Error-Correcting Codes in the Rank Metric

With Applications to Cryptography

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

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

January 30, 2018

[1]Mentored by prof. Simona Samardjiska

Contents

Past vs. Present

- In the past, cryptography was used to provide secure communication in an *ad-hoc manner*.
- This picture changed radically with the development of a rich and rigorous theory in the late 20th century.
- Today, we have algorithms for secure communication in two settings: *symmetric* and *asymmetric*.
- We will quickly review both; however, the remainder of this thesis focuses entirely on algorithms for secure communication in the asymmetric setting.

The Symmetric Setting

- The message m is *encrypted* under the secret key k to obtain the *ciphertext* c.
- The ciphertext c is *decrypted* under the same key k.
- If the two parties could share the secret key securely, why could they not do the same with the message?

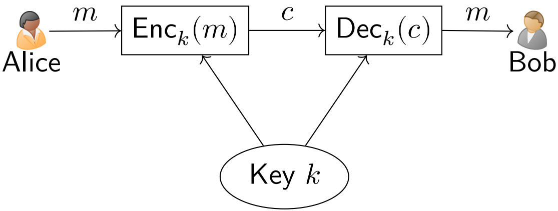

Figure: Secure communication in the symmetric setting.

The Asymmetric Setting

Cryptography in the asymmetric setting is called *public-key cryptography*. The RSA (Rivest et al.) cryptosystem is the most widely used and one of the first asymmetric cryptosystems:

1. Encryption is done with the intended recipient's *public key*, k_{pub}, known to everybody.
2. Decryption is done with the *private key*, k_{priv}, known only to the recipient.

Real-world analogy

Think of a mailbox: encryption corresponds to leaving mail in a person's mailbox, decryption to the person opening the mailbox to retrieve it.

RSA Basics

RSA relies on the computational complexity of *integer factorization*.

Example (Integer factorization)

If you were asked to multiply

$$17\,627\,949\,842\,247\,424\,607 \times 15\,969\,639\,761\,398\,924\,673,$$

you would need several minutes with pen and paper to arrive at 281 512 008 712 700 373 730 275 954 373 439 628 511. But, what if you were asked to find two integers, p and q, such that

$$p \times q = 281\,512\,008\,712\,700\,373\,730\,275\,954\,373\,439\,628\,511?$$

Unfortunately, Shor published an algorithm that can factor integers efficiently and hence “break” RSA. The only drawback: it needs a quantum computer.

The McEliece Cryptosystem

- A cryptosystem from roughly the same time as RSA is believed to be hard even for quantum computers: the McEliece cryptosystem.
- It relies on the hardness of decoding random codes over $\mathbb{F}_q$ in the Hamming metric.
- The problem was proven $\mathcal{NP}$-complete when $q = 2$ by Berlekamp et al.; and in the general case by Barg.
- The best attacks against McEliece come from a family of algorithms called *information set decoding* (ISD).

Issues with McEliece

...and Fixing Them

- Unfortunately, McEliece requires a key size of roughly 192 kB to achieve 128-bit security against information set decoding $\implies$ very prohibitive for embedded devices.
- Gabidulin et al. proposed the GPT cryptosystem which uses error-correcting codes in the *rank metric*, as opposed to McEliece's Goppa codes which correct errors in the Hamming metric.
- GPT is believed to require much smaller keys in order to achieve the same security.

Cryptography and the Information Transmission Model

- The information transmission model comes from the landmark work of Shannon.
- It can be utilized for cryptographic purposes by means of error-correcting codes. McEliece is one of the most prominent examples of such *code-based cryptography*.
- We assume transmission to be error-free and add noise deliberately for encryption.

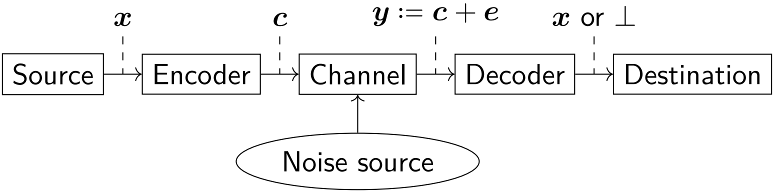

Figure: Transmitting information over a noisy channel.

