Algorithmic techniques in algebraic cryptanalysis and their applications

Makarim, R.H.

6 Solving MQ Challenges

Contents

In Section 2.2, we have discussed the computational hardness of the MQ problem that can be used as a basis for the security of public-key cryptography even in the presence of quantum computers. There are several constructions of public-key cryptography schemes whose security are based on the hardness of the MQ problem over finite fields such as [FPR17a, DCP+17, BPSV17]. While theoretical asymptotic complexity analysis to solve the MQ problem is important, it is also necessary to assess the concrete hardness of this computational problem.

In practice, studying the resources required to solve concrete MQ problem instances can be used to determine security parameters for cryptographic schemes that are deemed sufficient for concrete security levels. The difficulty of solving the MQ problem over a finite field depends on several factors such as order of the finite field, number of variables, number of equations, sparsity/density of the system, etc. In order to investigate the impact of these parameters to the complexity of solving the MQ problem in practice, one needs to construct concrete instances of the MQ problems with realistic parameters. By realistic parameters, we refer to parameters such that corresponding MQ problems are neither too easy nor too hard to solve in practice.

In April 2015, Yasuda et al. [YDH+15a] presented an open public challenge for the MQ problem with various parameters and increasing difficulty. One of the aims of the challenge is to evaluate the current state-of-the-art implementation of different algorithms that can solve MQ problems over finite fields. The official website of the MQ challenges can be found in the following link

`https://www.mqchallenge.org`.

In this chapter, we will describe these challenges and our efforts to solve them. Section 6.1 gives an overview of the MQ challenge, the rationale behind categorization of the challenges and how the parameters were selected. Section 6.2 describes existing state-of-the-art implementation of algorithms that managed to find solutions for some MQ challenge parameters. In Section 6.3 we present an overview of hybrid approach to solve the MQ problem over finite fields. We use the hybrid approach to solve several signature-type MQ challenges using M4GB algorithm. We will discuss it further in Section 6.4. In Section 6.5, we will give cost estimation in terms of CPU time and memory to solve larger parameters of signature-type MQ challenge with respect to M4GB.

6.1 Overview of the MQ Challenge

The MQ challenges are sets of concrete instances of the MQ problems over finite fields in various level of difficulties. The goal of the MQ challenges is to stimulate the research on algorithms to solve the MQ problems over finite fields and to analyze their applicability in practice. The challenge was initiated following similar directions of other existing open challenges for hard computational problems used as trapdoor in public-key cryptography. For instance, the RSA factoring challenge [RSA91], the Elliptic Curve Cryptography challenge (ECC) by Certicom [ECC97], and the Lattice Challenge [SVP10]. These challenges provided many publicly accessible concrete instances of problems with various parameters in increasing difficulties.

The three parameters that needs to be set to construct an instance of the MQ problem over a finite fields are: the order of the finite field q, the number of variables n, and the number of equations m. The authors of the MQ challenges selected $\mathbb{F}_2$, $\mathbb{F}_{2^8}$, and $\mathbb{F}_{31}$ as the possible base fields for the polynomials. The extension field $\mathbb{F}_{2^8}$ is constructed using the irreducible polynomial $x^8 + x^4 + x^3 + x^2 + 1 \in \mathbb{F}_2[x]$. The rationale of choosing $\mathbb{F}_{2^8}$ and $\mathbb{F}_{31}$ is because typically operations in an even-characteristic finite field are more efficient than in odd-characteristic finite fields. This difference will affect the complexity of solving larger instances of MQ problems in practice. The choice of $\mathbb{F}_2$ is considered to be a special case since there are dedicated approaches to solve those challenges, such as [FJ03, BCC$^+$10, BCC$^+$13].

The MQ challenges consider two types of challenge constructions. The first type of construction is for encryption-type parameters and the second type is for signature-type parameters. The former is characterized by an overdefined system of equations, where $m > n$ and the latter is characterized by an underdefined system of equations where $m < n$. Note that a random system of equations over $\mathbb{F}_q$ with $m > n$ has a probability approximately $\frac{1}{q^{m-n}}$ to have a solution. In order to guarantee the existence of a solution for an overdefined system, it is necessary to first randomly select a vector $s \in \mathbb{F}_q^n$. Once the coefficients of polynomials $f_1, \ldots, f_m \in \mathbb{F}_q[x_1, \ldots, x_n]$ have been generated, the constant coefficients $c_1, \ldots, c_m$ of each polynomial $f_1, \ldots, f_m$ must be adjusted to $c_i \leftarrow c_i - f_i(s)$ for all $i \in \{1, \ldots, m\}$. In this way, the overdefined system is guaranteed to have a solution, which is the vector s itself. However, for random systems with $m < n$, it is generally unnecessary to generate a solution in advance.

For MQ challenges representing public-key encryption, the authors selected parameter values with $m = 2n$ as in the case of several existing proposals such as the ABC scheme [TDTD13] and ZHFE [PBD14]. As for signature schemes, the organizer took Rainbow [DS05] as a representative of MQ-based digital signature schemes and selected parameter values with $n \approx 1.5m$.

