Bachelor's Thesis Computing Science

Structured MQ Challenges

Tristan Lumi
s1117772

August 10, 2026

First supervisor/assessor:
dr. Simona Samardjiska

Second assessor:
prof. dr. Peter Schwabe

Radboud University

Abstract

The Multivariate Quadratic problem is an NP-complete problem that is used as the basis for many multivariate cryptographic schemes. Often attacking these schemes reduces the problem to a structured MQ instance, which becomes the limiting factor for the attack. Therefore, developing new methods for attacking these structured systems becomes vital for evaluating the security of schemes that can be reduced to these systems. In order to improve current cryptanalytical methods, we have created a number of challenges based on structured MQ systems, with the aim of providing practical targets for testing new methods. For the structure of these systems, we have chosen to create challenges based on bilinear MQ polynomial systems, multi-homogeneous polynomial systems, and the Kipnis-Shamir modeling of MinRank. Given that the difficulty of solving these structured systems is directly tied to the parameters with which they are constructed, we explore the relation between different parameters and the estimated time complexity to solve these structured systems. Using these estimations, we choose minimal parameter sets with which to generate the challenges, ensuring that the challenges are not trivial or impossible to solve.

Contents

Chapter 1

Introduction

Ever since the theoretical foundations for quantum computing were first established in the latter half of the 20th century, it became clear that developments in quantum computing could upend modern cryptographic methods. This became apparent by the development of Shor's algorithm in 1994 [25] and its implications on the vulnerability of schemes such as RSA and the Diffie-Hellman key exchanges, as Shor was able to show that a capable-enough quantum computer would be able to factor large integers in polynomial time. Over the following decade, further developments in quantum computing theory fortified the expectation that contemporary public-key cryptography would be broken by quantum computers. Therefore, NIST launched a standardization project for post-quantum cryptography in 2016, where the main goal of the project was to prepare for such a future where quantum computers can trivially solve modern public-key cryptoschemes. While the first four standards were chosen and announced in 2024, three of the four standards are lattice-based schemes, leading to a push for additional, non-lattice-based candidates to be added to the standard through the Additional Digital Signatures for the PQC Standardization Process competition. As of May 2026, NIST published their selection of 9 candidates to advance to the third round of the process, of which many candidates are multivariate schemes. These include `UOV` [20], `MAYO` [6], `SNOVA` [28] (which are based on `UOV`), and `MQOM` [5]. Prior to the third round selection, `GemSS` [9] and `Rainbow` [11] were also considered promising multivariate candidates, but did not advance to the third round.

These candidates are called multivariate public-key cryptosystems (MPKC's) and rely on the hardness of solving multivariate quadratic polynomial systems to remain secure. The problem of solving these polynomial systems is known as the MQ problem and is known to be NP-complete [17]. While many algorithms exist that are able to solve these systems [8, 29, 26, 15, 13, 16, 19], they only outperform exhaustive search reliably under certain upper bounds on parameter choices. Therefore finding the correct parameters with which to generate these polynomial systems is what guarantees the security of the cryptosystem against algebraic attacks [12].

The MQ problem is defined as finding a solution to a system of equations for polynomials of the following form:

$$p_i(x_1, x_2, ..., x_n) = \sum_{j=1}^{n} \sum_{k=1}^{n} \alpha_{j,k} x_j x_k + \sum_{j=1}^{n} \beta_j x_j + \gamma = 0$$

for $i \in [1, m]$ where m is the number of equations in the system and n the number of variables.

The broadness of the MQ problem is often used to generate schemes based on structured variants of the problem. This is done in order to produce "trapdoor" structures in the schemes

themselves, which hide intermediate secrets. Attacking these schemes often reduces the problem to solving structured (often bilinear) MQ polynomial systems, which becomes the limiting factor in solving instances of the scheme. Therefore, the complexity of attacking the scheme is derived from the ability to solve the structured systems produced by attacks.

In order to evaluate the efficiency of solving algorithms against the MQ problem, specific parameters must be chosen to make this process feasible for regular (non-quantum) hardware. For this purpose, many cryptography "challenges" have been proposed and published that aim to generate problems that should be solvable by regular hardware. The Fukuoka MQ Challenge [30] is based on the MQ problem, while many other such challenges also exist for other types of candidates for the post-quantum cryptography standard, such as the TU Darmstadt Lattice [21], Ideal Lattice [23], SVP [24], and LWE [18] challenges. The Technology Innovation Institute in Abu Dhabi launched a similar challenge in 2024 for the McEliece cryptosystem, aimed at evaluating its practical security [1]. In addition, the Ruhr-University Bochum released challenges for the Kyber scheme in 2023, and the MAYO and McEliece schemes in 2025 [27].

We are interested in expanding the domain of MQ challenges to improve cryptanalytical methods. For this we have created related challenges, based on bi- and multilinear variations of the MQ problem. This allows us to explore the hardness of solving structured MQ systems often encountered in attacks against MPKC's and potentially introduce new methods for tackling them.

1.1 Related Work

The Fukuoka MQ Challenge is based on the encryption and signature cases for MPKC, which serves as a challenge to break the cryptosystem, based on breaking the underlying MQ system [30]. As the solving strategy for the Fukuoka MQ Challenge is based on the underlying polynomials, due to it being an MPKC based challenge, there is an extra degree of difficulty in solving these instances. In addition, the underlying polynomials are based on the standard MQ case (defined as 1. in the Preliminaries), while most candidate schemes for the NIST PQC standard use structured polynomial systems.

In addition to the Fukuoka MQ challenge, the TII McEliece [1], TU Darmstadt [18, 21, 23, 24], and Bochum challenges [27] serve as interesting challenges for other quantum-resistant systems.

Therefore, we are interested in how non-standard variants of the this problem respond to these same algorithms, and how solving algorithms for other structures behave with these non-standard cases.

1.2 Our Contribution

In this work, we present new challenges for MQ cases with different types of structures. For this challenge, we will look at three bilinear variations of the MQ problem, bilinear MQ systems, multi-homogeneous MQ systems, and one which is based on the Kipnis-Shamir modeling of the MinRank problem [22].

In addition to outlining the instance generation of the challenge itself, we also provide estimations for the hardness of solving the instances for certain parameters. This guarantees that the instances are solvable and gives us insight into whether a solving algorithm over- or underperforms the estimation. Knowing this, we can improve and optimize solving algorithms against these systems.

This thesis outlines the types of polynomial systems we will be generating for the challenges, as well as how the challenge generator was designed. We then outline the benchmarking process

used to decide on minimal parameters and present the estimations for the hardness of solving these challenges. Finally, we explore the relation between the time complexity of solving these systems and their different potential parameter sets, with which we justify the parameter choices present in the challenges themselves.

Chapter 2

Preliminaries

The MQ (Multivariate Quadratic) problem is defined as the problem of solving multivariate quadratic polynomials, which are elements of some polynomial ring.

Below we define the "standard" system type of the MQ problem used to construct multivariate public key cryptosystems, as well as the systems used in the challenge instances. We generate three different types of systems, each different from the standard case.

Background on Finite Fields and Rings A field is a set together with two binary operations, addition and multiplication, such that the operations on that set satisfy field axioms. Thus, the set forms an abelian group under addition, the nonzero elements form an abelian group under multiplication, and multiplication distributes over addition. By extension, a finite field (denoted $\mathbb{F}_q$ for a field of order q) is a field with a finite number of elements. A ring, like a field, is a set with the operations addition and multiplication, but instead satisfies the ring axioms. In contrast to fields, rings do not require the existence of a multiplicative identity, multiplicative commutativity, or multiplicative inverses for all nonzero elements.

A polynomial ring (denoted as $\mathbb{F}_q[x_1, x_2, ..., x_n]$, with order q and number of variables n) has elements which are polynomials, with the property that the coefficients of the polynomials must be in $\mathbb{F}_q$.

Definition 1. (Standard MQ System) *Let q be a prime power, and $\mathbb{F}_q$ a finite field of order q. Let $n, m \in \mathbb{N}$, with n the number of variables and m the number of equations in the system. Let $\alpha, \beta, \gamma \in \mathbb{F}_q$. Then p_i with $i \in [1, m]$ is an element in $\mathbb{F}_q[x_1, x_2, ..., x_n]$ and is of the form:*

$$p_i(x_1, x_2, ..., x_n) = \sum_{j=1}^{n} \sum_{k=1}^{n} \alpha_{j,k} x_j x_k + \sum_{j=1}^{n} \beta_j x_j + \gamma = 0$$

p with elements $p_1, p_2, .., p_m$ is an MQ system.

