The MinRank Problem: Survey, Implementation, and One Application

Dario Gjorgjevski
gjorgjevski.dario@students.finki.ukim.mk

Project Work in Cryptography

2015/2016

1 Introduction

Multivariate cryptography is the generic term for asymmetric cryptographic primitives based on multivariate polynomials over a finite field $\mathbb{F}_q$. In the case when the degree of these polynomials is 2, we refer to them as multivariate quadratic ($\mathcal{MQ}$). Note that the quadratic case is as complex as the general case (up to a polynomial-time reduction). The most interesting one-way function in this context is the evaluation of multivariate polynomials. Namely, given $\mathbf{f} = (f_1(x_1, \ldots, x_n), \ldots, f_\eta(x_1, \ldots, x_n)) \in \mathbb{K}[x_1, \ldots, x_n]^\eta$ (where $\mathbb{K}$ is the ground field), the one-way function is defined as follows:

$$F : \mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{K}^n \mapsto (f_1(x_1, \ldots, x_n), \ldots, f_\eta(x_1, \ldots, x_n)) \,. \tag{1}$$

The problem of inverting (1) can be shown to be NP-hard by noticing that there is a trivial polynomial-time reduction from well-known NP-complete problems such as 3-SAT.

The MinRank problem (MR) is a fundamental problem in linear algebra of finding a low-rank linear combination of matrices. It was first stated and had its complexity analyzed in [BFS99]. The importance of the MinRank problem in cryptography stems from the fact that it was shown to be very closely related to several $\mathcal{MQ}$ cryptosystems such as HFE [Pat96; KS99] and TTM [Moh99; GC00]. In fact, as we shall see in section 2.1.2, MR instances can themselves be modeled as $\mathcal{MQ}$ systems.

In this essay I will attempt to provide a survey of the MinRank problem and some known methods for solving it, as well as look at the idea of using MR instances as the basis of a *zero-knowledge authentication protocol*.

2 The MinRank Problem

Definition 1 (MinRank over a field)**.** Let $\mathbf{M}_0; \mathbf{M}_1, \ldots, \mathbf{M}_m$ be matrices in $\mathcal{M}_{\eta \times n}(\mathbb{K})$. The MinRank problem instance $\mathsf{MR}(m, \eta, n, r, \mathbb{K}; \mathbf{M}_0; \mathbf{M}_1, \ldots, \mathbf{M}_m)$ asks us to find an m-tuple $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m) \in \mathbb{K}^m$ such that

$$\operatorname{rank}\left(\sum_{i=1}^{m} \alpha_i \mathbf{M}_i - \mathbf{M}_0\right) \leq r.$$

Remark. In practice, we will mostly be concerned with the case when $\mathbb{K}$ is a finite field, i.e., $\mathbb{K} = \mathbb{F}_q$.

Theorem 1 (Complexity of MinRank, [BFS99; Cou01])**.** *The MinRank problem is* NP*-complete.*

Proof sketch. There exists an effective algorithm to encode *any* system of multivariate polynomial equations as a MinRank instance. Another idea is to provide a reduction from the Syndrome Decoding problem to MinRank. □

Note that the proof sketch of theorem 1 implies that the MinRank problem is *very general*, i.e., various other computationally hard problems of practical importance can be reduced to solving MR instances.

2.1 Solving MinRank Instances

The two most practical and most widely documented methods of solving MR instances are:

- The so-called "kernel attack";
- Modeling MR instances as $\mathcal{MQ}$ systems.

2.1.1 The Kernel Attack

This was the first non-trivial attack against MinRank and was proposed by Goubin and Courtois [GC00]. The main idea is that instead of guessing $\boldsymbol{\alpha}$, we can guess the kernel of the linear combination of the MR matrices (as given in definition 1). Then, assuming that the kernel was guessed correctly, we can actually *solve* for $\boldsymbol{\alpha}$.