In total there are six (6) different types of MQ challenges. The parameters of all type of challenges are described in Table 10. Type I, II, and III consist of overdefined system of equations with $2n$ equations and n variables. Type IV, V and VI consist of underdefined systems of equations with m equations and $\approx 1.5m$ variables.

The smallest challenge parameters (n and m) for each type were chosen based on estimations that it would take at least one month to solve them. For systems over $\mathbb{F}_2$, the estimation is based on the benchmark of libFES [BCC$^+$10].

Type	$\mathbb{F}_q$	Category	(n, m)	Parameter
I	$\mathbb{F}_2$	Encryption	$m = 2n$	$55 \leq n \leq 100$
II	$\mathbb{F}_{2^8}$	Encryption	$m = 2n$	$35 \leq n \leq 54$
III	$\mathbb{F}_{31}$	Encryption	$m = 2n$	$34 \leq n \leq 53$
IV	$\mathbb{F}_2$	Signature	$n \approx 1.5m$	$55 \leq m \leq 100$
V	$\mathbb{F}_{2^8}$	Signature	$n \approx 1.5m$	$16 \leq m \leq 35$
VI	$\mathbb{F}_{31}$	Signature	$n \approx 1.5m$	$16 \leq m \leq 35$

Table 10: *Six (6) different types of MQ challenges. The smallest choice of n and m were selected based on an estimation that it would take at least one month of computation to find a solution using four 6-cores 2.9GHz Intel Xeon CPU E5-4617 equipped with 15MB Intel smart cache and 1TB of RAM.*

For systems over $\mathbb{F}_{2^8}$ and $\mathbb{F}_{31}$, the estimation is based on running the Gröbner bases algorithm implemented in Magma [BCP97] v2.19-9 on four 6-cores 2.9GHz Intel Xeon CPU E5-4617 equipped with 15MB cache and 1TB of RAM. The experiments considers the direct approach to solve the system of equations for encryption-type parameters, i.e., type I, II and III. For the signature-type parameters (type IV, V, and VI) the benchmark were done on square system obtained by randomly fixing the value of $n - m$ variables. The results also estimated that the time complexity to solve encryption-type parameters and signature-type parameters should increase by a factor of 2 and 10, respectively.

6.2 Related Work

Since its public announcement, there have been various approaches to solve the MQ challenges. Some approaches were dedicated towards systems over $\mathbb{F}_2$ while others are tailored for the non binary field encryption-type or signature-type parameters. Thus we classify the approaches into three (3) different groups and we summarize them in this section.

For systems over $\mathbb{F}_2$, i.e., type I and IV, almost all approaches were equally effective to find solutions for both overdefined and underdefined systems of equations. At the time of writing there are six (6) algorithms that managed to solve type I and IV challenges: High-Performance Crossbred [BS23], exhaustive search with gray-code enumeration [BCC$^+$13], Crossbred algorithm [JV17], a variant of Crossbred algorithm implemented in GPU [NNY18], and two other algorithms that, to the best of our knowledge, have no accompanying publications (in the MQ challenge submission, the authors of these two algorithms referred to them as "improved crossbred" and "a new algorithm". We will refer to them similarly to avoid confusion). We summarize these results in Table 11. We see that the Crossbred algorithm and its variants have solved the highest parameter for type I challenge ($n = 83$) and type IV challenge ($m = 76$).

For non-binary field overdefined systems of equations, i.e., type II and III challenges, there are three (3) algorithms that managed to find solutions for some of the parameters: the XL algorithm [CCNY12], the "Modified-F_4" algorithm, and the "F_4-style" algorithm. However, we are not aware of any publications that described the last two algorithms or their implementations. The XL algorithm in [CCNY12] uses the Block Wiedemann algorithm [Cop94] to find a solution for the sparse system of linear equations defined over a finite field,

Algorithms	Type		Reference
	I (n)	IV (m)	
High-Performance Crossbred	75-77, 80, 83	60, 64, 68, 71-72, 75-76	[BS23, JV17]
Improved Crossbred†	74	68-69	-
Crossbred-GPU	55-74	67	[NNY18]
"A new algorithm"†	55-62, 67, 69	55, 60-67	-
Crossbred	67-74	-	[JV17]
Exhaustive Search	55-66	55-66	[BCC+13]

Algorithms	Resources	
High-Performance Crossbred	$n = 75, 76, 77$	1024 AMD EPYC 7J13 cores
	$n = 80$	2048 AMD EPYC 7J13 cores
	$n = 83$	3488 AMD EPYC 7J13 cores
	$m = 60$	Intel Xeon Gold 6240 (40 cores)
	$m = 64$	Intel Xeon Gold 6240 (2560 cores)
	$m = 68$	Intel Xeon Gold 6240 (5120 cores)
	$m = 71$	Intel Xeon Gold 6248 (10240 cores)
	$m = 72, 75, 76$	Intel Xeon Gold 6248 (20480 cores)
Improved Crossbred†	8x Intel i7 8700 with 32GB RAM and 10x GeForce GTX 1080Ti	
Crossbred-GPU	$n = 55, \ldots, 67$	Nvidia GTX 980 graphics card.
	$n = 68, \ldots, 74$	27 x AMD FX(tm)-8350 with 11 x Nvidia GTX 780 CUDA v7.5, 1 x Nvidia GTX 780 CUDA v8.0 and 15 x Nvidia GTX 980 CUDA v7.5.
"A new algorithm"†	Intel i9-7900X @ 3.30 GHz and a cluster of Intel Xeon @ 2.6 GHz	
Crossbred	Heterogeneous cluster of Intel Xeon @ 2.7-3.5 Ghz. The number of cores used for $n = 67, \ldots, 74$ are respectively equal to $256, 192, 512, 4096, 1760, 5320, 608$ and 448.	
Exhaustive Search	Rivyera, 128 Spartan 6 FPGAs.	