Definition 2. (Bilinear MQ System) *Let q be a prime power, and $\mathbb{F}_q$ a finite field of order q. Let x and y be two sets of variables in p_i. Let $s, t, m \in \mathbb{N}$, with s the number of variables in x, t the number of variables in y, and m the number of equations in the system. Let $\alpha, \beta, \gamma, c \in \mathbb{F}_q$. Then p_i with $i \in [1, m]$ is an element in $\mathbb{F}_q[x_1, x_2, ..., x_s, y_1, y_2, ..., y_t]$ and is of the form:*

$$p_i(x_1, x_2, ..., x_s, y_1, y_2, ..., y_t) = \sum_{j=1}^{s} \sum_{k=1}^{t} \alpha_{j,k} x_j y_k + \sum_{j=1}^{s} \beta_j x_j + \sum_{k=1}^{t} \gamma_k y_k + \delta = 0$$

p with elements $p_1, p_2, .., p_m$ is an MQ system.

Definition 3. (Multi-homogeneous System)
Let q be a prime power, and $\mathbb{F}_q$ a finite field with order q. Let $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$ be sets of variables for some index i:

$$\mathbf{x}^{(i)} = (x_{1i}, x_{2i}, ..., x_{ni})$$
$$\mathbf{y}^{(i)} = (y_{1i}, y_{2i}, ..., y_{mi})$$

where $n, m \in \mathbb{N}$ are the respective sizes of the sets. Then for t, s the number of sets of $\mathbf{x}$ and $\mathbf{y}$ respectively:

$$p^{(i)}(x_1, x_2, ..., x_{nt}, y_1, y_2, ..., y_{ms}) = \mathcal{P}^{(i)}(\mathbf{x}^{(j)}, \mathbf{y}^{(k)})$$

where

$$\mathcal{P}^{(i)}(\mathbf{x}^{(j)}, \mathbf{y}^{(k)}) = \sum_{a=1}^{n}\sum_{b=1}^{m} \alpha_{a,b}^{(i)} x_{ja} y_{kb} + \beta^{(i)} = 0$$

p with elements $p_1, p_2, .., p_m$ is an MQ system.

Definition 4. (The MinRank Problem) *Let $\mathbb{F}_q$ be a finite field of order q, and m, n, k, r positive integers. Then for k matrices:*

$$M_1, M_2, ..., M_k \in \mathcal{M}_{n\times m}(\mathbb{F}_q)$$

find $\lambda_1, \lambda_2, \ldots, \lambda_k$ such that:

$$\texttt{Rank}(\sum_{l=1}^{k} \lambda_l M_l) \leq r$$

Definition 5. (Kipnis-Shamir Modeling of MinRank) *Given a MinRank(n,m,k,r) problem with $M_1, M_2, ..., M_k \in \mathcal{M}_{n\times m}(\mathbb{F}_q)$, then for some variables $\boldsymbol{x}$:*

$$\begin{pmatrix} & x_{1,1} & \cdots & x_{1,r} \\ I_{n-r} & \vdots & \ddots & \vdots \\ & x_{n-r,1} & \cdots & x_{n-r,r} \end{pmatrix} \cdot M_\lambda = 0_{(n-r)\times m}$$

generates a multilinear system of equations in $\boldsymbol{x}$ and λ.

Chapter 3

Research

In order to generate challenge instances for this project, the strategy with which to randomly generate the instances and the parameters chosen ensure the difficulty of solving the challenge instances. Therefore, the potential strategies and parameter sets must be evaluated and compared thoroughly. For instance generation, two main strategies stand out as candidates to create random systems of equations: choosing the equations fully at random and building the equations around a predetermined random solution. In the case of fully random equations, it cannot be guaranteed that any given system of equations generated has a solution. Therefore, to ensure that the challenges remain solvable, we used a predetermined random solution around which to build the equations. In this case, it is known that there exists at least one solution to the challenge. In some instances, there might be more than one solution to the system (such as underdetermined systems), in which case we can accept those as valid solutions as well. In 3.1 we outline the algorithms used to generate bilinear, multi-homogeneous, and Kipnis-Shamir systems, as well as how the random number generation works.

To decide which parameters to use as a lower and upper bounds for creating the challenges, we will use the `CryptographicEstimators` [12] library built by TII to estimate the complexity of solving the challenge types with different parameters. With these estimations, we are able to examine which parameter sets guarantee the fairness of the challenges, while ensuring that they remain difficult enough.

3.1 Algorithms

Instance Generation

To generate the challenges, we construct bilinear, multi-homogeneous, and Kipnis-Shamir system instances at random as defined in the Preliminaries. The method of instance generation differs between the three systems, and therefore each need to be outlined separately. All three instances use the same Random Number Generation method:

1. Choose a seed of length $2 \times$ security level from `urandom`.
2. Initialize `SHAKE256` with the seed as input.
3. Whenever we need a random number, pick the next 4 bytes from the `SHAKE256` hash. This is done by keeping track of the amount of bytes exhausted from the start of the hash with a variable. With this method, we avoid reusing hash bytes for RNG, which could be vulnerable to key recovery attacks through predictable values.

The method for generating the challenge instances themselves are as follows:

Bilinear MQ System:

1. Choose $s, t, m, q \in \mathbb{N}$ as parameters.
2. Choose a random vector `sol` of size $s + t$ with elements in $\mathbb{F}_q$. This will be the solution to the system.
3. Choose a random vector `coeffs` of size $s \cdot t$ with elements in $\mathbb{F}_q$.
4. Evaluate the polynomial with `sol` as the solution and `coeffs` as the coefficients and append the result as the constant γ.
5. Repeat 3. and 4. m times until the system of m equations is generated.

Multi-homogeneous System:

1. Choose $n, m \in \mathbb{N}$ as sizes of the sets and $t, s \in \mathbb{N}$ as the amount of sets.
2. Choose $n \cdot t$ and $m \cdot s$ vectors with elements in $\mathbb{F}_q$. These serve as the solution to the system.
3. Choose a random vector `coeffs` of size $n \cdot m$ with elements in $\mathbb{F}_q$.
4. Evaluate the polynomial with the selected sets as the solution and `coeffs` as the coefficients. Append the result to the polynomial as the constant $\beta^{(i)}$.
5. Repeat 3. and 4. until there are no more sets left to combine.

Kipnis-Shamir Modeling of MinRank:

1. Choose $n, m \in \mathbb{N}$ as the dimensions of the matrices., $k \in \mathbb{N}$ as the number of random matrices generated, $r \in \mathbb{N}$ as the rank of the dedicated matrix and $q \in \mathbb{N}$ as the order of the field.
2. Generate a random matrix $M_* \in \mathcal{M}_{n \times m}(\mathbb{F}_q)$ of rank r.
3. Choose a random vector λ of size $k - 1$ and elements in $\mathbb{F}_q$.
4. Generate $k - 1$ random matrices $M \in \mathcal{M}_{n \times m}(\mathbb{F}_q)$ of full rank.
5. Evaluate $M_k = -M_* + \sum_{i=1}^{k-1} \lambda_i \cdot M_i$.
6. Generate a left-kernel `lhs` for M_k as defined in (5).
7. Create a new matrix `rhs` $= M_k + \sum_{i=1}^{k-1} y_i M_i$ for unevaluated variables `y`.
8. Apply `lhs` $\times$ `rhs`. This is the Kipnis-Shamir system.
9. Sort each equation in the system in graded reverse lex order.

3.2 Benchmarking

Given that the difficulty of solving these challenges is directly tied to the parameter selection for each system, we must investigate the relative difficulty of solving different systems for their execution time. As building full solvers with benchmarking built-in for these challenges goes beyond our current scope, we use the `CryptographicEstimators` library by Esser, Verbel, Zweydinger, and Bellini [12] to estimate the time complexity for solving the given problem.

The immediate advantage of using this library to estimate the hardness of the problem is the ability to see the differences in complexity for different solving algorithms. In this case, we can account for the efficiency of certain algorithms and ensure the challenges are not too simple for those algorithms, as opposed to basing the parameters off a standard attack. For these reasons, we will choose parameters based on the algorithm that provides the lowest time complexity estimation assumed by the `CryptographicEstimators` library.

In the case of the multi-homogeneous system, this is unfortunately not possible. Therefore, to estimate the complexity of solving these types of systems, we use the estimations found for the bilinear system, while accounting for differences in upper and lower bounds for time complexity between the two system types to find the optimal parameter sets.