More formally, let $\mathsf{MR}(m, \eta, n, r, \mathbb{F}_q; \mathbf{M}_0; \mathbf{M}_1, \ldots, \mathbf{M}_m)$ be a given MR instance. For some parameter $\boldsymbol{\beta} \in \mathbb{F}_q^m$, denote by $H_{\boldsymbol{\beta}}$ the linear combination $\sum_{i=1}^{m} \beta_i \mathbf{M}_i - \mathbf{M}_0$. We would like to have $\operatorname{rank}(H_{\boldsymbol{\beta}}) \leq r$, i.e., $\boldsymbol{\beta}$ to be a solution. If that were the case, then by the rank-nullity theorem of linear algebra, we would have $\dim(\ker H_{\boldsymbol{\beta}}) \geq n - r$.

We can proceed by randomly choosing k vectors $\mathbf{x}^{(i)} \in \mathbb{F}_q^n$, $1 \leq i \leq k$. If these vectors are such that they fall into the kernel of the MR instance, then solving for $\boldsymbol{\beta}$ in

$$\begin{cases} H_{\boldsymbol{\beta}} \mathbf{x}^{(1)} = \mathbf{0} \\ H_{\boldsymbol{\beta}} \mathbf{x}^{(2)} = \mathbf{0} \\ \qquad \vdots \\ H_{\boldsymbol{\beta}} \mathbf{x}^{(k)} = \mathbf{0} \end{cases} \tag{2}$$

would allow us to retrieve the instance's solution. (2) is a system in $k\eta$ equations and m unknowns. Since we want (2) to never be underdetermined and have its numbers of equations and unknowns match as closely as possible, we can choose $k = \left\lceil \frac{m}{\eta} \right\rceil$.

It was established that there are at least q^{n-r} vectors in $\ker H_{\boldsymbol{\beta}}$ whenever $\boldsymbol{\beta}$ is a solution. Having that in mind, we can see that

$$\Pr\left\{ \left(x^{(1)}, \ldots, x^{(k)} \right) \in \ker H_{\boldsymbol{\beta}} \right\} \geq q^{-kr} = q^{-\left\lceil \frac{m}{\eta} \right\rceil r}. \tag{3}$$

What (3) tells us is that on average we have to guess & solve $q^{\left\lceil \frac{m}{\eta} \right\rceil r}$ times. Accounting for the complexity of the solving (2), we get an overall complexity of $\mathcal{O}\left(m \left(\left\lceil \frac{m}{\eta} \right\rceil \eta \right)^2 q^{\left\lceil \frac{m}{\eta} \right\rceil r} \right)$. The takeaway is that this attack provides a significant speedup over a brute-force search whenever $\left\lceil \frac{m}{\eta} \right\rceil r \ll m$.

2.1.2 $\mathcal{MQ}$-Solving Attacks

Modeling the MinRank problem as an $\mathcal{MQ}$ system was first proposed by Kipnis and Shamir [KS99]. The key idea here is that instead of guessing, we try to explicitly construct the kernel of the linear combination of the MR matrices.

Again, let $\mathsf{MR}(m, \eta, n, r, \mathbb{F}_q; \mathbf{M}_0; \mathbf{M}_1, \ldots, \mathbf{M}_m)$ be a given MR instance and $H_{\boldsymbol{\beta}}$ be defined as in section 2.1.1. We already saw that whenever $\boldsymbol{\beta}$ is a solution, there are at least $n - r$ linearly independent vectors in $\ker H_{\boldsymbol{\beta}}$. Proceed by trying to fix these vectors as follows:

$$\begin{aligned} \mathbf{x}^{(1)} &= \begin{bmatrix} 1 & 0 & \cdots & 0 & x_1^{(1)} & \cdots & x_r^{(1)} \end{bmatrix}^T \\ &\vdots \\ \mathbf{x}^{(n-r)} &= \begin{bmatrix} 0 & 0 & \cdots & 1 & x_1^{(n-r)} & \cdots & x_r^{(n-r)} \end{bmatrix}^T. \end{aligned} \tag{4}$$

Remark. Equation (4) is true up to a *change of basis*, so it might happen that it does not help us arrive at a solution.

