The MinRank Problem

Survey, Implementation, and One Application

Dario Gjorgjevski[1]
gjorgjevski.dario@students.finki.ukim.mk

[1]Faculty of Computer Science and Engineering
Ss. Cyril and Methodius University in Skopje

January 12, 2016

Outline

Definition of the MinRank Problem

The MinRank problem (MR) is a fundamental problem in linear algebra of finding a low-rank linear combination of matrices.

Definition (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

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

In practice, we have $\mathbb{K} = \mathbb{F}_q$.

Complexity of the MinRank Problem

Theorem ([BFS99; Cou01])

The MinRank problem is NP*-complete.*

- MinRank's NP-completeness is what allows us to use it as an underlying problem in a zero-knowledge authentication scheme.
- We will also see a connection between MinRank and multivariate quadratic ($\mathcal{MQ}$) cryptosystems. Interestingly, any system of multivariate polynomial equations can be effectively encoded as a MR instance.

Outline

Key Idea Behind the Kernel Attack

- Proposed by Goubin and Courtois [GC00].
- Rather than guess a solution, guess its kernel. If the kernel is guessed correctly, the solution can be solved for.
- Let $H_{\boldsymbol{\beta}} = \sum_{i=1}^{m} \beta_i \mathbf{M}_i - \mathbf{M}_0$ ($\boldsymbol{\beta}$ is a parameter).
- If $\boldsymbol{\alpha}$ is a solution, $(\operatorname{rank} H_{\boldsymbol{\alpha}} \le r \iff \dim(\ker H_{\boldsymbol{\alpha}}) \ge n - r)$ $\implies$ the kernel's dimension can be relatively large making guessing more feasible.
- Given a correct guess, the solution $\boldsymbol{\alpha}$ can be retrieved in roughly cubic time by simply solving a linear system of equations.

The Kernel Attack Algorithm

Algorithm 1 The Kernel Attack on MinRank

Input: $\mathsf{MR}\,(m, \eta, n, r, \mathbb{F}_q; \mathbf{M}_0; \mathbf{M}_1, \ldots, \mathbf{M}_m)$
Output: A solution to the MR instance (if any)

repeat

$\mathbf{x}^{(i)} \leftarrow_{\$} \mathbb{F}_q^n,\ 1 \leq i \leq \left\lceil \frac{m}{\eta} \right\rceil$

$\boldsymbol{\beta} \leftarrow \text{solve}\ \left\{ \left(\sum_{j=1}^{m} \beta_j \mathbf{M}_j - \mathbf{M}_0 \right) \mathbf{x}^{(i)} = \mathbf{0} \right\},\ 1 \leq i \leq \left\lceil \frac{m}{\eta} \right\rceil$

until ($\boldsymbol{\beta}$ solves the MR instance) $\vee$ (the algorithm has been run sufficiently many times)

Guess & solve $q^{\left\lceil \frac{m}{\eta} \right\rceil r}$ times $\Longrightarrow \mathcal{O}\left(m \left(\left\lceil \frac{m}{\eta} \right\rceil \eta \right)^2 q^{\left\lceil \frac{m}{\eta} \right\rceil r} \right)$.

Key Idea Behind the $\mathcal{MQ}$ Modeling

- Proposed by Kipnis and Shamir [KS99].
- Instead of guessing the kernel, we can attempt to explicitly construct it.
- If $\boldsymbol{\alpha}$ is a solution, $\operatorname{rank} H_{\boldsymbol{\alpha}} \leq r \iff \dim(\ker H_{\boldsymbol{\alpha}}) \geq n - r$ $\iff \exists\, n - r$ linearly independent vectors in $\ker H_{\boldsymbol{\alpha}}$.
- Write these vectors systematically as $\mathbf{x}^{(i)} = \begin{bmatrix} \mathbf{e}_i & x_1^{(i)} & x_2^{(i)} & \cdots & x_r^{(i)} \end{bmatrix}^T$, $1 \leq i \leq n - r$, where $\mathbf{e}_i \in \mathbb{F}_q^{n-r}$ and the $x_j^{(i)}$'s are newly-introduced variables.

The $\mathcal{MQ}$ System

Therefore, we can model a MR instance as an $\mathcal{MQ}$ system:

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

(1) is a quadratic system of $\eta(n-r)$ equations in $r(n-r)+m$ variables.

Solving the $\mathcal{MQ}$ System

- The best method we have for solving multivariate polynomial systems of equations are lex Gröbner bases.
- Gröbner bases are defined w.r.t. monomial orderings. A lex Gröbner basis can be thought of as a generalization of Gaussian elimination.
- The theoretical complexity of computing a Gröbner basis for a system with m equations in n variables is $\mathcal{O}\left(m\binom{n+d_{\text{reg}}}{d_{\text{reg}}}^{\omega}\right)$, where d_{reg} is the maximum degree reached during the computation and $2 \leq \omega \leq 3$ is the exponent in the complexity of matrix multiplication.
- The system given in (1) exhibits certain structural properties (it is formed by bilinear equations), so the complexity observed in practice is much lower.