As we want to investigate the difficulty of solving different types of systems, we also have to consider the field they are in. The Fukuoka MQ challenge and numerous other papers cite using challenges of fields $\mathbb{F}_{31}$ and $\mathbb{F}_{256}$, [30] as they pose two interesting base fields, with 31 being a Mersenne prime and 2^8 a popular choice for a prime power. For these reasons, we will also use these two for the base fields of the challenges. In addition to these, we will also investigate solving complexity for systems in $\mathbb{F}_{16}$ as it is the field proposed for the SNOVA systems [28] and MiRa [3]. Given that there have been shown to be different degrees of difficulty for solving MQ systems with different ratios of variables to equations, we will also investigate the computational complexity of solving both underdetermined and overdetermined systems.

The `CryptographicEstimators` library offers time and memory complexity estimations for solving these problems as both $\mathbb{F}_q$ complexity and bit complexity [12]. In order to properly compare the hardness of solving these different systems, we will be focusing on the bit complexity estimations, as well as which algorithms provide the best estimations for the bilinear systems.

3.2.1 Bilinear Estimations

Below we show the estimations generated by `CryptographicEstimators` for the MQ problem, which we will use to model the difficulty of solving the bilinear MQ system. n represents the amount of total variables, while m represents the total amount of equations in the estimation. We define the parameter relations for overdetermined systems to be $2n = m$ and $1.5m = n$ for underdetermined systems.

n	m	Time	Memory	Algorithm
2	4	9.0	5.0	ExhaustiveSearch
3	6	9.0	5.0	ExhaustiveSearch
4	8	13.585	6.755	HybridF5
5	10	18.0	8.0	ExhaustiveSearch
6	12	20.418	13.966	PXL
7	14	23.013	18.165	PXL
8	16	25.151	19.861	PXL
9	18	28.827	21.084	PXL
10	20	30.332	22.558	PXL
11	22	33.673	27.066	PXL
12	24	35.637	28.467	PXL
13	26	38.918	29.637	PXL
14	28	40.344	30.912	PXL
15	30	42.35	32.082	PXL
16	32	45.552	36.905	PXL
17	34	47.286	38.075	PXL
18	36	50.563	39.212	PXL
19	38	52.002	40.307	PXL
20	40	55.112	45.284	PXL
21	42	56.903	46.389	PXL
22	44	58.563	47.436	PXL
23	46	62.268	48.552	PXL
24	48	63.832	49.547	PXL
25	50	66.6	54.703	PXL
26	52	68.459	55.717	PXL
27	54	70.272	56.687	PXL
28	56	74.02	57.787	PXL
29	58	75.664	58.718	PXL
30	60	78.436	64.012	PXL
31	62	80.351	64.965	PXL
32	64	82.194	65.884	PXL
33	66	85.809	66.971	PXL
34	68	87.455	67.858	PXL
35	70	89.408	72.999	PXL
36	72	92.261	74.169	PXL
37	74	94.064	75.05	PXL
38	76	96.973	80.418	PXL
39	78	99.015	81.32	PXL

Table 3.1: Overdetermined Complexity Estimations in $\mathbb{F}_{16}$

n	m	Time	Memory	Algorithm
4	2	9.0	6.0	ExhaustiveSearch
6	3	13.612	8.755	PXL
6	4	15.451	12.644	PXL
8	5	17.744	14.259	PXL
10	6	19.659	15.615	PXL
12	7	21.299	16.785	PXL
12	8	24.282	18.107	PXL
14	9	27.718	22.224	PXL
16	10	29.356	23.286	PXL
18	11	30.841	24.256	PXL
18	12	32.803	26.093	PXL
20	13	34.398	27.064	PXL
22	14	35.808	27.959	PXL
24	15	37.08	28.789	PXL
24	16	41.294	29.789	PXL
26	17	42.378	30.563	PXL
28	18	44.566	35.988	PXL
30	19	45.688	36.818	PXL
30	20	47.44	37.603	PXL
32	21	48.669	38.347	PXL
34	22	49.85	39.054	PXL
36	23	51.01	39.728	PXL
36	24	55.292	40.728	PXL
38	25	57.263	46.477	PXL
40	26	58.301	47.202	PXL
42	27	59.289	47.898	PXL
42	28	61.58	48.567	PXL
44	29	62.886	49.211	PXL
46	30	64.094	49.832	PXL
48	31	65.221	50.431	PXL
48	32	69.375	51.431	PXL
50	33	70.677	57.463	PXL
52	34	71.633	58.107	PXL
54	35	72.571	58.731	PXL
54	36	75.858	59.336	PXL
56	37	77.143	59.923	PXL
58	38	78.338	60.494	PXL
60	39	79.88	33.344	BooleanSolveFXL

Table 3.2: Underdetermined Complexity Estimations in $\mathbb{F}_{16}$

n	m	Time	Memory	Algorithm
2	4	11.071	6.322	F5
4	8	16.394	12.966	PXL
6	12	19.987	15.937	PXL
8	16	23.212	18.136	PXL
10	20	28.372	20.377	PXL
12	24	32.11	25.473	PXL
14	28	34.806	27.077	PXL
16	32	38.083	29.387	PXL
18	36	40.176	30.713	PXL
20	40	44.303	32.545	PXL
22	44	48.19	36.918	PXL
24	48	51.242	39.316	PXL
26	52	53.108	40.43	PXL
28	56	55.225	42.371	PXL
30	60	60.164	44.048	PXL
32	64	62.498	24.471	BooleanSolveFXL
34	68	64.346	27.744	BooleanSolveFXL
36	72	65.543	28.264	BooleanSolveFXL
38	76	66.681	28.757	BooleanSolveFXL
40	80	72.183	28.996	BooleanSolveFXL
42	84	74.088	32.322	BooleanSolveFXL
44	88	75.192	32.809	BooleanSolveFXL
46	92	76.251	33.277	BooleanSolveFXL
48	96	81.719	33.503	BooleanSolveFXL
50	100	83.673	36.871	BooleanSolveFXL
52	104	84.712	37.335	BooleanSolveFXL
54	108	85.715	37.784	BooleanSolveFXL
56	112	86.685	38.218	BooleanSolveFXL
58	116	92.112	38.429	BooleanSolveFXL

Table 3.3: Overdetermined Complexity Estimations in $\mathbb{F}_{31}$

n	m	Time	Memory	Algorithm
4	2	10.571	5.322	HybridF5
6	4	19.987	16.522	PXL
10	6	25.253	18.487	PXL
12	8	30.606	24.418	PXL
16	10	36.369	27.388	PXL
18	12	41.175	33.293	PXL
22	14	45.133	35.905	PXL
24	16	50.762	38.397	PXL
28	18	55.4	44.343	PXL
30	20	61.977	46.711	PXL
34	22	66.389	52.741	PXL
36	24	70.444	54.816	PXL
40	26	75.287	60.83	PXL
42	28	81.335	63.141	PXL
46	30	86.08	69.222	PXL
48	32	91.847	75.26	PXL
52	34	95.356	77.202	PXL
54	36	101.26	79.492	PXL
58	38	106.196	85.616	PXL

Table 3.4: Underdetermined Complexity Estimations in $\mathbb{F}_{31}$

n	m	Time	Memory	Algorithm
2	4	12.454	7.0	F5
4	8	19.862	12.853	Crossbred
6	12	23.63	15.545	Crossbred
8	16	26.581	18.814	PXL
10	20	30.578	24.286	PXL
12	24	33.492	26.151	PXL
14	28	36.188	27.755	PXL
16	32	42.481	33.426	PXL
18	36	44.665	34.954	PXL
20	40	46.831	19.697	BooleanSolveFXL
22	44	49.573	37.596	PXL
24	48	54.269	39.994	PXL
26	52	55.726	23.789	BooleanSolveFXL
28	56	57.068	24.359	BooleanSolveFXL
30	60	62.567	47.068	PXL
32	64	64.382	48.105	PXL
34	68	65.729	28.422	BooleanSolveFXL
36	72	66.926	28.942	BooleanSolveFXL
38	76	68.063	29.436	BooleanSolveFXL

Table 3.5: Overdetermined Complexity Estimations in $\mathbb{F}_{256}$

n	m	Time	Memory	Algorithm
4	2	12.62	7.0	F5
6	4	22.53	17.2	PXL
10	6	30.696	25.259	PXL
12	8	36.779	25.096	PXL
16	10	41.034	30.912	PXL
18	12	46.082	36.665	PXL
22	14	53.197	42.377	PXL
24	16	59.477	42.391	PXL
28	18	64.331	48.173	PXL
30	20	69.239	53.932	PXL
34	22	75.07	59.671	PXL
36	24	82.228	65.394	PXL
40	26	87.821	64.853	PXL
42	28	92.918	70.687	PXL
46	30	97.97	76.498	PXL
48	32	103.244	82.29	PXL
52	34	110.481	88.064	PXL
54	36	116.57	87.256	PXL
58	38	121.665	93.133	PXL