Bearing this in mind, we can see that the $\mathcal{MQ}$ system

$$\left(\sum_{i=1}^{m} \beta_i \mathbf{M}_i - \mathbf{M}_0 \right)_{\eta \times n} \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(n-r)} \\ \vdots & \vdots & \ddots & \vdots \\ x_r^{(1)} & x_r^{(2)} & \cdots & x_r^{(n-r)} \end{bmatrix}_{n \times (n-r)} = \mathbf{0}_{\eta \times (n-r)} \tag{5}$$

over $\mathbb{F}_q\left[\beta_1, \ldots, \beta_m, x_1^{(1)}, \ldots, x_r^{(n-r)}\right]$, consisting of $\eta(n-r)$ equations in $r(n-r)+m$ variables, accurately models the solution to the MR instance. The complexity of solving (5) varies greatly depending on the method used – nevertheless, it is exponential in the worst case. The authors of [KS99] proposed using *relinearization*. In appendices A.3 and A.4 two methods will be discussed: one based on Gröbner bases and affine varieties, and the other based on *eXtended Linearization* (XL).

2.2 Zero-Knowledge Authentication Based on MinRank

The use MR instances as an underlying NP-complete problem in constructing an efficient zero-knowledge authentication protocol was proposed by Courtois [Cou01].

2.2.1 Preliminaries

Definition 2 (Zero-knowledge proof)**.** A *zero-knowledge proof* is a method by which one party (the *prover* P) can prove to another party (the *verifier* V) that a statement is true, without conveying *any* information apart from the fact that the statement is indeed true.

A zero-knowledge proof must satisfy three properties:

1. **Completeness.** A legitimate prover always gets accepted.
2. **Soundness.** An illegitimate prover is rejected with some fixed probability.
3. **Zero-knowledge.** No verifier can extract any information from a prover, even in multiple interactions.

2.2.2 The Authentication Scheme

The key idea behind the authentication scheme is stated in lemma 1.

Lemma 1. *Let* $\mathbf{M}$ *be an* $\eta \times n$ *matrix of rank* $r \leq \min(\eta, n)$*. Let* $\mathbf{S}$ *and* $\mathbf{T}$ *be two uniformly distributed random nonsingular matrices of orders* η *and* n *resp. Then* $\mathbf{SMT}$ *is uniformly distributed among all* $\eta \times n$ *matrices of rank* r*.*

Proof sketch. $\mathbf{S}$ and $\mathbf{T}$ correspond to compositions of elementary row and column operations resp., which preserve the rank of the matrix. $\mathbf{M}$ and $\mathbf{SMT}$ are said to be *matrix equivalent.* □

Lemma 1 tells us that given a MR instance, we can *mask* its solution by multiplying it from the left and the right with nonsingular matrices. All we need now is a way to force the prover to "play by the rules." We define the **public key** to be a hard[1] MR instance $\mathsf{MR}(m, \eta, n, r, \mathbb{F}_q; \mathbf{M}_0; \mathbf{M}_1, \ldots, \mathbf{M}_m)$, and the **private key** to be the solution $\boldsymbol{\alpha}$ to the MR instance as per definition 1, with $\mathbf{M} = \sum_{i=1}^{m} \alpha_i \mathbf{M}_i - \mathbf{M}_0$. The prover is going to convince the verifier of her knowledge of $\mathbf{M}$ (and $\boldsymbol{\alpha}$). The authentication is performed for multiple rounds: a prover P succeeds in verifying herself if and only if she successfully answers all queries posed by the verifier V.

In order to get the prover P to commit to playing by the rules, we use a collision-resistant hash function H as a random oracle to which P makes commitments. Figure 1 shows one round of MinRank authentication.

Theorem 2 ([Cou01])**.** *The MinRank authentication scheme is a* zero-knowledge proof.

Proof.

1. **Completeness.** It is clear that a legitimate prover, i.e., one with a correct $\boldsymbol{\alpha}$ always gets verified.
2. **Soundness.** An illegitimate prover being able to get verified implies that she either solved the MR instance or found a collision in H. Both of these are computationally unfeasible – MinRank is NP-complete (theorem 1) and H is assumed to be collision-resistant. Since the queries $\mathcal{Q} \in \{1, 2\}$ can be answered by any non-cheating prover, we see that the probability of an illegitimate prover not being rejected is $\left(\frac{2}{3} + \mathsf{negl}\right)^{\#\text{rounds}}$.
3. **Zero-knowledge.** This is a direct corollary of lemma 1.

□

[1] Meaning both the matrices $\mathbf{M}_0; \mathbf{M}_1, \ldots, \mathbf{M}_m$ and the solution $\boldsymbol{\alpha}$ are uniformly distributed. See appendix A.1 for the exact algorithm.

