dots,X^{q^{n-1}})$. For $0\leq j\leq n-1$, let $\boldsymbol{F}^{*(j)} := \boldsymbol{N}^{*(j)} + \boldsymbol{N}^{*(j)^\mathsf{T}}$, where $\boldsymbol{N}^{*(j)}$ is the upper-triangular matrix representing the quadratic form f^{q^j}. Each of these matrices is symmetric with zeroes on the diagonal. Finally, let $\boldsymbol{F} := \boldsymbol{F}^{*(0)}$.

Former MinRank attacks. The attack from [7] 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 [7, §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 [30].

MinRank by Tao et al. However, the situation is not the same for the new MinRank attack from [39] and a fortiori for our attack. A noticeable difference compared to [7,30] 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 [39, Prop. 4], whereas a low rank matrix in [7,30] is typically of the form $\boldsymbol{W}\boldsymbol{F}^{*(i)}\boldsymbol{W}^{\mathsf{T}}$.

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

In this section, we try to explain why a Gröbner basis for Modeling 2 may be found in degree 3 when $n_{c_T} < n_u$ with high probability. As already mentioned, we can mostly ensure that $n_{c_T} \geq n_u$ with the parameters of GeMSS, so these arguments are essentially for the sake of completeness. In particular, we leave the investigation of this precise probability for future work.

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 2. 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 $1 \leq i \leq c_T$.

A.3 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. This can be stored as an array of size $\binom{n_c}{d+1}$, where each coordinate contain a bit string of length $N(d+1) \cdot \log_2(N n_c)$ 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 sake 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 corresponding to the indexes in $V_2[a_0]$ are obtained by reading in $V_1[b_0]$ the coordinates indicated by $V_3[a_0]$.

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 $n_u = n - 1 \leq n_{c_T} = \binom{2d+1}{d} - 1$.

B Supplementary material: some neglected costs from Section 8

In this section, we provide justification for some extra costs which are not taken into account in our model to estimate the memory complexity of a Support-Minors attack on Rainbow using XL. For that purpose, we give theoretical formulae for these extra costs which turned out negligible compared to the complexity of the whole computation when evaluated on the concrete Rainbow parameters.

B.1 Computing the SM Macaulay Matrix entries on the fly

All of the coefficients of the Macaulay matrix $\boldsymbol{M}(\mathcal{Q}_b)$ are derivable from the $N \times n_r n_c$ coefficients of all the linear forms which are entries of the support matrix $\boldsymbol{M}$. More precisely, the coefficient of $\mu u_i \boldsymbol{c}_{J\setminus j_\ell}$ in the row labelled by $\mu Q_{j,J}$ will be $(-1)^\ell$ times the coefficient of u_i in $\boldsymbol{M}_{j,j_\ell}$ (see Fact 2). As such, each processor can compute all relevant coefficients of the Macaulay matrix by accessing (once per coefficient) a memory of size only $Nn_r n_c \log_2(q)$ bits. This can either be its own local memory or in the case of extreme parallelism a memory shared with a handful of other processors. Access to this memory is sufficiently sequential that we can ignore the cost of sending addresses in each memory query.

With this in mind, let us look at the work a processor would do in order to compute the, on average, n'_r partial sums (since we work with a square submatrix $\boldsymbol{M}(\mathcal{Q}_b)'$ with $n'_r < n_r$ rows), indexed by j, in a block of equations with shared memory access patterns. The processor would need to step through the possible values of i and ℓ and compute the addresses of elements of $\boldsymbol{v}$ to be multiplied by the potentially nonzero coefficients of each equation. These addresses do not depend on j so they only need to be computed once per block of equations. The work involved in computing an address consists of multiplying a degree 1 monomial by a degree $b-1$ monomial, which can be modeled as inserting an element into an ordered list of b elements of size $\log_2(N)$, and removing an element from a list of $d+1$ elements of size $\log_2(n_c)$. Vector elements from these locations would then be cached and for each i, ℓ, and j, $(-1)^\ell$ times the coefficient of u_i in $\boldsymbol{M}_{j,j_\ell}$ would be multiplied by the corresponding cached vector element and all products sharing a j would be summed together to produce the partial sum for j. The costs associated with computing coefficients of the Macaulay matrix on the fly are then, per field multiplication, approximately:

$$\log_2(q) \cdot C_2 \sqrt{N n_r n_c \cdot \log_2(q)} + \frac{1}{n'_r} \left(b \cdot \log_2(N) + d \cdot \log_2(n_c) \right).$$