Table 3.6: Underdetermined Complexity Estimations in $\mathbb{F}_{256}$

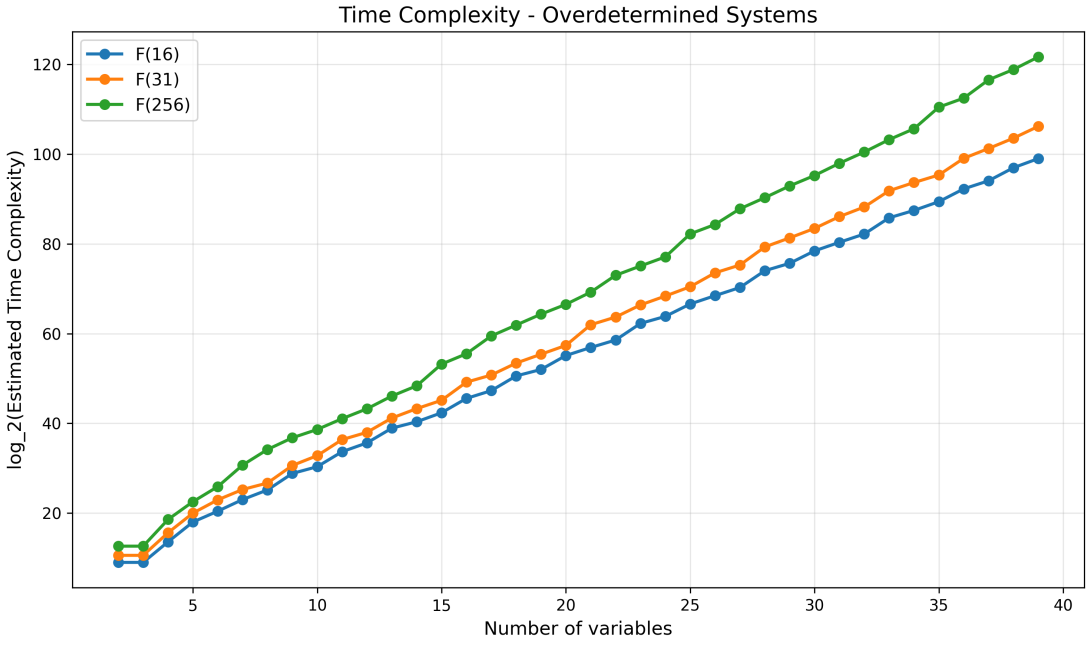

Figure 3.7: Time Comparison for Overdetermined Systems

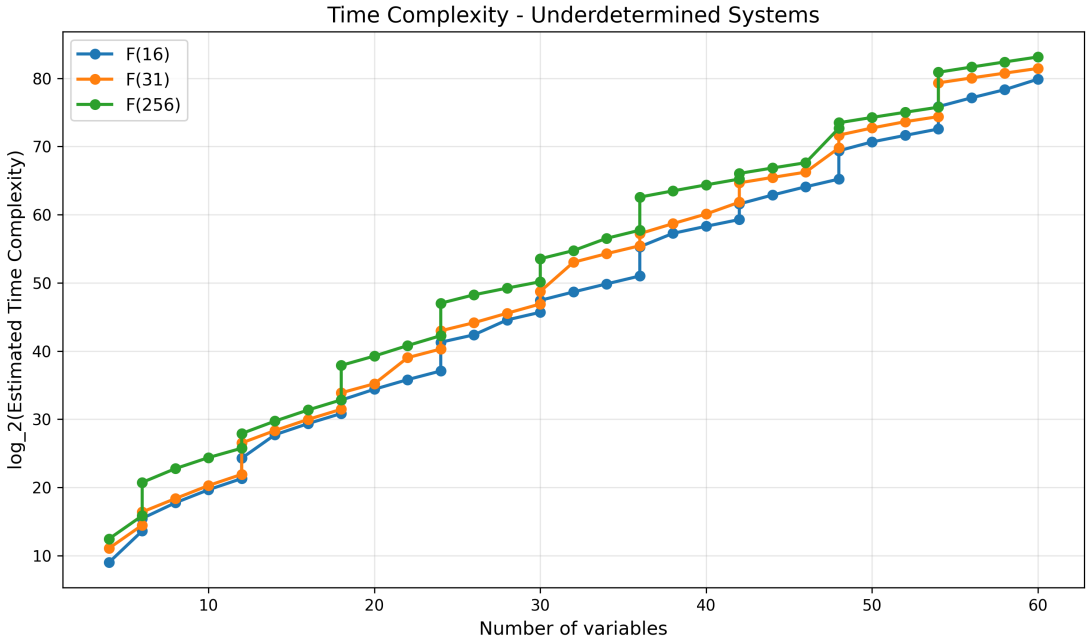

Figure 3.8: Time Comparison for Underdetermined Systems

Figure 3.7 and 3.8 show the differences in time complexity for over- and underdetermined bilinear MQ systems with different fields. It is clear in both figures that time complexity increases vaguely linearly for systems of all three fields, with a regular staircase structure for underdetermined systems. It can also be noted that for both overdetermined and underdetermined systems, systems in $\mathbb{F}_{16}$ have the lowest complexity for the same number of variables, followed by systems in $\mathbb{F}_{31}$ and finally systems in $\mathbb{F}_{256}$. Using this trend, we can safely estimate what the bit complexity to solve would be for instances with much larger numbers of variables, such as determining lower bounds for schemes based on a bilinear underlying structure. It should also be noted that the bit complexity to solve differs marginally between systems with different fields but equal numbers of variables and equations. This can be attributed to multiplication in large fields taking more bit operations to complete when compared to smaller fields.

It can also be noted in 3.1-3.6 that `PXL` regularly performs best for underdetermined systems, while in overdetermined systems, `BooleanSolveFXL` outperforms `PXL` in some cases for large m. In the case of overdetermined systems in $\mathbb{F}_{16}$ however, `PXL` continues to perform best compared to other solving algorithms. However, it should be taken into account that the memory complexity in cases where `BooleanSolveFXL` beats `PXL` on time is significantly lower than when `PXL` performs best, indicating that `PXL` might not be the optimal solving algorithm for some parameter sets under certain conditions.

3.2.2 MinRank Estimations

Currently, there does not exist a simple way to estimate the hardness of solving MinRank instances under the Kipnis-Shamir modeling. Therefore, we use the standard algorithms provided by the `CryptographicEstimators` library to estimate overall time complexity of solving Min-

Rank, with the caveat that the real hardness to solve under the Kipnis-Shamir modeling will differ based on the solving algorithms used.

Since there are many parameters that contribute to the difficulty of solving MinRank, certain parameters have to be fixed to allow for the differences in difficulty to be distinguishable. For these reasons, we will investigate how changing different parameter types affects the the estimated time complexity to solve MinRank instances:

- **Field:** $\mathbb{F}_{31}$, $\mathbb{F}_{2^8}$, $\mathbb{F}_{16}$
- **Number of Rows:** 4 to 20
- **Number of Matrices:** 20, 50, 120

To find sample parameters for estimating the complexity of solving MinRank, we use the parameters proposed by the MiRa specification as a reference. They expect 128 bits of security [3] for instances with the following parameters:

- $q = 16$
- $n, m = 16$
- $k = 120$
- $r = 5$

for q the order of the field, n, m the dimension of the matrix, k the number of matrices and r the rank of the resulting matrix. We will use these parameters as a strict upper bound for the estimations, as we want to guarantee that the challenges are possible to solve.

From the estimations, it is clear that the rank of the matrix instance compared to the size of the matrix can significantly impact the difficulty in solving the instance. Therefore, we estimate the complexity for solving MinRank instances not only in terms of matrix size and the total number of matrices, but also for the ranks of the matrices. Here, the most interesting cases are low ranks (matrices of rank $r = k$, for small k in comparison to matrix size), middle ranks (matrices of rank roughly $\frac{\text{Rows}}{2}$), and high ranks (matrices of rank $r = Rows - k$ for some $k \geq 1$). However, as deciding which ranks can be considered "low", "middle", and "high" is based on mostly arbitrary choices, we will look at how time complexity scales according to a linear increase in matrix rank when the other parameters are fixed.

It should be noted that for these estimations we are primarily looking at square matrices. Non-square matrices could also pose as interesting instances, however their estimated complexities do not differ substantially enough from their square counterparts to be considered as a distinguishable parameter type for the challenges we create.