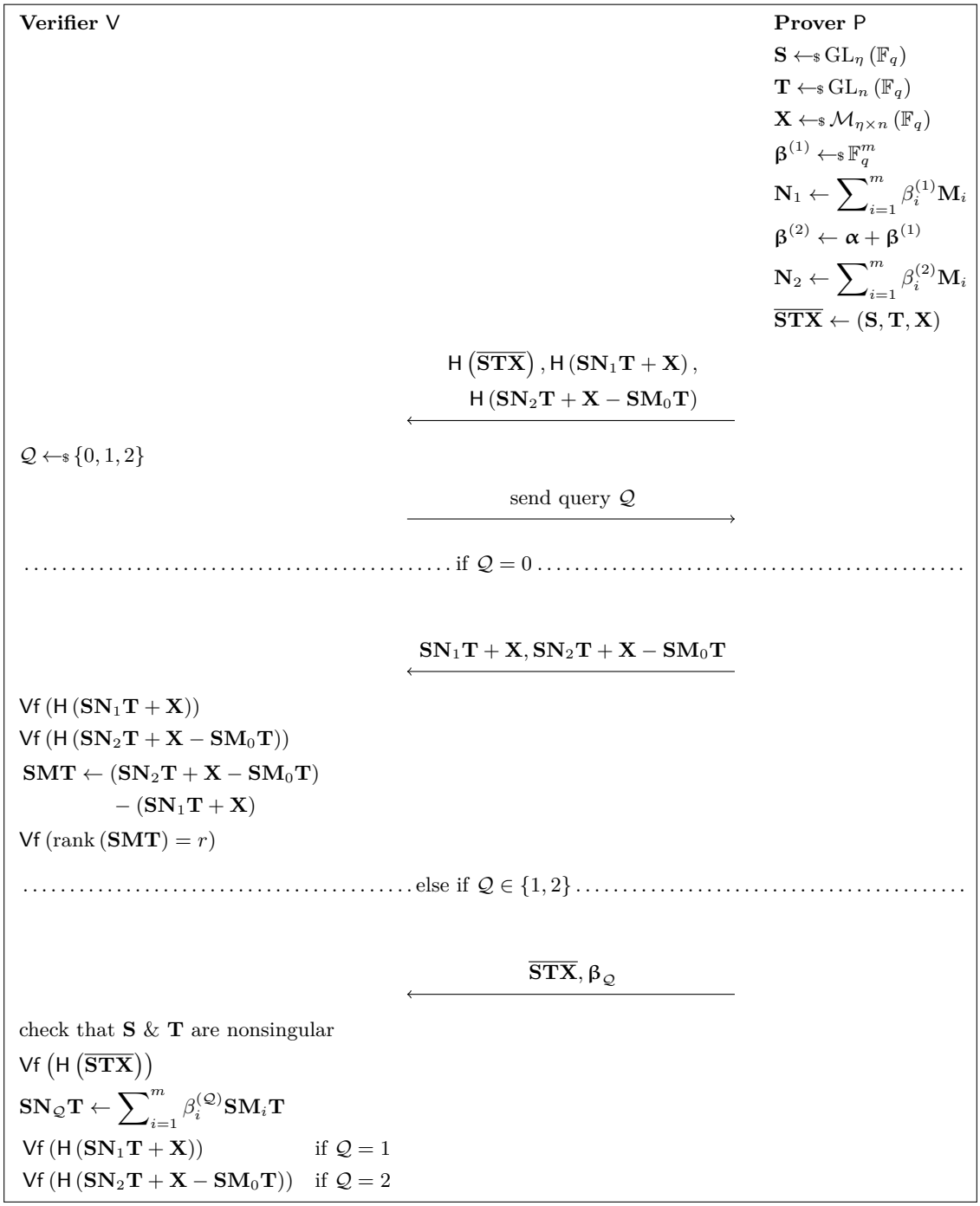

Figure 1: One Round of Affine MinRank Authentication

3 Conclusion

The MinRank problem is of crucial importance to many cryptographic schemes and ideas. We have seen that it embodies many computationally hard problems, including those used in various $\mathcal{MQ}$ schemes and even cryptosystems based on decoding linear codes such as the McEliece cryptosystem.