Table 11: *Summary of algorithms and resources used to solve type I and IV challenges.*

Algorithms	Type		Reference
	II (n)	III (n)	
XL	-	$34-38$	[CCNY12]
F_4-style†	$36-37$	37	-
Modified-F_4†	35	34	-

Algorithms	Resources	
XL	$n = 34, \ldots, 36$	4 x AMD Opteron 6282 SE
	$n = 37, 38$	2x AMD EPYC 7742
F_4-style†	4 x Intel(R) Xeon(R) CPU E5-4669 v4, 2.20GHz, 1TB RAM	
Modified-F_4†	Intel(R) Xeon(R) CPU E5-2620 v4, 2.10GHz with 256GB RAM	

Table 12: *Summary of algorithms and resources used to solve type II and III challenges.*

which is the case of the linearized system in XL. This algorithm currently holds the record for type III challenge with $n = 38$ while the "F_4-style" algorithm holds the record for type II challenge with $n = 37$. We summarize these results in Table 12.

For non-binary field underdefined systems of equations, i.e., type V and VI, there are six (6) algorithms that successfully found solutions for some of the parameters: M4GB [MS17], F_4 based on Hilbert-driven algorithm [ST23], Improved-F_4, MiniGB-F_4, M4GB 1.2 and Faugère's F_5 algorithm [Fau02]. Note that at the time of writing, we are not aware of any accompanying publications that describe MiniGB-F_4, Improved-F_4 and M4GB 1.2 algorithms. However a brief description of the MiniGB-F_4 can be found in the abstract of the following presentation [Che16]. It mentioned MiniGB-F_4 as a Buchberger-style Gröbner bases algorithm that maintains minimal basis throughout the computation. For the Improved-F_4 algorithm, the description given in the MQ challenge submission is that the algorithm takes advantages of the sparsity of the matrices combined with GPU-accelerated elimination steps.* The M4GB algorithms hold the record for both type V and VI respectively for the period of 5.8 years (Oct 2017-Aug 2023) and 6.3 years (Jul 2017-Oct 2023).

6.3 Overview of Hybrid Approach

One of the properties of polynomials in MPKC schemes is that they are defined over a finite field. In some cases, the order of the finite field is relatively "small". For instance, the public-key of Gui [DCP+17], one of the multivariate-based digital signature proposal for NIST post-quantum cryptography, uses polynomials over $\mathbb{F}_2$. Another NIST post-quantum proposal LUOV [BPSV17] uses polynomials over $\mathbb{F}_{2^r}$ for $r \in \{8, 48, 64, 80\}$.

The hybrid approach to solve a zero-dimensional system of multivariate polynomials over a finite field is a combination of a Gröbner basis algorithm and exhaustive search. The method was proposed by Bettale, Faugère, and Perret

† We are not aware of any publication that mentioned this approach in detail.
* `https://www.mqchallenge.org/details/details_V_20230822.html`

Algorithms	Type		Reference
	V (m)	VI (m)	
F_4 based on Hilbert-driven	-	$21-22$	[ST23]
Improved-F_4[†]	20	-	-
M4GB	$16-19$	$16-20$	[MS17]
MiniGB-F_4	$18, 19$	-	[Che16]
M4GB1.2[†]	$16-17$	-	-
F_5	16	16	[Fau02]

Algorithms	Resources	
F_4 based on Hilbert-driven	AMD EPYC 7742, 2TB RAM	
Improved-F_4	1 x AMD Ryzen Threadripper 3970X, 192GB RAM, 2 x GeForce RTX 3080Ti	
M4GB	2 x Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz and 128GB RAM	
MiniGB-F_4	$m = 18$	8 x Intel(R) Xeon(R) CPU E5620, 1 x Intel(R) Xeon(R) CPU E3-1230 v3 and 1 x Intel(R) Xeon(R) CPU E3-1245 v3.
	$m = 19$	2 x AMD opteron 6376x4 512GB RAM.
M4GB1.2	Intel(R) Xeon(R) Silver 4208 CPU @ 2.10GHz, 32GB RAM	
F_5	Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz	

Table 13: *Summary of algorithms and resources used to solve type V and VI challenges.*

in [BFP09, BFP12]. It broke some proposed parameters of the TRMS signature scheme [WHL+05] and the UOV signature scheme [KPG99, BERW08]. The hybrid approach also managed to find a collision in some proposed multivariate hash functions [DY07].