Contents

Linear Codes

Definition (Linear Code)

An $[n, k]$-code $\mathcal{C}$ over a finite field $\mathbb{F}$ is a k-dimensional subspace of the vector space $\mathbb{F}^n$. The elements of $\mathcal{C}$ are called *codewords*.
$\mathcal{C}$ is an $[n, k, d]$-code if $d = \min\{\|\boldsymbol{x} - \boldsymbol{y}\| : \boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}, \boldsymbol{x} \neq \boldsymbol{y}\}$ for some norm $\|\cdot\|$. d is called the *minimum distance* of the code with respect to $\|\cdot\|$.

An $[n, k, d]$-code can correct an error $\boldsymbol{e}$ if and only if

$$\|\boldsymbol{e}\| \leq t =: \lfloor (d-1)/2 \rfloor.$$

t is called the *error-correcting capacity* of $\mathcal{C}$.

Generating and Utilizing Linear Codes

Definition (Generator Matrix)

A full-rank matrix $\boldsymbol{G} \in \mathbb{F}^{k \times n}$ is said to be a *generator matrix* for the $[n, k]$-code $\mathcal{C}$ if its rows span $\mathcal{C}$ over $\mathbb{F}$. In other words, if

$$\mathcal{C} = \{\boldsymbol{x}\boldsymbol{G} : \boldsymbol{x} \in \mathbb{F}^k\}.$$

$\boldsymbol{G}$ defines an *encoding map* $f_{\boldsymbol{G}} : \mathbb{F}^k \to \mathbb{F}^n$ given by $\boldsymbol{x} \mapsto \boldsymbol{x}\boldsymbol{G}$.

1. A message $\boldsymbol{x} \in \mathbb{F}^k$ is encoded as $\boldsymbol{c} := \boldsymbol{x}\boldsymbol{G} \in \mathbb{F}^n$.
2. The codeword $\boldsymbol{c}$ is inflicted by additive noise $\boldsymbol{e}$ with $\|e\| \leq t$, and received as $\boldsymbol{y} := \boldsymbol{c} + \boldsymbol{e}$.
3. $\boldsymbol{y}$ is decoded into $\boldsymbol{x}$ using an efficient decoding procedure for $\mathcal{C}$. (Decoding can of course be done without an efficient procedure, but will be computationally very demanding.)

The Rank Metric

Definition (Rank Norm)

Let $\boldsymbol{x} := \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \in \mathbb{F}_{q^m}^n$ and $\{\beta_1, \ldots, \beta_m\}$ be a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. For all $i \in \{1, \ldots, n\}$, we can write $x_i = \sum_{j=1}^{m} x_{i,j}\beta_j$ with $x_{i,j} \in \mathbb{F}_q$. The *rank norm* $\|\cdot\|_q$ is defined as

$$\|\boldsymbol{x}\|_q := \operatorname{rank} \begin{bmatrix} x_{1,1} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,m} \end{bmatrix}. \tag{1}$$

The rank norm of is independent of the choice of basis and induces a metric called the *rank metric* (or *rank distance*):

$$\mathrm{d}_{\mathrm{R}}(\boldsymbol{x} - \boldsymbol{y}) := \|\boldsymbol{x} - \boldsymbol{y}\|_q.$$

Generating Gabidulin Codes

Gabidulin constructed a family of Maximum Rank Distance codes over $\mathbb{F}_{q^m}$ of length $n \leq m$. For any $x \in \mathbb{F}_{q^m}$ and any $i \in \mathbb{Z}$, $x^{[i]} := x^{q^i}$. (The operation is applied component-wise to vectors and matrices.)