Schemes such as these and the one discussed in section 2.2.2 are of interest to the world of "post-quantum cryptography." The publication of Shor's algorithm [Sho94] means that if a quantum computer were to be constructed, "traditional cryptography"—cryptography based on integer factorization and discrete logarithms—would become immediately insecure.

Furthermore, a few attacks on the MinRank problem were reviewed. These attacks appear to be well-suited for attacking the MinRank authentication scheme.

References

[BFS99] Jonathan F. Buss, Gudmund S. Frandsen, and Jeffrey O. Shallit. "The Computational Complexity of Some Problems of Linear Algebra". In: *Journal of Computer and System Sciences* 58.3 (June 1999), pp. 572–596. ISSN: 0022-0000. DOI: 10.1006/jcss.1998.1608 (cit. on p. 1).

[CLO15] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms*. 4th ed. Undergraduate Texts in Mathematics. Springer, 2015. ISBN: 978-3-319-16721-3. DOI: 10.1007/978-3-319-16721-3 (cit. on p. 6).

[Cou+00] Nicolas Courtois et al. "Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations". In: *Advances in Cryptology — EUROCRYPT '00*. Ed. by Bart Preneel. Vol. 1807. Lecture Notes in Computer Science. Springer, 2000, pp. 392–407. ISBN: 978-3-540-67517-4. DOI: 10.1007/3-540-45539-6_27 (cit. on p. 6).

[Cou01] Nicolas T. Courtois. "Efficient Zero-Knowledge Authentication Based on a Linear Algebra Problem MinRank". In: *Advances in Cryptology — ASIACRYPT '01*. Vol. 2248. Lecture Notes in Computer Science. Springer, 2001, pp. 402–421. DOI: 10.1007/3-540-45682-1_24 (cit. on pp. 1, 3, 6, 7).

[FLP08] Jean-Charles Faugère, Françoise Levy-dit-Vehel, and Ludovic Perret. "Cryptanalysis of MinRank". In: *Advances in Cryptology – CRYPTO '08*. Ed. by David Wagner. Vol. 5157. Lecture Notes in Computer Science. Springer, 2008, pp. 280–296. ISBN: 978-3-540-85173-8. DOI: 10.1007/978-3-540-85174-5_16 (cit. on p. 6).

[GC00] Louis Goubin and Nicolas T. Courtois. "Cryptanalysis of the TTM Cryptosystem". In: *Advances in Cryptology — ASIACRYPT '00*. Ed. by Tatsuaki Okamoto. Vol. 1976. Lecture Notes in Computer Science. Springer, 2000, pp. 44–57. ISBN: 978-3-540-41404-9. DOI: 10.1007/3-540-44448-3_4 (cit. on pp. 1, 2).

[KS99] Aviad Kipnis and Avi Shamir. "Cryptanalysis of the HFE Public Key System by Relinearization". In: *Advances in Cryptology — CRYPTO '99*. Ed. by Michael Wiener. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 19–30. ISBN: 978-3-540-66347-8. DOI: 10.1007/3-540-48405-1_2 (cit. on pp. 1–3, 6).

[Moh99] Tzuong-Tsieng Moh. "A Public Key System With Signature and Master Key Function". In: *Communications in Algebra* 27 (5 1999), pp. 2207–2222. ISSN: 1532-4125. DOI: 10.1080/00927879908826559 (cit. on p. 1).

[Pat96] Jacques Patarin. "Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms". In: *Advances in Cryptology — EUROCRYPT '96*. Ed. by Ueli Maurer. Vol. 1070. Lecture Notes in Computer Science. Springer, 1996, pp. 33–48. ISBN: 978-3-540-61186-8. DOI: 10.1007/3-540-68339-9_4 (cit. on p. 1).

[Sag15] SageMath Developers, The. *Sage Mathematics Software (Version 6.10)*. Dec. 18, 2015. URL: http://www.sagemath.org/ (cit. on p. 5).

[Sho94] Peter W. Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring". In: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. SFCS '94. Washington, DC, USA: IEEE Computer Society, 1994, pp. 124–134. ISBN: 0-8186-6580-7. DOI: 10.1109/SFCS.1994.365700 (cit. on p. 4).