The main idea of the hybrid approach is to solve many systems with less number of variables than the original ones by fixing, say, k out of the n original variables. This approach reduces the complexity of directly solving the original system of equations using the Gröbner basis algorithm at the expense of having to solve many of these subsystems. Furthermore, we can run the Gröbner basis algorithm on each of the q^k subsystems in parallel where q is the order of the finite field.

We first recall the notion of degree of regularity as the main parameter to measure the complexity of Gröbner bases algorithm.

Definition 6.1 (Degree of Regularity [BFS04])**.** *The degree of regularity of a zero-dimensional ideal* $I = \langle f_1, \ldots, f_m \rangle \subset \mathbb{F}[x_1, \ldots, x_n]$ $(m \geq n)$ *is*

$$d_{reg} = \min \left\{ d \geq 0 : \dim_{\mathbb{F}}(\{f \in I, \deg(f) = d\} \cup \{0\}) = \binom{n+d-1}{d} \right\}.$$

Remark 6.2. *Intuitively, the degree of regularity is the smallest* d *such that the number of linearly independent nonzero polynomials of degree* d *in a zero-dimensional ideal* I *is equal to the number of monomials of degree* d*.*

In order to understand the complexity of solving random system of equations, the notion "random system" is formalized as regular sequence and semi-regular sequence [Bar04]. We are mainly interested with the notion of semi-regular sequence since a random overdefined system of equations is generally assumed to be semi-regular (see for instance Hypothesis 3.3 in [BFP09] and Hypothesis 1 in [BFP12]).

Definition 6.3 (Semi-Regular Sequence [BFP09])**.** *A sequence of homogeneous polynomials* $(f_1, \ldots, f_m) \subset \mathbb{F}[x_1, \ldots, x_n]$ *of respective degrees* $d_1, \ldots, d_m$ *is semi-regular if*

- $\langle f_1, \ldots, f_m \rangle \neq \mathbb{F}[x_1, \ldots, x_n]$ *and*
- *for all* $i \in \{1, \ldots, m\}$ *and* $g \in \mathbb{F}[x_1, \ldots, x_n]$, $g \cdot f_i \in \langle f_1, \ldots, f_{i-1} \rangle$ *and* $\deg(g \cdot f_i) < d_{reg}$ *implies that* $g \in \langle f_1, \ldots, f_{i-1} \rangle$.

In [BFS04, Proposition 6], it has been shown that the degree of regularity of a semi-regular sequence can be computed explicitly.

Proposition 6.4 (Degree of Regularity of Semi-Regular Sequence [BFS04])**.** *The degree of regularity of a semi-regular sequence* $(f_1, \ldots, f_m) \in \mathbb{F}[x_1, \ldots, x_n]$ *of respective degrees* $d_1, \ldots, d_m$ *is given by the index of the first non-positive coefficient of*

$$\sum_{k \geq 0} c_k z^k = \frac{\prod_{i=1}^{m} (1 - z^{d_i})}{(1-z)^n}.$$

The complexity analysis of hybrid approach is based on the complexity of F_5 algorithm. The complexity of computing Gröbner basis of a zero-dimensional ideal with F_5 algorithm is described in the following proposition.

Proposition 6.5 (Complexity of F_5 [BFP12, BFSY05])**.** *The complexity of computing Gröbner basis of a semi-regular zero-dimensional ideal in n variables and m equations with F_5 algorithm is*

$$\mathcal{O}\left(\binom{n+d_{reg}}{d_{reg}}^{\omega}\right) \tag{6.1}$$

where d_{reg} is the degree of regularity of the system and $2 \leq \omega \leq 3$ is the linear algebra constant.[*]

Remark 6.6. *Although the number of equations m is not explicitly mentioned in Equation 6.1, note that the degree of regularity depends on m.*

When an n-variable system of equations is defined over a finite field $\mathbb{F}_q$ and we are only concerned with solutions in $\mathbb{F}_q^n$, it is possible to find all solutions by exhaustive search, which has complexity $\mathcal{O}(q^n)$. Moreover, the exhaustive search is easily parallelizable with negligible memory cost. Thus, one can implement it in massively parallel architectures such as Graphical Processing Units (GPUs) or Field Programmable Gate Arrays (FPGAs). The effectiveness of this approach has been shown in practice by solving systems of equations over $\mathbb{F}_2$ [BCC+10, BCC+13]. In a scenario where it is not feasible to perform exhaustive search, one can run a Gröbner basis algorithm with degree-reverse lexicographic ordering followed by the FGLM algorithm (if necessary) to obtain the set of solutions.

The main question then for the hybrid approach is to find the value of k such that the total expected complexity is minimal. Before obtaining the total expected complexity of the hybrid approach, we first need to understand the complexity of solving a single subsystem with m equations and $n-k$ variables. The authors of [BFP09] made the following assumption.

Assumption 6.7. *Let $\{f_1, \ldots, f_m\} \subset \mathbb{F}_q[x_1, \ldots, x_n]$ be a semi-regular system of equations with degree d. For all $0 \leq k \leq n$ and $(v_1, \ldots, v_k) \in \mathbb{F}_q^k$, we assume that*

$$\{f_1(x_1, \ldots, x_{n-k}, v_1, \ldots, v_k), \ldots, f_m(x_1, \ldots, x_{n-k}, v_1, \ldots, v_k)\}$$

is also semi-regular system of equations.