Definition (Gabidulin Code)

Let $\boldsymbol{g} := \begin{bmatrix} g_1 & \cdots & g_n \end{bmatrix} \in \mathbb{F}_{q^m}^n$ with all g_i independent over $\mathbb{F}_q$. (This implies $n \leq m$.) The *Gabidulin code* of dimension k, $\mathfrak{G}_k(\boldsymbol{g})$, is generated by

$$\boldsymbol{G} := \begin{bmatrix} g_1^{[0]} & \cdots & g_n^{[0]} \\ \vdots & \ddots & \vdots \\ g_1^{[k-1]} & \cdots & g_n^{[k-1]} \end{bmatrix}. \tag{2}$$

Gabidulin codes have $d = n - k + 1$, hence errors of rank $t = \lfloor (n-k)/2 \rfloor$ can be corrected in time $\mathcal{O}(d^3 + dn)$.

The GPT Cryptosystem

System Parameters and Key Generation

System Parameters

- Choose $k, n, m \in \mathbb{N}$ such that $k < n \leq m$.
- Define the error-correcting capacity $t = \lfloor (n-k)/2 \rfloor$.
- Choose $l \in \mathbb{N}$ with $l \ll n$.

Key Generation

- Let $\boldsymbol{g} \in \mathbb{F}_{q^m}^n$ with $\|\boldsymbol{g}\|_q = n$ and let $\boldsymbol{G}$ be a generator of $\mathfrak{G}_k(\boldsymbol{g})$.
- Let $\boldsymbol{S} \in \mathsf{GL}_k(\mathbb{F}_{q^m})$ and $\boldsymbol{P} \in \mathsf{GL}_{n+l}(\mathbb{F}_q)$. Note that $\boldsymbol{P}$ is an *isometry* in the rank metric: $\|\boldsymbol{x}\boldsymbol{P}\|_q = \|\boldsymbol{x}\|_q$.
- Let $\boldsymbol{X}_1 \in \mathbb{F}_{q^m}^{k \times l}$ and $\boldsymbol{X}_2 \in \mathbb{F}_{q^m}^{k \times n}$ with $\operatorname{rank} \boldsymbol{X}_2 < t$.
- Define the *distortion transformation*

$$\mathcal{D}(\boldsymbol{G}) := \boldsymbol{S} \begin{bmatrix} \boldsymbol{X}_1 & \boldsymbol{G} + \boldsymbol{X}_2 \end{bmatrix} \boldsymbol{P}.$$

The GPT Cryptosystem

Public Key, Encryption, and Decryption

- The public key consists of $\boldsymbol{G}_{\text{pub}} := \mathscr{D}(\boldsymbol{G})$ and $t_{\text{pub}} := t - \operatorname{rank} \boldsymbol{X}_2$.
- To encrypt $\boldsymbol{x} \in \mathbb{F}_{q^m}^k$, choose $\boldsymbol{e} \in \mathbb{F}_{q^m}^n$ with $\|\boldsymbol{x}\|_q \leq t_{\text{pub}}$ and compute $\boldsymbol{y} := \boldsymbol{x}\boldsymbol{G}_{\text{pub}} + \boldsymbol{e}$.
- To decrypt $\boldsymbol{y} \in \mathbb{F}_{q^m}^n$, apply the decoding procedure for $\mathfrak{G}_k(\boldsymbol{g})$ to the last n components of $\boldsymbol{y}\boldsymbol{P}^{-1}$ to obtain $\boldsymbol{x}\boldsymbol{S}$, then multiply by $\boldsymbol{S}^{-1}$. The math works out just nicely.

Common Form of the GPT Public Key

We simplified and generalized the following theorem due to Kshevetskiy.

Theorem (GPT Public Key)

Let $\boldsymbol{G}_{\mathsf{pub}}$ *be a public GPT generator, and define* $\operatorname{rank}\boldsymbol{X}_2 =: t_{\boldsymbol{X}_2}$*. Then, there exist* $\boldsymbol{P}^* \in \mathsf{GL}_{l+n}(\mathbb{F}_q)$, $\boldsymbol{X}^* \in \mathbb{F}_{q^m}^{k\times(l+t_{\boldsymbol{X}_2})}$*, and* $\boldsymbol{G}^*$ *which generates an* $[n-t_{\boldsymbol{X}_2}, k]$*-Gabidulin code* $\mathfrak{G}_k(\boldsymbol{g}^*)$ *such that*

$$\boldsymbol{G}_{\mathsf{pub}} = \boldsymbol{S}\begin{bmatrix}\boldsymbol{X}^* & \boldsymbol{G}^*\end{bmatrix}\boldsymbol{P}^*.$$