A Technical Documentation

The implementation was written in SageMath [Sag15]. It includes three files:

- `mr_auth.py`: this file provides MinRank instance generation and a toy implementation of the zero-knowledge authentication scheme;
- `mr_attk.py`: this file implements the kernel and $\mathcal{MQ}$-solving attacks;
- `mr_util.py`: this file provides some utility functions;
- `mr_test.py`: this file runs some tests to showcase the implementation.

All code was tested on an Intel i5-4670 CPU @ 3.40 GHz running an ArchLinux operating system.

A.1 Instance Generation

Instances are generated according to algorithm 1. Internally they are represented as `MinRankInstance` objects. It is easy to see that this algorithm generates a correct instance. It is also relatively fast: generating 10 000 instances with $m = 10, \eta = n = 6, r = 3, q = 65521$ required 10.252 s.

Algorithm 1 MinRank Instance Generation [Cou01]

Input: $m, \eta, n, r, q \in \mathbb{N}_+$ $(r \leq \min(\eta, n))$
Output: An instance $\mathsf{MR}(m, \eta, n, r, \mathbb{F}_q; \mathbf{M}_0; \mathbf{M}_1, \ldots, \mathbf{M}_m)$ with solution $\boldsymbol{\alpha}$ and $\mathbf{M} = \sum_{i=1}^{m} \alpha_i \mathbf{M}_i - \mathbf{M}_0$

1: $\mathbf{M}_0; \mathbf{M}_1, \ldots, \mathbf{M}_{m-1} \leftarrow_{\$} \mathcal{M}_{\eta \times n}(\mathbb{F}_q)$
2: $\mathbf{M} \leftarrow_{\$} \mathcal{M}_{\eta \times n}^{\mathrm{rank}=r}(\mathbb{F}_q)$
3: $\boldsymbol{\alpha} \leftarrow_{\$} \mathbb{F}_q^m$ s.t. $\alpha_m \neq 0$
4: $\mathbf{M}_m \leftarrow \left(\mathbf{M} + \mathbf{M}_0 - \sum_{i=1}^{m-1} \alpha_i \mathbf{M}_i\right) / \alpha_m$ ▷ This ensures that $\mathbf{M}_m$ is "correct" w.r.t. $\boldsymbol{\alpha}$ and $\mathbf{M}$
5: **return** $\mathsf{MR}(m, \eta, n, r, \mathbb{F}_q; \mathbf{M}_0; \mathbf{M}_1, \ldots, \mathbf{M}_m)$, $\boldsymbol{\alpha}$, $\mathbf{M}$

A.2 Running the Kernel Attack

The implementation is very straightforward and basically follows section 2.1.1. In order to run the attack, invoke the `run_kernel` method of a `MinRankAttack` object. There is an optional boolean parameter `constrained_run` which indicates whether the checking should stop after $q^{\lceil \frac{m}{\eta} \rceil r}$ attempts; otherwise it is done until a valid solution is found.

A.3 Solving the $\mathcal{MQ}$ System Using Gröbner Bases

The authors of [FLP08] proposed a one-to-one correspondence between the affine variety

$$V(\mathcal{I}_{\mathrm{KS}}) = \{\mathbf{x} \in \mathbb{F}_q^{r(n-r)+m} : f(\mathbf{x}) = 0 \; \forall f \in \mathcal{I}_{\mathrm{KS}}\}$$

and the solution of (5), where $\mathcal{I}_{\mathrm{KS}}$ is the ideal generated by the equations of (5). This allows us to use the functionality of SageMath/Singular to compute a solution.

The attack can be performed by invoking the `run_groebner` method of a `MinRankAttack` object. Internally, it uses the Singular procedure `stdfglm`, which computes a Gröbner basis w.r.t. a degree-reverse lexicography ordering, and then converts it to lexicographic ordering using the FGLM algorithm. Lexicographic ordering is needed in order to ensure a triangular form of the system. One can think of this method as a generalization of the Gaussian elimination to $\mathcal{MQ}$ systems [CLO15, ch. 2, 3].