The complexity of the hybrid approach for semi-regular system of equations is stated in the following proposition.

Proposition 6.8 ([BFP12])**.** *Using Stirling's approximation, i.e. $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$, the complexity of solving a semi-regular system of equations with a hybrid approach using F_5 algorithm is equal to*

$$q^k \left(\frac{1}{\sqrt{2\pi}} \cdot \frac{(n-k+d_{reg}(k))^{n-k+d_{reg}(k)+\frac{1}{2}}}{(n-k)^{n-k+\frac{1}{2}} d_{reg}(k)^{d_{reg}(k)+\frac{1}{2}}}\right)^{\omega}$$

where $2 \leq \omega \leq 3$ is the linear algebra constant and $d_{reg}(k)$ is the degree of regularity of each subsystem obtained after fixing k variables.

[*] The complexity of F_5 algorithm mentioned in [BFP09] is equal to $\mathcal{O}\left(\left(m \cdot \binom{n+d_{reg}-1}{d_{reg}}\right)^{\omega}\right)$. This was later revised in [BFP12] since Equation 6.1 is a more appropriate complexity estimation for semi-regular systems (see Assumption 6.7).

The initial result on the hybrid approach in [BFP09] suggested that one should find the optimal value of k by simply comparing the total complexities of the hybrid approach for different values of $0 \leq k \leq n$. This approach is described in Algorithm 6.1.

Input: q - order of the finite field
Input: n - number of variables of the system
Input: m - number of equations of the system
Input: $d_1, \ldots, d_m$ - the degree of each polynomial
Result: k - the best theoretical trade-off for hybrid approach
1 $A \leftarrow [\,]$
2 **for** $k \leftarrow 0$ **to** n **do**
3 $\quad d_k \leftarrow$ index of the first non-positive coefficient in the series $\frac{\prod_{i=1}^{m}(1-z^{d_i})}{(1-z)^{n-k}}$
4 $\quad A[k] \leftarrow q^k \left(\frac{1}{\sqrt{2\pi}} \cdot \frac{(n-k+d_k)^{n-k+d_k+\frac{1}{2}}}{(n-k)^{n-k+\frac{1}{2}} \cdot d_k^{d_k+\frac{1}{2}}} \right)^{\omega}$
5 **return** k such that $A[k]$ is minimum.

Algorithm 6.1: An algorithm to find the optimal value of k in hybrid approach.

An improved analysis of hybrid approach in [BFP12] provided an explicit formula for the best trade-off k to solve a system of quadratic equations, that is

$$\left\lceil n \cdot \frac{10.86 \cdot \omega^2}{(4.16 \cdot \log_2 q - 3.14 \cdot \omega)^2} \right\rceil \tag{6.2}$$

(see [BFP12, Proposition 3.3]). Using Equation 6.2, the complexity of the hybrid approach for a random quadratic system of equations over $\mathbb{F}_q$ is asymptotically equivalent to

$$2^{n\omega(1.38-0.63\omega \log_2(q)^{-1})} \tag{6.3}$$

when $n \to \infty, q \to \infty$ and $\log_2(q) \ll n$ (see [BFP12, Theorem 3.1]).

We would like to remark that a similar approach was also proposed for the XL algorithm in [CKPS00] called the FXL (Fixed XL) algorithm. The paper stated that the complexity of solving a system of equations using the XL algorithm can be reduced by guessing a small number of variables. The asymptotic complexity of the FXL was further described in [YCC04]. Recall that the XL algorithm is a redundant variant of the F_4 algorithm, thus it is considered to be less efficient than both the F_4 and the F_5 algorithm. Furthermore, the asymptotic analysis of the FXL algorithm in [YCC04] did not yield a method to determine the optimal number of guessed variables.

6.4 Solving Type V and VI of the MQ Challenge

In this section we will discuss how the M4GB algorithm was able to find solutions of some of the MQ challenges. The M4GB algorithm managed to solve signature-type parameters (underdefined systems) with coefficients in $\mathbb{F}_{31}$ and $\mathbb{F}_{2^8}$, i.e., type V and type VI of MQ challenge. We used the following two machines to break these challenges.

A) A desktop machine equipped with Intel(R) Core(TM) i7-2600K CPU @ 3.40 GHz with four (4) processor cores and 16GB RAM

B) A NUMA (Non-Uniform Memory Access) machine with two nodes. Each node is equipped with Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz with ten (10) processor cores and 128GB RAM.

We refer to the above machines as machine A and B respectively. The machine A was used to find solutions for the smallest parameter of type V and type VI, which consist of 24 variables and 16 equations defined over $\mathbb{F}_{2^8}$ and $\mathbb{F}_{31}$ respectively. Larger parameters were solved by running M4GB on machine B. A summary of challenges solved by M4GB is shown in Table 14. The solutions found are provided in Appendix A. It is expected for a random underdefined system of equations to have multiple solutions. However, it is sufficient to find a single solution in the base field in order to break a given parameter.