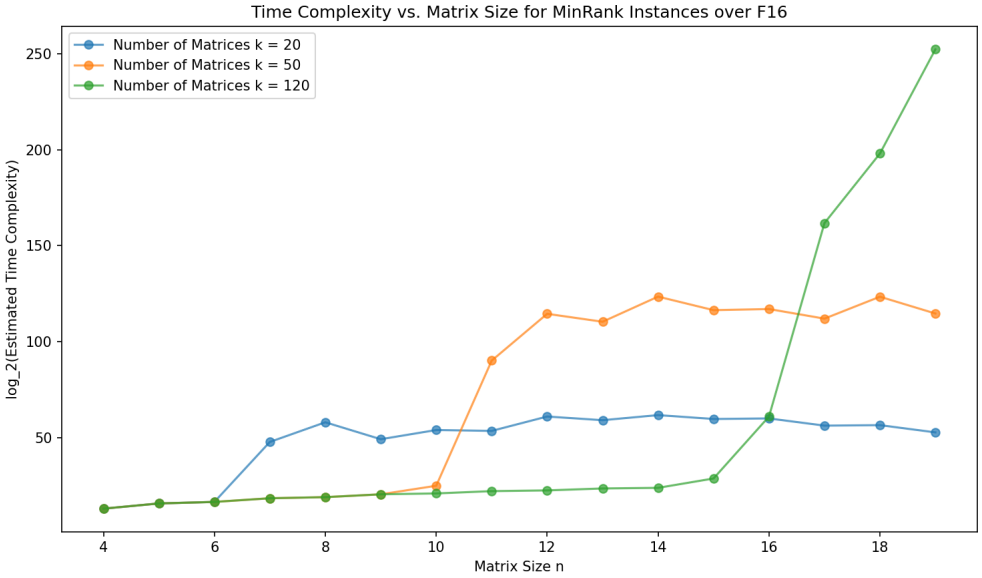

Figure 3.9: Performance comparison for MinRank instances in $\mathbb{F}_{16}$ with increasing matrix sizes

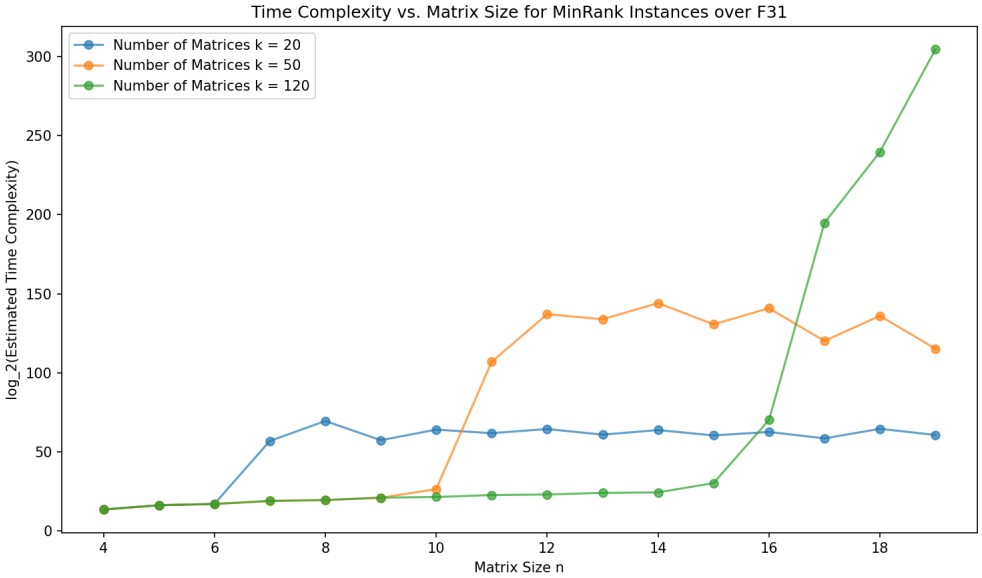

Figure 3.10: Performance comparison for MinRank instances in $\mathbb{F}_{31}$ with increasing matrix sizes

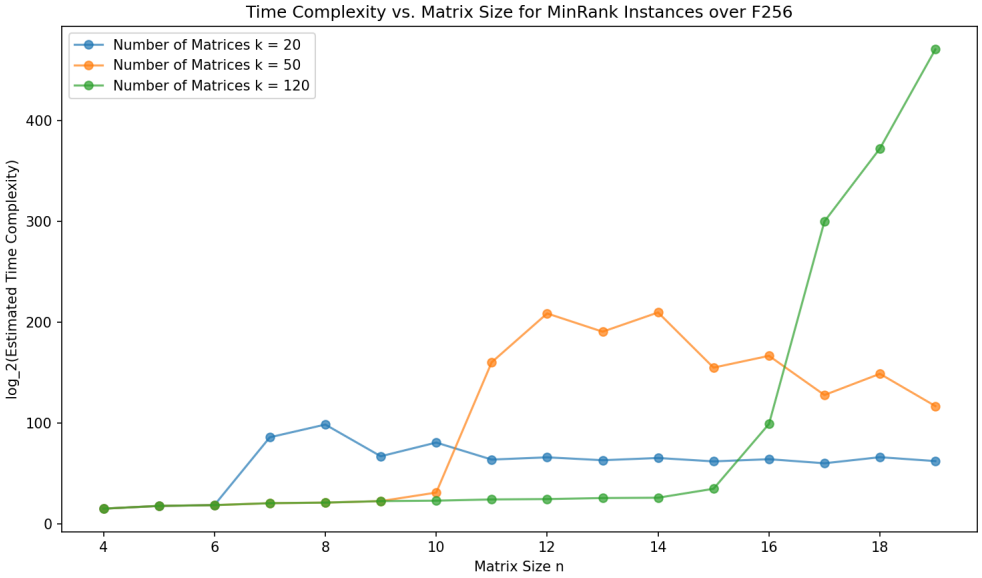

Figure 3.11: Performance comparison for MinRank instances in $\mathbb{F}_{256}$ with increasing matrix sizes

Figures 3.9-3.11 show how complexity to solve MinRank instances links to both matrix size and the number of matrices k in the instance. In this case, the rank of each MinRank instance was taken as $\frac{n}{2}$, with n the size of the matrices. In all three fields, the estimations demonstrate the same trends for how matrix size and k relate to time complexity, showing that the size of the field itself is not the primary factor of complexity to solve for any given MinRank instance. However, it is also important to note that the complexity trends seen in $\mathbb{F}_{16}$ are roughly half of those in $\mathbb{F}_{256}$, which indicates that a larger field order can amplify the number of operations needed to solve an instance, but does not introduce any exploitable factors that would significantly change solving strategy. This can likely be attributed to the cost of multiplication being higher over larger fields, like seen in the complexity estimations represented by Figure 3.8. In comparison, the number of matrices k seem to directly correlate to how easy it is to solve MinRank instances with different matrix sizes n. It is also important to note that in all three figures, there is a "limit" trend for instances with $k = \{20, 50\}$, where the time complexity does not increase beyond a certain threshold, which is important to take into account to not unintentionally generate challenges that are too easy.

Therefore, choosing parameters for challenges based on MinRank systems do not need to purely rely on field order to generate challenges, but rather to make the challenges easier or harder based on the other parameters already chosen. In the following figures, we will look at how matrix rank and the number of matrices k relates to complexity to solve in order to make better choices for the parameter sets for the MinRank challenges.