These costs were neglected in our analysis since they turned out negligible when computed on the concrete Rainbow parameters. For example, when considering the SM $+\mathcal{P}$ attack on Rainbow I, the costs come to about 2^7 bit-operation equivalents per field multiplication, which is more than 1000 times smaller than the costs we explicitly accounted for.

B.2 Canonicalization method for submatrix $\boldsymbol{M}(\mathcal{Q}_b)'$.

Let N_b denote the number of rows in the Macaulay matrix $\boldsymbol{M}(\mathcal{Q}_b)$, where $\mathcal{Q}$ is the Support-Minors system. An easy way of canonically ordering these N_b rows 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. We may then refer to such a triple by an index s taking values between 0 and $N_b - 1$. The challenging part is actually deterministically computing a pseudorandom subset of these equations to be included in the submatrix $\boldsymbol{M}(\mathcal{Q}_b)'$.

This task can be accomplished in a distributed manner by simply sharing a seed for a PRF among all of the processing clusters from our memory model and having all processing clusters keep a running count of ν_s, the number of rows of index $< s$ that have been chosen to be included in $\boldsymbol{M}(\mathcal{Q}_b)'$. These processing clusters then use the PRF to determine with probability $\frac{V-\nu_s}{N_b-s}$ whether to include row s in $\boldsymbol{M}(\mathcal{Q}_b)'$. The same rows are generated from the same seed in the same way every time a matrix vector multiplication is performed. Note that even with 2^h processing clusters each running a PRF costing as much as AES-128 (about 2^{15} bit-operations) a total of $3V \times N_b$ times, the extra cost of this simple canonicalization method is only a small percentage of the cost of the total attack for any of the parameter sets we are considering. For example, in the rectangular

MinRank attack on Rainbow-I, the cost added by canonicalization with the above method is $2^{139.1}$, which would only increase the complexity of the attack from $2^{139.4}$ to $2^{140.3}$. Thus, even the most obvious method of canonicalization is effective. Still, this section aims at providing a way of making this extra cost truly negligible.

Proposed canonicalization. The idea in our approach is that only the designated accumulator really needs to keep a running count of the number ν_s of rows that have been chosen to be included in $\boldsymbol{M}(\mathcal{Q}_b)'$. For row s, each processing cluster may then send partial sums with a different acceptance probability p_s that does not depend on ν_s. As long as this acceptance probability is greater than $\frac{V-\nu_s}{N_b-s}$ for all s with probability $1-o(1)$, the designated accumulator can, with probability $1-o(1)$, correct the probability of the full sum at row s being computed down to $\frac{V-\nu_s}{N_b-s}$ for all rows via rejection. In this procedure, only the designated accumulator needs to keep track of ν_s by computing a PRF for each s value. The other processing clusters will only need to compute the PRF for s values corresponding to rows where the processing cluster in question may need to compute a partial sum, depending on the PRF output. Recall that computing a full sum for a row of the Macaulay matrix requires on average $h\frac{d+1}{n_c}$ partial sums from processing clusters. Therefore, for each matrix-vector product, the PRF will be computed, on average, $1+h\frac{d+1}{n_c}$ times for each of the N_b rows of $\boldsymbol{M}(\mathcal{Q}_b)$. The only thing remaining is to determine a suitable value of p_s and to determine the added cost of the PRF evaluations and the extra partial sums being sent to the accumulator. We provide a suitable answer to the first question in Lemma 2 and the answer to the second question immediately follows, see Lemma 3.