Our strategy to solve type V and VI MQ challenges uses the hybrid approach. We first fix the value of $n - m$ variables in order to yield a square system with equal number of variables and equations. This will be followed by the hybrid approach iterating over all possible values of another k number of variables. In most cases, we selected $k = 1$ except for the largest parameter of type VI that we solved, where we selected $k = 2$. A summary of this strategy is given in Algorithm 6.2.

It is possible that the square system F' in Algorithm 6.2 has no solution. For a system over a prime field, it is known that a random instance of a square system with quadratic polynomials has no solution with probability $1/e$ asymptotically in n [FB09, Corollary 3.1].* If F' eventually had no solution, the Algorithm 6.2 would run another iteration to generate a new F' by choosing a new $(v_1, \ldots, v_t) \in \mathbb{F}_q^t$. The benefit of this approach is that the computation of Gröbner bases from line 7 up to line 19 is trivially parallelizable, i.e., we can compute Gröbner basis for each F'' independently. Moreover, we can also estimate the average complexity and worst-case complexity in practice to obtain a solution for the whole system by computing a Gröbner basis for one of the subsystems F''.

For instance, solving one subsystem of type VI with 15 variables and 16 equations took 1 hour and 10 minutes wall-clock time. An average total CPU-time estimate to solve type VI parameter with $m = 16$ would thus take 18.7 hours, assuming that the generated square system has a solution. The implementation of M4GB uses 837MB of memory to solve each subsystem, which allowed simultaneous computation on 8 subsytems within the available 16GB of memory in machine A. We broke both type V and VI challenges for $m = 16$ on the quadcore desktop machine A in 9.3 hours and 1.2 hours wall-clock time after covering 29% and 22.6% of the search space respectively. Note that this is significantly faster than the original expected one month estimate on a Xeon system using Magma by the authors of the MQ challenges.

For larger parameters for type V and VI, we used machine B. We ran 10 simultaneous Gröbner bases computations using M4GB by forcing each process to one NUMA node using the `numactl` program.

* More generally, given a multivariate polynomial system with $n+\alpha$ random equations of degree $d \geq 2$ in n variables over $\mathbb{F}_p$, with a prime p. The probability that the system has no solution is $e^{-p^{-\alpha}}$ asymptotically in n. For the proof, see [FB09, Section 3]

Input: $F = \{f_1, \ldots, f_m, x_1^q - x_1, \ldots, x_n^q - x_n\} \subset \mathbb{F}_q[x_1, \ldots, x_n]$ with $n > m$
Input: $k \in \mathbb{Z}_{>0}$
Output: A solution $(s_1, \ldots, s_n) \in \mathbb{F}_q^n$ of F (assume it exists)

$G \leftarrow \{\}$
solution_found $\leftarrow$ False
$t \leftarrow n - m$
while solution_found = False **do**
  Choose randomly $(v_1, \ldots, v_t) \in \mathbb{F}_q^t$
  $F' \leftarrow \{f(x_1, \ldots, x_m, v_1, \ldots, v_t) : f \in F\}$
  for $(w_1, \ldots, w_k) \in \mathbb{F}_q^k$ **do**
    $F'' \leftarrow \{f'(x_1, \ldots, x_{m-k}, w_1, \ldots, w_k) : f' \in F'\}$
    $G \leftarrow$ GröbnerBasis(F'') ▷ Compute the reduced Gröbner bases
    if $G \neq \{1\}$ **then**
      while $\exists g \in G : g - \mathsf{LT}(g) \notin \mathbb{F}_q$ **do**
        Let x be a variable in g
        for $c \in \mathbb{F}_q$ **do**
          $G' \leftarrow$ GröbnerBasis$(G \cup \{x + c\})$
          if $G' \neq \{1\}$ **then**
            $G \leftarrow G'$
            break
      solution_found $\leftarrow$ True
      break
G `is of the form` $\{x_1 + c_1, \ldots, x_{m-k} + c_{m-k}\}$ `where` $c_i \in \mathbb{F}_q$.
$(s_1, \ldots, s_n) \leftarrow (-c_1, \ldots, -c_{m-k}, w_1, \ldots, w_k, v_1, \ldots, v_t)$
return $(s_1, \ldots, s_n)$

Algorithm 6.2: A strategy based on hybrid approach to find a solution for an underdefined system of equations over a finite field

For type VI with $m = 19$, we modified our strategy to compute simultaneous Gröbner bases on subsystems with $n = 18$ variables. We first observed that in the final stage, the M4GB algorithm takes a significant amount of time using a single thread. Thus, instead of waiting for the whole process to finish, one can start the Gröbner basis computation for the next subsystem as soon as this last stage of the previous computation begins. Of course this requires that the available memory is sufficiently large for these concurrently running Gröbner basis computations. In this way, all processors are fully occupied and it significantly reduces the time to obtain a solution.