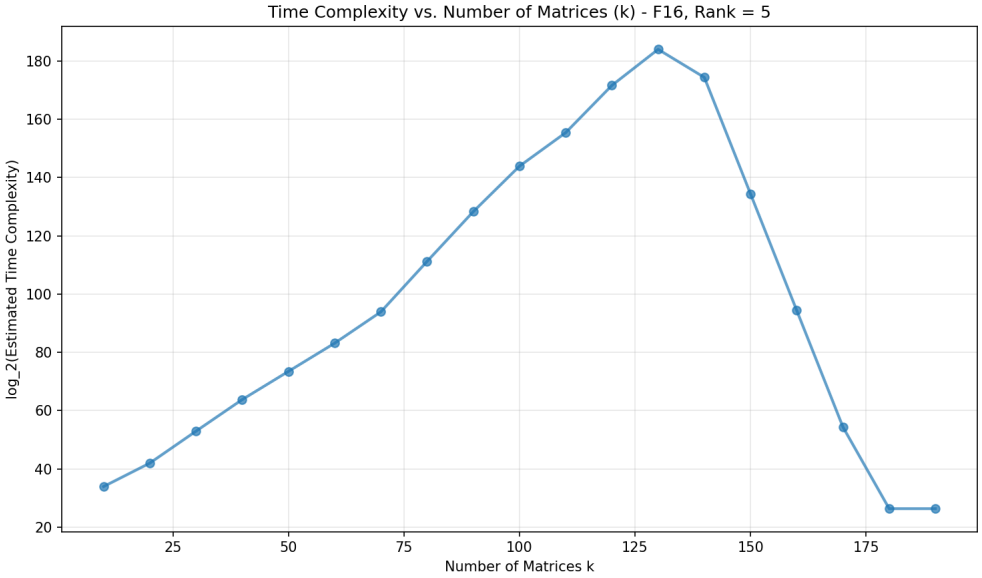

Figure 3.12: Performance comparison for MinRank instances in $\mathbb{F}_{16}$ with increasing number of matrices k

Figure 3.12 shows how the number of matrices affects time complexity for fixed matrix size and rank. In this case, we took the matrix size and rank from the MiRa specification ($n = 16, r = 5$) [3] as reference values. Like the data represented by Figure 3.13, there is linear increase until a pivot point for k before a sharp decline in estimated complexity to solve. This peak is referred to in some literature as the Rank Gilbert-Varshanov Bound [3], where $k = (n-r)(m-r)$ for an $n \times m$ matrix. This bound determines the complexity to solve any generalized MinRank instance, as for $k > (n-r)(m-r)$, polynomials in the MinRank instance are underdetermined and the number of solutions grows exponentially [14]. At and below the Rank GV Bound ($k \leq (n-r)(m-r)$), the instance is said to be a 0-dimensional system and has a finite number of solutions, which means that for k approaching the bound, complexity to solve increases linearly with k. Therefore, for any parameter set, k near the Rank GV bound offers the highest complexity when the other parameters are fixed.

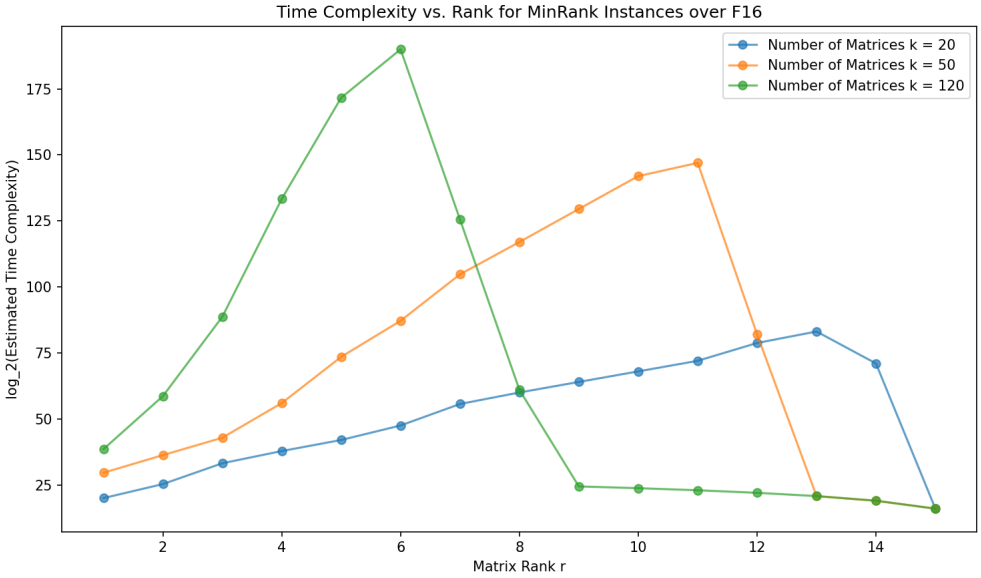

Figure 3.13: Performance comparison for MinRank instances in $\mathbb{F}_{16}$ with increasing rank

Figure 3.13 shows how the determined rank of the MinRank instance relates to time complexity and the difference in this relation for instances with different numbers of matrices k. In this case, the estimations were based on MinRank instances with matrix size $n = 16$. The data represented here shows how the Rank Gilbert-Varshanov bound responds to changes in rank, as for smaller k the bound is approached at much higher ranks than for higher k. Given that Figures 3.11 and 3.12 show how matrix rank and k are directly tied to each other and seem to be the primary factors for determining the complexity to solve any given MinRank instance, we will choose parameters by fixing some matrix size n and rank r, and then using the trends shown in Figures 3.11 and 3.12 to adjust k and scale the difficulty of the challenges where necessary.

3.2.3 Multi-homogeneous Systems

As in the case of the Kipnis-Shamir system, there does not exist a direct way to estimate the time complexity to solve multi-homogeneous systems. However, given that they are closely related to the bilinear MQ system, we will use the estimation data from the MQ estimations to gain insight into which parameter sets would be best suited for this project. In this case, we can assume that the complexity to solve remains roughly the same for the same n, m as the bilinear system, with the caveat that different set sizes for the input variables will also lead to a significant change in solve complexity (smaller set sizes lead to higher solving time and vice versa). For these reasons, we aim to keep set sizes relatively small, to keep complexity generally in line with the estimations made in 3.2.1. It should also be noted that for multi-homogeneous systems, the number of equations m is directly related to the number of input variables n and the set sizes chosen, so instead of choosing m directly, we derive it from the relation between the set size and input size.

3.3 Parameter Selection

To ensure that the challenges are difficult enough, we will look at parameter combinations with a time complexity of 2^{60} operations and higher. This choice is made with the intent of having a similar level of difficulty as the Fukuoka MQ Challenge, where for overdetermined systems in $\mathbb{F}_{31}$, they extrapolated that solving for $n > 34$ would require more than a month on their hardware [30]. In our estimations, the same parameter set has an estimated bit-time complexity of 2^{64}. Therefore, in the interest of rounding, we chose to include challenges where the parameter sets have an estimated complexity of 2^{60} and higher. This ensures that the challenges are still solvable, but leaves enough room for new approaches to attacks to be developed. For the Kipnis-Shamir systems, the rank of the matrices and the number of matrices used also play a large role in the time complexity to solve. Therefore, we will take minimum rank and k below the Rank Gilbert-Varshanov Bound such that they can be scaled towards the bound to make more difficult challenges. The matrix sizes and field orders themselves will be chosen according to the bit-time complexity limit of 2^{60}.

3.3.1 Runtime Estimations

Beyond the bit-time complexity limit we chose, we are also interested in the estimated runtime for solving challenges with the minimum parameter sets. This can be found by calculating the average amount of CPU cycles required on a single core to perform a single field operation. This allows us to use the conversion between bit-time complexity and field operations complexity provided by the `CryptographicEstimators` library to estimate runtime.

For x86 CPU's, it was shown by Chen et al. in their 2009 paper [10] that lookup tables are the most optimal method of implementing finite field multiplication in MPKC implementations, as they require frequent and repetitive multiplication operations. This method was then extended to AVX2 instructions by Beullens et al in their Classical Oil and Vinegar implementation [7].

Although Galois Field New Instructions (GF-NI) were introduced with the implementation of AVX512 in Intel CPU's in 2016, AVX2 and SSE implementations of finite field multiplication seem to be more thoroughly researched due to their wider availability in modern CPU's [7]. In order to estimate runtime for minimal challenge parameters, we will use the AVX2 implementation by Beullens et al. [7] for challenges in $\mathbb{F}_{16}$ and $\mathbb{F}_{256}$. While an SSE implementation for multiplication in $\mathbb{F}_{31}$ by lookup tables is described by Chen et al. [10], a C implementation of this method is not widely available. Therefore, we can calculate an upper bound estimate for the latency of generating a lookup table in $\mathbb{F}_{31}$ by naïve modular multiplication for each entry.

Calculating the estimated runtime for a large number total operations requires knowledge of the average clock cycles per operation as well as the CPU frequency. As lookup tables remain the most optimal way to do repetitive multiplication over finite fields [7], finding the average cycles per multiplication requires benchmarking the clock cycles required to generate the LUT's as well as the clock cycles required to access the result of a single multiplication in the LUT (although the latency of the LUT accesses would likely be optimized to require fewer accesses by reading LUT entries in batches instead of individually).

In the case of lookup table generation in $\mathbb{F}_{31}$, we simply measure the clock cycles required to generate a full lookup table with modular multiplication. However, it should be noted that an optimized multiplication implementation such as that described by Chen et al. would achieve significant speedups compared to naïve modular multiplication, although not as fast as generating lookup tables for multiplication in $\mathbb{F}_{2^s}$ [10].