For $0 \leq i \leq N_b-1$, let X_i be the binary random variable modeling the choice of row i to be included in $\boldsymbol{M}(\mathcal{Q}_b)'$, so that $\nu_s = \sum_{i=0}^{s-1} X_i$. By construction we have

$$\mathrm{E}(X_s|\nu_s) = \frac{V-\nu_s}{N_b-s}, \tag{19}$$

and one can prove that ν_s follows the hypergeometric law with parameters N_b (population size), V (number of successes) and s (number of trials), say $\nu_s \sim \mathcal{H}(s, V/N_b, N_b)$. By symmetries in the probability density function of a hypergeometric random variable, one also has $V-\nu_s \sim \mathcal{H}(N_b-s, V/N_b, N_b)$.

Lemma 2 *For $0 < \varepsilon < 1$, the inequality $p_{s,\varepsilon} > \dfrac{V-\nu_s}{N_b-s}$ holds for all $s \in \{0,\ldots,N_b-1\}$ with probability $\geq 1-\varepsilon$, where*

$$p_{s,\varepsilon} := \min\left(1, \frac{V}{N_b} + \frac{\alpha_{s,\varepsilon}}{N_b-s}\right),$$

and

$$\alpha_{s,\varepsilon} := \sqrt{\frac{(N_b-s)\ln(N_b/\varepsilon)}{2}}. \tag{20}$$

Proof. We do the proof for an s such that $V/N_b + \alpha_{s,\varepsilon}/(N_b - s) < 1$, otherwise $p_{s,\varepsilon} = 1$ and the result is clear. Let $Y_{N_b-s} = V - \nu_s$. We may provide a lower bound for $\mathcal{P}\left(p_{s,\varepsilon} > \frac{Y_{N_b-s}}{N_b - s}\right)$ by providing an upper bound for

$$\mathcal{P}\left(p_{s,\varepsilon} \leq \frac{Y_{N_b-s}}{N_b - s}\right) = \mathcal{P}\left(Y_{N_b-s} \geq \mathrm{E}[Y_{N_b-s}] + \alpha_{s,\varepsilon}\right).$$

Using the naïve tail bound $\mathcal{P}\left(Y \geq pn + tn\right) \leq e^{-2t^2 n}$ for $Y \sim \mathcal{H}(N, p, n)$ and $t > 0$, here with $Y = Y_{N_b-s}$, we obtain:

$$\mathcal{P}\left(Y_{N_b-s} \geq \mathrm{E}[Y_{N_b-s}] + (N_b - s)t\right) \leq e^{-2t^2(N_b-s)}.$$

Finally, taking $t = \frac{\alpha_{s,\varepsilon}}{N_b - s}$ and using Equation (20), we get

$$\mathcal{P}\left(p_{s,\varepsilon} > \frac{Y_{N_b-s}}{N_b - s}\right) \leq e^{-2\frac{\alpha_{s,\varepsilon}^2}{N_b-s}} = \frac{\varepsilon}{N_b}.$$

We can now apply the union bound to show

$$\mathcal{P}\left(\forall s \in \{0, \ldots, N_b - 1\}, p_{s,\varepsilon} > \frac{Y_{N_b-s}}{N_b - s}\right) \leq \sum_{s=0}^{N_b-1} \mathcal{P}\left(p_{s,\varepsilon} > \frac{Y_{N_b-s}}{N_b - s}\right) \leq \varepsilon.$$

□

Lemma 3 *For $0 < \varepsilon < 1$ and $0 \leq s \leq N_b - 1$, let $p_{s,\varepsilon}$ as defined in Lemma 2 and let μ_ε denote the average value of $p_{s,\varepsilon} - V/N_b$ over all values of s. We have*

$$0 < \mu_\varepsilon \leq \sqrt{\frac{2\ln(N_b/\varepsilon)}{N_b}}. \tag{21}$$

Proof. We have

$$\mu_\varepsilon = \frac{1}{N_b}\sum_{s=0}^{N_b-1}\frac{\alpha_{s,\varepsilon}}{N_b - s} = \frac{1}{N_b}\sum_{s=1}^{N_b}\frac{\alpha_{N_b-s,\varepsilon}}{s} = \frac{\sqrt{\ln(N_b/\varepsilon)}}{\sqrt{2}N_b}\sum_{s=1}^{N_b}\frac{1}{\sqrt{s}}.$$

Therefore $\mu_\varepsilon \leq \frac{\sqrt{\ln(N_b/\varepsilon)}}{\sqrt{2}N_b}\int_0^{N_b}\frac{1}{\sqrt{s}}ds = \sqrt{\frac{2\ln(N_b/\varepsilon)}{N_b}}$. □