Type	(n, m)	Machine Used	# Node	k	Time (sec)	Memory	Duration	Search Space Covered
V*	$(24, 16)$	A	1	1	1844	1.11GB	$\approx$ 9.3 hours	29%
V	$(25, 17)$	B	1	1	24125	3.24GB	$\approx$ 46.33 hours	46.5%
V	$(27, 18)$	B	2	1	94266	11.75GB	$\approx$ 10.9 days	95.7%
V	$(28, 19)$	B	2	1	478194	45.05GB	$\approx$ 69.75 days	71.5%
VI	$(24, 16)$	A	1	1	4273	836.9MB	$\approx$ 1.2 hours	22.6%
VI	$(25, 17)$	B	1	1	30955	3.25GB	$\approx$ 9.87 hours	64.5%
VI	$(27, 18)$	B	1	1	194474	11.9GB	$\approx$ 31.48 hours	12.9%
VI	$(28, 19)$	B	2	1	474008	47.1GB	$\approx$ 7.61 days	54.8%
VI	$(30, 20)$	B	2	2	139088	12.67GB	$\approx$ 11.32 days	21.5%

Table 14: *Summary of MQ challenge parameters solved using M4GB*

* The CPU time and memory usage for this parameter were run on machine B due to the loss of information.

6.5 Cost Estimation for Other Parameters

In this section we provide a cost estimation to find solution for unsolved instances of type V and VI of MQ challenge with hybrid approach using the M4GB algorithm. We focus on estimating the total CPU time and memory usage of the implementation described in Section 5.6.

6.5.1 Approach

Recall that solving an underdefined system of equations with n variables and m equations, where $n > m$, is equivalent to solving a system with m variables and m equations obtained after fixing the value of $n - m$ variables from the original system. To simplify our methodology, we only estimate the cost of hybrid approach on systems with equal number of variables and equations and we assume that such system obtained from type V or VI of MQ challenges has a solution. We also assume that the CPU time and memory usage of M4GB implementation in Section 5.6 grow exponentially in terms of the number of variables n.

$m = n$		16	17	18	19	20	21	22	23	24	25
k	$\mathbb{F}_{2^8}$	2	1	2	1	2	1	2	2	2	2
	$\mathbb{F}_{31}$	4	5	5	3	5	4	6	5	4	6

$m = n$		26	27	28	29	30	31	32	33	34	35
k	$\mathbb{F}_{2^8}$	**3**	2	**3**	2	**3**	**3**	**3**	**3**	2	**3**
	$\mathbb{F}_{31}$	5	7	6	5	7	6	**8**	7	6	**8**

Table 15: *Optimal number of fixed variables k for F_5 algorithm applied to type V and type VI parameters of MQ challenge using Algorithm 6.1 assuming $\omega = 2.4$. The numbers highlighted in bold are the maximum k for the given finite field.*

Note that since type V and VI challenges are parametrized by the number of equations m and we restrict our approach on systems with $n = m$ variables, the results in this section are described in terms of m in order to give a unified picture of the estimation.

For a given number of equations/variables m and an order of a finite field q, the first step of hybrid approach is to find the optimal value of the number of fixed variables k. Here by optimal we mean the one that minimizes the total CPU time.* In order to find the optimal value k in practice, we shall examine the cost of running M4GB algorithm on subsystems generated after fixing k variables.

For each q and m, we run the M4GB algorithm for systems with $m - 1, m - 2, \ldots, m - k_{\max}$ variables where $k_{\max}$ is the maximum number of fixed variables. We select the value $k_{\max}$ by finding the maximum number of fixed variables for a given q using the theoretical result described in Algorithm 6.1

* Note that it is also possible to define the optimal value k in terms of other costs such as memory usage, energy, monetary cost, etc. We leave the study on those costs for future work.

for $n = m = 16, \ldots, 35$ and $d_1 = \ldots = d_m = 2$. We obtain $k_{\max} = 3$ for $q = 256$ and $k_{\max} = 8$ for $q = 31$. The complete result is given in Table 15.

After obtaining $k_{\max}$ for both $q = 31$ and $q = 256$, we collected the CPU time and memory usage of M4GB algorithm on subsystems of type V and VI starting from $m = 16$ up to the number of equations that are feasible to solve using a single node of machine B. The full result is described in Appendix B. The total CPU time and total memory usage are the product of the CPU time and memory usage by q^k. To simplify the estimation, we took the base-2 logarithm of the total CPU time as well as total memory usage and use the *least square method* to predict the cost for the unsolved parameters.

6.5.2 Result

For type V, our estimation suggests that guessing one variable from the square system is the optimal strategy for all the remaining unsolved challenge parameters, i.e., $m = 20, \ldots, 35$. This can be seen from the growth of the CPU hours for type V challenge in Figure 11 and the detail description in Table 16. For the next unsolved parameter, i.e., $m = 20$, we estimate that it would cost approximately 402045 CPU hours $\approx$ 45.9 CPU years and 52.61 TB of memory.