We benchmarked these operations on an Intel Core i7-13700H processor with turbo boost disabled. The non-turbo clock speed was measured at 2.918 GHz, and the benchmark was ran

on a Fedora 43 system with `clang 21.1.8`. The average was taken by looking at the total amount of CPU cycles required to complete 1,000,000 operations. For the $\mathbb{F}_{31}$ lookup table generation, the result was measured as the average cycles of generating 1,000,000 lookup tables with entries calculated sequentially.

Field	Average Cycles per single LUT generation
$\mathbb{F}_{16}$	4.629
$\mathbb{F}_{31}$	1476.21
$\mathbb{F}_{256}$	3.394

Table 3.14: Average CPU Cycles Required to Generate a Single Multiplication Table in $\mathbb{F}_q$

For $\mathbb{F}_{16}$ and $\mathbb{F}_{256}$, 16 and 256 LUT's respectively are required in total to precompute all field multiplications prior to them being required. This simplifies multiplications later to simple L1 cache calls. In Beullens et al.'s implementation, table generation is optimized by taking advantage of AVX2 vector operations, which allows for multiple tables to be generated in parallel. In the case of $\mathbb{F}_{16}$, all 16 tables can be generated in a single call to the function `gf16v_generate_multab_16_avx2`, while for $\mathbb{F}_{256}$, 16 calls to `gf256v_generate_multabs_avx2` are required, as it only generates 16 tables at a time.

For $\mathbb{F}_{31}$, we found that for naïve modular multiplication, each entry would require on average 1.536 clock cycles to complete. This is much slower than the creation of entries in lookup tables for $\mathbb{F}_{16}$ and $\mathbb{F}_{256}$, which can be attributed to the implementation of this benchmark not being very thorough or using any well-known optimization techniques such as those proposed by Beullens et al. [7] and Chen et al. [10]. As discussed earlier, an optimized lookup table generator for $\mathbb{F}_{31}$ would still likely be slower than those of $\mathbb{F}_{16}$ and $\mathbb{F}_{256}$, as modular reduction in binary fields is much more efficient than those of odd prime fields [10].

We then assume that 2^{60} L1 cache accesses are required for each multiplication, since the multiplication tables fit cleanly into the L1 cache for most modern CPU's. In the case of most Intel architecture, L1 cache latency has been measured at roughly 4 cycles [2]. Therefore, the average cost of multiplication can be calculated as the average of precomputing the LUT's and accessing the results of required multiplications across the space of the total number of operations.

In `CryptographicEstimators`, the conversion between field operations complexity and bit complexity is taken as bit-time complexity = (field operations complexity)$(\log q)^\theta$ for some $\theta \in [0, 2]$ [12]. In the case of the MQ estimator, $\theta = 0$ by default [4]. Therefore we can use the bit-time complexity estimations presented in Section 3.2 as the total number of operations to extrapolate the runtime for solving challenges with the minimal parameter sets.

Given that the number of cycles required to generate the LUT's compared to the clock cycles required to read the results of each multiplication is negligible, the bottleneck for multiplication in this case is the cost of L1 cache accesses and the average cycles per multiplication over any relatively small finite field is roughly equal to the cost of reading the LUT's. For the purpose of the estimations, we will calculate the estimated runtime in seconds as:

$$\frac{\text{LUT generation} + (\text{L1 cache latency} \times \text{total operations})}{\text{CPU frequency}}$$

For a single core on an i7-13700H processor with a non-turbo frequency of 2.918 GHz, challenges with an estimated time complexity of 2^{60} operations require:

Field	Estimated Runtime in Seconds	Estimated Runtime in Years
$\mathbb{F}_{16}$	1.471×2^{30}	50.1
$\mathbb{F}_{31}$	1.471×2^{30}	50.1
$\mathbb{F}_{256}$	1.471×2^{30}	50.1

Table 3.15: Estimated Runtime for Bit-time Complexity of 2^{60} on a Single Core

3.4 Challenge Format

The challenges are presented in a similar format as those from the Fukuoka MQ Challenge [30]. In the case of the multi-homogeneous and Kipnis-Shamir challenges, other challenge parameters are also specified in the instance. Below we show example challenges for each type:

```
Galois Field : GF(16) / x^4 + x + 1
Number of x variables (nx) : 2
Number of x variables (ny) : 3
Number of equations (m) : 10
Order : graded reverse lex order

**********************
7 1 7 12 1 13 15 0 11 15 7 11 ;
11 13 1 2 7 2 13 12 3 2 15 10 ;
11 10 5 8 14 3 14 13 4 12 1 1 ;
6 8 15 3 3 15 8 9 3 14 14 8 ;
5 5 12 3 8 5 4 0 6 6 13 1 ;
7 5 13 15 6 8 10 7 1 0 14 9 ;
0 8 8 3 14 11 6 1 1 10 8 4 ;
11 5 11 13 11 3 5 10 13 14 1 1 ;
9 10 5 14 15 12 11 7 7 10 14 9 ;
6 5 11 14 2 0 15 5 11 4 11 7 ;
```

Figure 3.16: Example Bilinear Challenge over $\mathbb{F}_{16}$

```
Galois Field : GF(31) / x + 28
Number of sets of x variables (t) : 2
Number of variables in an x set (n) : 3
Number of sets of y variables (s) : 2
Number of variables in a y set (m) : 3
Number of equations : 4
Order : graded reverse lex order

**********************
25 20 29 25 18 8 28 18 23 11 ;
8 27 3 17 29 17 6 7 5 24 ;
6 27 25 2 17 24 12 5 21 1 ;
13 0 19 27 5 5 23 21 27 16 ;
```

Figure 3.17: Example Multi-homogeneous Challenge over $\mathbb{F}_{31}$

For both the bilinear and multi-homogeneous challenges, the expected solution format is $[x$ variables$, y$ variables$]$.

```
Galois Field : GF(256) / x^8 + x^4 + x^3 + x^2 + 1
Matrix Dimensions : 5x5
Matrix Rank : 3
Number of Matrices (k) : 3
Order : graded reverse lex order

**********************
21 149 152 228 229 236 226 7 22 235 237 132 ;
152 7 212 236 66 107 21 228 154 37 183 169 ;
212 228 229 107 46 75 152 236 20 52 184 56 ;
229 236 66 75 153 205 212 107 159 248 88 31 ;
66 107 46 205 108 83 229 75 76 156 123 212 ;
21 149 152 228 229 236 65 212 22 235 237 135 ;
152 7 212 236 66 107 149 229 154 37 183 136 ;
212 228 229 107 46 75 7 66 20 52 184 140 ;
229 236 66 75 153 205 228 46 159 248 88 63 ;
66 107 46 205 108 83 236 153 76 156 123 128 ;
```

Figure 3.18: Example Kipnis-Shamir Challenge over $\mathbb{F}_{2^8}$

In the case of the Kipnis-Shamir challenges, the expected solution format is

$$[\lambda \text{ variables}, x_{0,0}, ..., x_{0,r}, ..., x_{n-r,1}, ..., x_{n-r,r}]$$

where the x variables are the kernel variables as defined in the definition of the Kipnis-Shamir modeling in the Preliminaries.

Chapter 4

Conclusions

In this thesis, we have built a challenge for structured MQ systems that give an opportunity to the cryptographic community to develop new solving strategies against the selected system types. After a thorough investigation into the expected time to solve for different parameter sets for these systems, we have decided on the minimum parameters for the challenges as the following:

Overdetermined Systems		
Field	Parameters	Estimated Runtime in Seconds
$\mathbb{F}_{16}$	(23,46)	1.77×2^{32}
$\mathbb{F}_{31}$	(30,60)	1.67×2^{30}
$\mathbb{F}_{256}$	(30,60)	1.09×2^{33}

Underdetermined Systems		
Field	Parameters	Estimated Runtime in Seconds
$\mathbb{F}_{16}$	(42,28)	1.1×2^{32}
$\mathbb{F}_{31}$	(30,20)	1.44×2^{32}
$\mathbb{F}_{256}$	(28,18)	1.85×2^{34}

Table 4.1: Minimum Parameters (n, m) for Bilinear MQ Challenges

Overdetermined Systems		
Field	Parameters	Estimated Runtime in Seconds
$\mathbb{F}_{16}$	(24,48)	1.31×2^{34}
$\mathbb{F}_{31}$	(30,60)	1.67×2^{30}
$\mathbb{F}_{256}$	(30,60)	1.09×2^{33}

Underdetermined Systems		
Field	Parameters	Estimated Runtime in Seconds
$\mathbb{F}_{16}$	(42,28)	1.1×2^{32}
$\mathbb{F}_{31}$	(30,20)	1.44×2^{32}
$\mathbb{F}_{256}$	(28,18)	1.85×2^{34}

Table 4.2: Minimum Parameters (n, m) for Multi-homogeneous MQ Challenges

Field	(n, m, r, k)	Estimated Runtime in Seconds	(n, m, r, k)	Estimated Runtime in Seconds
$\mathbb{F}_{16}$	(16,16,5,39)	1.34×2^{31}	(12,12,5,27)	1.97×2^{32}
$\mathbb{F}_{31}$	(16,16,5,39)	1.98×2^{32}	(12,12,5,23)	1.3×2^{31}
$\mathbb{F}_{256}$	(12,12,5,22)	1.61×2^{30}	(10,10,4,24)	1.07×2^{34}

Table 4.3: Minimum Parameters (n, m, r, k) for Kipnis-Shamir Challenges

These parameter choices were made using estimations from the `CryptographicEstimators` library for the time complexity to solve these systems. The goal of the project is to develop algorithms that are able to regularly beat these estimations, furthering the field of MQ solving algorithms for structured systems often seen in attacks against MPKC's.