Implementation of the Attacks

- The implementations are done in SageMath and follow the theoretical foundations in a straightforward manner.
- The kernel attack is a simple implementation of algorithm 1.
- Gröbner basis computation is done using the SINGULAR procedure `stdfglm`. Internally, it uses the F_4 algorithm to compute a Gröbner basis w.r.t. a degrevlex ordering, and then converts it to a lex ordering using the FGLM algorithm. Once the Gröbner basis is computed, solving (1) is trivial and handled by SageMath's `variety()` method, which computes the affine variety of an ideal.

Outline

Key Idea Behind the Protocol

The protocol was proposed by Courtois [Cou01]. The key idea is stated in the following lemma.

Lemma

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.

The takeaway is that a MinRank solution can be effectively *masked* by two isomorphisms. In order to force a prover to "play by the rules," a collision-resistant hash function H is used to make commitments.

The Prover Setup

1. A uniformly chosen random combination $\boldsymbol{\beta}^{(1)}$ of the $\mathbf{M}_i$'s. $\mathbf{N}_1 = \sum_{i=1}^{m} \beta_i^{(1)} \mathbf{M}_i$.
2. Let $\boldsymbol{\beta}^{(2)} = \boldsymbol{\alpha} + \boldsymbol{\beta}^{(1)}$, where $\boldsymbol{\alpha}$ is the MinRank solution a legitimate prover should have access to. $\mathbf{N}_2 = \sum_{i=1}^{m} \beta_i^{(2)} \mathbf{M}_i$.
3. Random nonsingular matrices $\mathbf{S}$ and $\mathbf{T}$, and a completely random matrix $\mathbf{X}$.
4. The prover commits the hash values of the $(\mathbf{S}, \mathbf{T}, \mathbf{X})$ triple, and of $\mathbf{S}\mathbf{N}_1\mathbf{T} + \mathbf{X}$ and $\mathbf{S}\mathbf{N}_2\mathbf{T} + \mathbf{X} - \mathbf{S}\mathbf{M}_0\mathbf{T}$.

The Verifier

The verifier sends a random query ($\mathcal{Q} \leftarrow_{\$} \{0, 1, 2\}$) and either:

- Checks the committed hashes of the $(\mathbf{S}, \mathbf{T}, \mathbf{X})$ triple and one of the $\mathbf{N}_i$'s; or
- Checks the committed hashes of $\mathbf{N}_1$, $\mathbf{N}_2$, *and the rank of* $\mathbf{S}\mathbf{N}_2\mathbf{T} + \mathbf{X} - \mathbf{S}\mathbf{M}_0\mathbf{T} - \mathbf{S}\mathbf{N}_1\mathbf{T} + \mathbf{X} = \mathbf{S}\left(\sum_{i=1}^{m} \alpha_i \mathbf{M}_i - \mathbf{M_0}\right)\mathbf{T}$. This step is the backbone of the authentication, as by the previous lemma it remains a solution to the MinRank instance.

The protocol is *black box zero-knowledge* with a cheating probability of $\frac{2}{3}$. A prover authenticating herself means either solving the NP-complete problem of MinRank, or finding a collision in the hash function H and playing "dishonestly." Authentication is carried out in multiple rounds and is successful if and only if each round is successful.

Implementation of the Protocol

- The implementation follows the description of the protocol. It is built around two objects, `Prover` and `Verifier` who are each associated to `MinRankInstance` objects.
- Legitimate provers are represented as `LegitimateProver` objects and can be given access to `MinRankInstance` objects.
- Instance generation is done according to the algorithm outlined in [Cou01], i.e. instances are generated such that both the $\mathbf{M}_i$'s and the solution $\boldsymbol{\alpha}$ are uniformly distributed.
- There is no strict concept of public/private keys in the toy implementation, but in practice the keys are quite short as most of their parts can be generated by a pseudo-random generator from a shared seed.

Performance

- Instance generation is relatively fast: generating 10 000 instances $m = 10, \eta = n = 6, r = 3, q = 65521$ required 10.252 s.
- Authentication performance depends largely on the parameter set (parameter sets A and C include few matrices over $\mathbb{F}_{65521}$, while D includes many matrices over $\mathbb{F}_2$).

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

References

- 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* 3 (June 1999).
- Nicolas T. Courtois. "Efficient Zero-Knowledge Authentication Based on a Linear Algebra Problem MinRank". In: *Advances in Cryptology — ASIACRYPT '01*. Lecture Notes in Computer Science. Springer, 2001.
- Louis Goubin and Nicolas T. Courtois. "Cryptanalysis of the TTM Cryptosystem". In: *Advances in Cryptology — ASIACRYPT '00*. Ed. by Tatsuaki Okamoto. Lecture Notes in Computer Science. Springer, 2000. ISBN: 978-3-540-41404-9.
- Aviad Kipnis and Avi Shamir. "Cryptanalysis of the HFE Public Key System by Relinearization". In: *Advances in Cryptology — CRYPTO '99*. Ed. by Michael Wiener. Lecture Notes in Computer Science. Springer, 1999. ISBN: 978-3-540-66347-8.