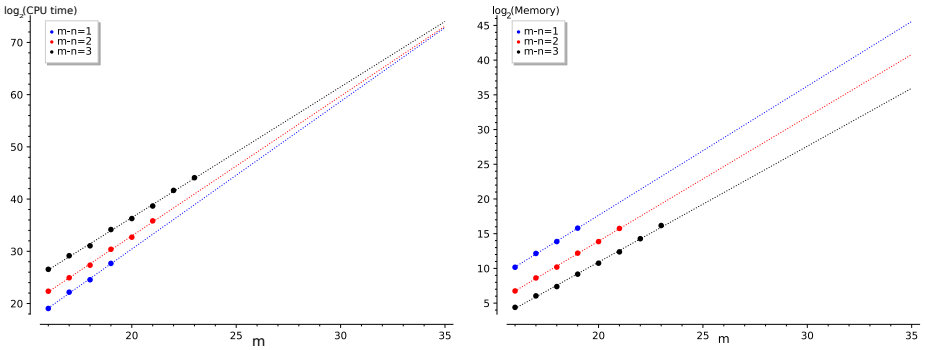

Figure 11: *Estimated growth of total CPU time (left) and per-subsystem memory usage (MB) (right) of hybrid approach for type V MQ challenge using the* M4GB *algorithm.*

For type VI challenge, the optimal number of variables to fix differs for different values of m. Our estimation suggests to fix 3 variables from the generated square system to solve the next unsolved parameter, which is $m = 21$. The cost to solve the next unsolved parameter, i.e., $m = 21$, would approximately takes 217452 CPU hours $\approx$ 24.82 CPU years and 159.17 TB. However, this is not the case for $m = 22, 23$ where guessing 2 variables from the generated square system is the optimal strategy to minimize the total CPU time. For $m \geq 24$, our estimation suggests to fix 3 variables except for the largest challenge parameter $m = 35$ where it suggests to fix 5 variables. The results are described in Table 16 and Figure 12.

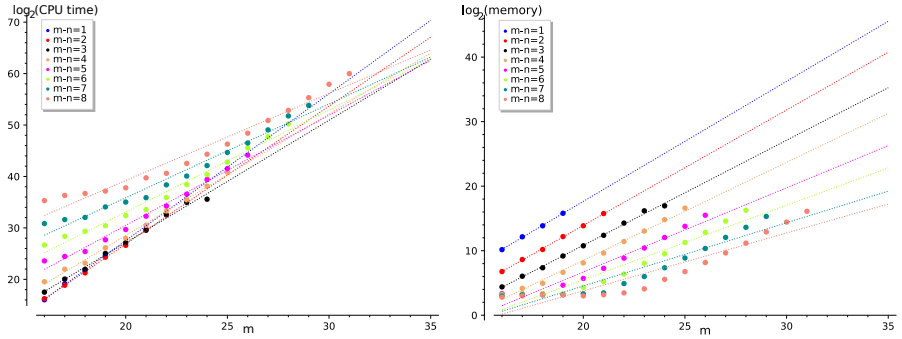

Figure 12: *Estimated growth of total CPU time (left) and per-subsystem memory usage (MB) (right) of hybrid approach for type VI MQ challenge using the* M4GB *algorithm.*

m	$m-n$		≈ CPU hours		≈ Memory (GB)	
	V	VI	V	VI	V	VI
20	1	-	402045	-	210	-
21	1	3	$2.85 \cdot 10^6$	217452	764	5.47
22	1	2	$2.02 \cdot 10^7$	$1.39 \cdot 10^6$	2776	189
23	1	2	$1.44 \cdot 10^8$	$8.94 \cdot 10^6$	10087	654
24	1	3	$1.02 \cdot 10^9$	$1.43 \cdot 10^7$	36643	130
25	1	3	$7.23 \cdot 10^9$	$1.57 \cdot 10^8$	133110	528
26	1	3	$5.13 \cdot 10^{10}$	$8.15 \cdot 10^8$	483536	1631
27	1	3	$3.64 \cdot 10^{11}$	$4.24 \cdot 10^9$	$1.76 \cdot 10^6$	5042
28	1	3	$2.58 \cdot 10^{12}$	$2.20 \cdot 10^{10}$	$6.40 \cdot 10^6$	15584
29	1	3	$1.83 \cdot 10^{13}$	$1.15 \cdot 10^{11}$	$2.32 \cdot 10^7$	$4.81 \cdot 10^4$
30	1	3	$1.30 \cdot 10^{14}$	$5.95 \cdot 10^{11}$	$8.44 \cdot 10^7$	$1.49 \cdot 10^5$
31	1	3	$9.23 \cdot 10^{14}$	$3.10 \cdot 10^{12}$	$3.06 \cdot 10^8$	$4.61 \cdot 10^5$
32	1	3	$6.55 \cdot 10^{15}$	$1.61 \cdot 10^{13}$	$1.11 \cdot 10^9$	$1.42 \cdot 10^6$
33	1	3	$4.64 \cdot 10^{16}$	$8.37 \cdot 10^{13}$	$4.04 \cdot 10^9$	$4.40 \cdot 10^6$
34	1	3	$3.30 \cdot 10^{17}$	$4.35 \cdot 10^{14}$	$1.47 \cdot 10^{10}$	$1.36 \cdot 10^7$
35	1	5	$2.34 \cdot 10^{18}$	$1.95 \cdot 10^{15}$	$5.32 \cdot 10^{10}$	$8.37 \cdot 10^4$

Table 16: *Estimated cost to solve larger parameters of type V (top) and VI (bottom) using our implementation of* M4GB *algorithm running on Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30 GHz. Memory usage is for a single subsystem and the total system memory required depends on the number of subsystems being solved simultaneously.*