Using Lemma 3 instantiated with $\varepsilon = N_b^{-1}$, we thus expect an average probability of no more than $\frac{V}{N_b} + 2\sqrt{\frac{\ln N_b}{N_b}}$ that a partial sum is transmitted. We therefore conclude that the overhead from sending extra partial sums is no more than a factor of $\left(1 + \frac{2\sqrt{N_b \ln N_b}}{V}\right)$. This is clearly negligible for all of the Rainbow parameter sets under consideration, for example in the SM+$\mathcal{P}$ hybrid attack on Rainbow-I, this factor is approximately $1 + 2^{-23}$. Finally, with the proposed

canonicalization the PRF only needs to be performed $3V \times N_b \left(1 + h\frac{d+1}{n_c}\right)$ times. For instance, the overall cost associated to this canonicalization is only $2^{128.1}$ bit operations compared to $2^{139.1}$ for the naive method in the SM+$\mathcal{P}$ hybrid attack on Rainbow-I. Note also that for the GeMSS parameters we analyze, it is possible to use $p_{s,\varepsilon} = 1$ with very little overhead. This is because we use $b = 1$ in all cases, and when $b = 1$ the optimal choices of attack parameters give V/N_b very close to 1. For other attack parameters, though, there may be significant benefit from choosing a more optimized formula for $p_{s,\varepsilon}$.

B.3 Measuring Processor Utilization for Parallel XL

Recall that the rows of the considered square submatrix $\boldsymbol{M}(\mathcal{Q}_b)'$ of the Macaulay matrix may be partitioned into $\binom{N+b-2}{b-1}\binom{n_c}{d+1}$ blocks of average size $n_r' < n_r$. Each such block is analyzed by some number of processing clusters that compute partial sums for some of these rows. We saw in Section 8.2 that the average number of such processing clusters is equal to $g := h(d+1)/n_c$.

One may place a processor within each active processing cluster for each block of rows to compute partial sums as well as a processor at the central accumulator to add the partial sums for that block and write the result to memory. As g is an average, this strategy requires approximately $(g+1) \cdot \#\text{blocks} = (g+1) \cdot \binom{N+b-2}{b-1}\binom{n_c}{d+1}$ processors in total. Note that all of the processors in the processing clusters are operating simultaneously while the processors in the central accumulator are idle, and the processors in the processing clusters are idle while the processors in the central accumulator are simultaneously adding the partial sums. In particular, let t_p represent the greatest time required by a processor in a processing cluster generating partial sums and let t_σ represent the greatest time required by a processor in the central accumulator to compute the total sum corresponding to these rows. The quantity t_p includes $N(d+1)$ field multiplications and at most one fewer field addition, while the quantity t_σ includes at most d field additions. Since each field multiplication costs roughly $2(\log_2(q)^2 + \log_2(q))$ bit operations while each field addition requires only $\log_2(q)$ bit operations, we see that $t_p \approx 2N(d+1)\log_2(q)^2$, while $t_\sigma \approx d\log_2(q)$.

The total idle time for one matrix-vector multiplication is then approximately $\#\text{blocks} \cdot t_p + g \cdot \#\text{blocks} \cdot t_\sigma$, whereas the total processing time is approximately $(g+1) \cdot \#\text{blocks} \cdot (t_p + t_\sigma)$. Taking the quotient of total idle time relative to the total time computing the matrix-vector product, we observe that we incur an extra factor of

$$\frac{t_p + gt_\sigma}{(g+1)(t_p+t_\sigma) - (t_p + gt_\sigma)} = \frac{t_p + gt_\sigma}{gt_p + t_\sigma}$$

due to parallelization. Note that the ratio t_p/t_σ is greater than $2N\log_2(q)$. Thus, for any instance we consider here, t_p is at least hundreds of times larger than t_σ. In particular, the penalty for parallelization is roughly g^{-1}, which can only contribute a small fraction of a bit to the complexity of the calculation. Thus, the task of running XL on Support-Minors instances as we propose in this paper is in theory embarrassingly parallel.