For tiny instances $(m, \eta, n \approx 7)$, I found the kernel attack to be more effective, possibly because of its simplicity. However, [FLP08] states that the Gröbner basis attack is feasible even for parameter sets proposed in the MinRank authentication scheme. In particular, the theoretical bound on the complexity of computing a Gröbner basis of a polynomial system with m equations in n variables is given by $\mathcal{O}\left(m\binom{n+d_{\mathrm{reg}}}{d_{\mathrm{reg}}}^{\omega}\right)$ where d_{reg} is the degree of regularity (i.e., the highest degree reached during the computation), and $2 \leq \omega \leq 3$ is the exponent in the complexity of matrix multiplication. Despite this, the observed complexity was much smaller and can in fact be bounded by a *polynomial* due to the structure of (5) (it is formed by bilinear equations).

A problem which may arise is $\mathcal{I}_{\mathrm{KS}}$ ending up *not being radical*. To eliminate that possibility, we append the field equations to (5), which ensures that $\sqrt{\mathcal{I}_{\mathrm{KS}}} = \mathcal{I}_{\mathrm{KS}}$ while not changing the solution set.

A.4 Solving the $\mathcal{MQ}$ System Using XL

Relinearization, as it was proposed by [KS99] was found to be inefficient as it introduced way too many additional variables. To alleviate that issue, the authors of [Cou+00] proposed a simplified and improved version of relinearization called *eXtended Linearization* (XL).

The idea is to raise the degree of the system sufficiently enough to be able to apply Gaussian elimination as if it were a linear system. The hope is that this step yields a univariate polynomial which can be solved and substituted back in.

The XL algorithm takes a degree parameter $D \geq 2$, which indicates the maximal degree to which the system will be raised. It can be run using the `run_xl` method of `MinRankAttack`. Unfortunately, I found the XL algorithm to cope poorly with $\mathcal{MQ}$ systems arising from MinRank instances.

A.5 Implementation of the Authentication Scheme

For the purposes of the MinRank authentication scheme, two main objects are provided:

- A prover, represented by the `Prover` class;
- A verifier, represented by the `Verifier` class.

Both these objects are tied to `MinRankInstance` objects. The `Prover` and `Verifier` classes mimic their corresponding roles as shown in fig. 1. A typical round of execution is shown in listing 1. In order to provide a wrapper around the procedure shown in listing 1, the function `authentication_driver` may be used, taking a prover, verifier, and (optionally) the number of rounds as parameters.

```
# Assume P is the prover and V the verifier
P.prepare_round()  # Compute everything the prover needs for a round
P.send_hashes(V)   # Send the hashes, i.e., make a commitment
V.send_query(P)    # Send the query to the prover who will answer it

# Alternatively, execute this function
authentication_driver(P, V, 35)  # Perform 35 rounds
```

Listing 1: Executing One Round of MinRank Authentication

As a cryptographic hash function a built-in implementation of SHA-256 is used[2]. Due to the toy's implementation "closed world" nature, there is *no* concept of public and private keys. Instead, a `Prover` subclass `LegitimateProver` represents a prover who can be given "access to" a MinRank instance. This is achieved by invoking the `give_access` method of `MinRankInstance`.

Interestingly, even in a real world implementation, the key sizes are quite small. As part of algorithm 1, $\mathbf{M}_0$; $\mathbf{M}_1, \ldots, \mathbf{M}_{m-1}$; $\mathbf{M}$ need not be transmitted as they can be generated from a common 160 bit seed [Cou01]. This leaves only $\mathbf{M}_m$ to be transmitted, giving a $160 + \eta n \log_2 q$ bit public key. The private key is $m \log_2 q$ bits long (needed for $\boldsymbol{\alpha}$).

Table 1 shows some time measurements of authentications with different parameter sets. We see that parameter set D, which includes 81 19×19 matrices over $\mathbb{F}_2$ is a lot slower than parameter sets A and C, both of which include 10 matrices (of dimension 6×6 and 11×11 resp.) over $\mathbb{F}_{65521}$. As expected, we also notice that a successful authentication takes more time than an unsuccessful one.

Table 1: Time Needed for Authentications (100 Authentications, 35 Rounds Each)

Parameter set [Cou01]	Time (legitimate P, in s)	Time (illegitimate P, in s)
A	18.349	1.763
C	133.610	11.450
D	1050.127	91.196

[2] Objects are hashed by hashing their internal representation. This is likely far from the optimal way, but it does serve its purpose; at least in a toy implementation such as this one.