Bibliography

[1] TII McEliece Challenge, 2023. URL: https://crowdchallenge.tii.ae/mceliece-challenges/index.

[2] Andreas Abel and Jan Reineke. uops.info: Characterizing latency, throughput, and port usage of instructions on intel microarchitectures. In *ASPLOS*, ASPLOS '19, pages 673–686, New York, NY, USA, 2019. ACM. URL: http://doi.acm.org/10.1145/3297858.3304062, doi:10.1145/3297858.3304062.

[3] Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesus-Javier Chi-Dominguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, Matthieu Rivain, and Jean-Pierre Tillich. MIRA Specifications.

[4] Emanuele Bellini, Rusydi H. Makarim, Carlo Sanna, and Javier Verbel. An estimator for the hardness of the mq problem. In *Progress in Cryptology - AFRICACRYPT 2022: 13th International Conference on Cryptology in Africa, AFRICACRYPT 2022, Fes, Morocco, July 18–20, 2022, Proceedings*, page 323–347, Berlin, Heidelberg, 2022. Springer-Verlag. doi:10.1007/978-3-031-17433-9_14.

[5] Ryad Benadjila, Thibauld Feneuil, and Matthieu Rivain. MQ on my Mind: Post-Quantum Signatures from the Non-Structured Multivariate Quadratic Problem . In *2024 IEEE 9th European Symposium on Security and Privacy (EuroSP)*, pages 468–485, Los Alamitos, CA, USA, July 2024. IEEE Computer Society. URL: https://doi.ieeecomputersociety.org/10.1109/EuroSP60621.2024.00032, doi:10.1109/EuroSP60621.2024.00032.

[6] Ward Beullens. Mayo: Practical post-quantum signatures from oil-and-vinegar maps. In Riham AlTawy and Andreas Hülsing, editors, *Selected Areas in Cryptography*, pages 355–376, Cham, 2022. Springer International Publishing.

[7] Ward Beullens, Ming-Shing Chen, Shih-Hao Hung, Matthias J. Kannwischer, Bo-Yuan Peng, Cheng-Jhih Shih, and Bo-Yin Yang. Oil and vinegar: Modern parameters and implementations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):321–365, Jun. 2023. URL: https://tches.iacr.org/index.php/TCHES/article/view/10966, doi:10.46586/tches.v2023.i3.321-365.

[8] Andreas Björklund, Petteri Kaski, and Ryan Williams. Solving Systems of Polynomial Equations over GF(2) by a Parity-Counting Self-Reduction. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:13, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.

de/entities/document/10.4230/LIPIcs.ICALP.2019.26, doi:10.4230/LIPIcs.ICALP.2019.26.

[9] Antoine Casanova, Jean-Charles Faugère, Gilles Macario-Rat, Jacques Patarin, Ludovic Perret, and Jocelyn Ryckeghem. GeMSS: A Great Multivariate Short Signature. Research report, UPMC - Paris 6 Sorbonne Universités ; INRIA Paris Research Centre, MAMBA Team, F-75012, Paris, France ; LIP6 - Laboratoire d'Informatique de Paris 6, December 2017. URL: https://inria.hal.science/hal-01662158.

[10] Anna Inn-Tung Chen, Ming-Shing Chen, Tien-Ren Chen, Chen-Mou Cheng, Jintai Ding, Eric Li-Hsiang Kuo, Frost Yu-Shuang Lee, and Bo-Yin Yang. Sse implementation of multivariate pkcs on modern x86 cpus. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, pages 33–48, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[11] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 164–175, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[12] Andre Esser, Javier A. Verbel, Floyd Zweydinger, and Emanuele Bellini. {SoK: CryptographicEstimators - a Software Library for Cryptographic Hardness Estimation}. In *AsiaCCS*. {ACM}, 2024.

[13] Jean Charles Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero (f5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, ISSAC '02, page 75–83, New York, NY, USA, 2002. Association for Computing Machinery. doi:10.1145/780506.780516.

[14] Jean-Charles Faugère, Mohab Safey El Din, and Pierre-Jean Spaenlehauer. On the complexity of the generalized minrank problem. *Journal of Symbolic Computation*, 55:30–58, 2013. URL: https://www.sciencedirect.com/science/article/pii/S0747717113000485, doi:10.1016/j.jsc.2013.03.004.

[15] Jean-Charles Faugére. A new efficient algorithm for computing gröbner bases (f4). *Journal of Pure and Applied Algebra*, 139(1):61–88, 1999. URL: https://www.sciencedirect.com/science/article/pii/S0022404999000055, doi:10.1016/S0022-4049(99)00005-5.

[16] Hiroki Furue and Momonari Kudo. Polynomial XL: A variant of the XL algorithm using macaulay matrices over polynomial rings. In *Post-Quantum Cryptography: 15th International Workshop, PQCrypto 2024, Oxford, UK, June 12–14, 2024, Proceedings, Part II*, page 109–143, Berlin, Heidelberg, 2024. Springer-Verlag. doi:10.1007/978-3-031-62746-0_6.

[17] Michael Garey and David Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. A Series of Books in the Mathematical Sciences. W.H. Freeman and Company, 1979.

[18] F. Göpfert and A. Yakkundimath. LWE Challenge, 2015. URL: https://latticechallenge.org/lwe_challenge/index.php.

[19] Antoine Joux and Vanessa Vitse. A crossbred algorithm for solving boolean polynomial systems. In Jerzy Kaczorowski, Josef Pieprzyk, and Jacek Pomykała, editors, *Number-Theoretic Methods in Cryptology*, pages 3–21, Cham, 2018. Springer International Publishing.

[20] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 206–222, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[21] R. Lindner, M. Rueckert, P. Baumann, and L. Nobach. Lattice challenge, 2010. URL: https://latticechallenge.org/.

[22] Kai-Chun Ning, Lars Ran, and Simona Samardjiska. Multi-homogeneous XL. Cryptology ePrint Archive, Paper 2025/2060, 2025. URL: https://eprint.iacr.org/2025/2060.

[23] T. Plantard and M. Schneider. Ideal lattice challenge, 2012. URL: https://latticechallenge.org/ideallattice-challenge/index.php.

[24] M. Schneider and N. Gama. SVP Challenge, 2010. URL: https://latticechallenge.org/svp-challenge/index.php.

[25] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. arXiv:https://doi.org/10.1137/S0097539795293172, doi:10.1137/S0097539795293172.

[26] Xijin Tang and Yong Feng. A new efficient algorithm for solving systems of multivariate polynomial equations. Cryptology ePrint Archive, Report 2005/312, 2005. URL: https://eprint.iacr.org/2005/312.

[27] The Bochum Challenges Team. Bochum challenges, June 2026. Available at https://bochum-challeng.es. URL: https://bochum-challeng.es.

[28] Lih-Chung Wang, Po-En Tseng, Yen-Liang Kuan, and Chun-Yen Chou. A simple non-commutative UOV scheme. Cryptology ePrint Archive, Paper 2022/1742, 2022. URL: https://eprint.iacr.org/2022/1742.

[29] Bo-Yin Yang, Wei-Jeng Wang, Shang-Yi Yang, Char-Shin Miou, and Chen-Mou Cheng. Fast exhaustive search for polynomial systems over f3. Cryptology ePrint Archive, Paper 2023/731, 2023. URL: https://eprint.iacr.org/2023/731.

[30] Takanori Yasuda, Xavier Dahan, Yun-Ju Huang, Tsuyoshi Takagi, and Kouichi Sakurai. MQ challenge: Hardness evaluation of solving multivariate quadratic problems. Cryptology ePrint Archive, Paper 2015/275, 2015. URL: https://eprint.iacr.org/2015/275.