Furthermore, $\mathfrak{G}_k(\boldsymbol{g}^*)$ *can correct more than* t_{pub} *errors.*

Distinguishing Properties

Definition

For any $i \in \mathbb{N}$, let $\Lambda_i : \mathbb{F}_{q^m}^{k \times n} \to \mathbb{F}_{q^m}^{ik \times n}$ be the $\mathbb{F}_q$-linear operator defined as:

$$\Lambda_i(\boldsymbol{X}) := \begin{bmatrix} \boldsymbol{X}^{[0]} \\ \vdots \\ \boldsymbol{X}^{[i-1]} \end{bmatrix}.$$

For any code $\mathcal{C}$ generated by $\boldsymbol{G}$, denote by $\Lambda_i(\mathcal{C})$ the code generated by $\Lambda_i(\boldsymbol{G})$. It turns out that Λ_i can distinguish Gabidulin codes from random.

Gabidulin vs. Random

Lemma

Let $\boldsymbol{g} \in \mathbb{F}_{q^m}^n$ *with* $\|\boldsymbol{g}\|_q = n$. *For* $i \in \mathbb{N}$ *such that* $i \leq n - k - 1$,

$$\Lambda_i(\mathfrak{G}_k(\boldsymbol{g})) = \mathfrak{G}_{k+i}(\boldsymbol{g}).$$

On the other hand, if $\boldsymbol{G}$ is a randomly-drawn matrix, we obtain something quite different. Overbeck formulated a successful attack against GPT using these properties.

Lemma

If $\mathcal{C} \subset \mathbb{F}_{q^m}^n$ *is a code generated by a random matrix* $\boldsymbol{G} \in \mathbb{F}_{q^m}^{k \times n}$,

$$\dim \Lambda_i(\mathcal{C}) = \min\{n, (i+1)k\}$$

with high probability.

Reparation by Loidreau

Crucial to the structural attack against GPT are:

- The distinguishing properties of $\mathfrak{G}_k(\boldsymbol{g})$ under Λ_i.
- The invariance of $\boldsymbol{P} \in \mathrm{GL}_{n+l}(\mathbb{F}_q)$ under the Frobenius automorphism: $\boldsymbol{P}^{[i]} = \boldsymbol{P}$.

Loidreau tackled these issues by

1. Letting $\boldsymbol{P} \in \mathrm{GL}_{n+l}(\mathcal{W})$, where $\mathcal{W}$ is a λ-dimensional subspace of $\mathbb{F}_q^m$.
2. Using $\boldsymbol{P}^{-1}$ to scramble and $\boldsymbol{P}$ to unscramble instead of the other way around.

This has the effect of resisting the structural attack at the cost of reducing the error-correcting capacity by a factor of λ.

No Structural Exploits

Our best attacks against this reparation are *generic*. The state-of-the-art comes from Gaborit et al.; we will formulate and assess an ISD-inspired algorithm in the rank metric.

Contents

Support-Trapping Algorithm by Gaborit et al.

- $\mathrm{supp}(\boldsymbol{e}) :=$ subspace $\mathcal{E}$ which contains each e_i.
- Assume $\|\boldsymbol{e}\|_q \leq w \implies \mathcal{E}$ is w.l.o.g. w-dimensional.
- However, we may need to guess $\mathcal{E}' > \mathcal{E}$ to be able to solve for $\boldsymbol{e}$.
- Gaborit et al. showed how to retrieve $\boldsymbol{e}$ in time $\mathcal{O}(q^{w\lceil mk/n \rceil})$ by:
  1. Taking $\mathcal{E}'$ to be of dimension $\lceil (n-k)m/n \rceil$; and
  2. Writing each e_i in $\mathcal{E}'$ and solving the resulting linear system.

An ISD-Inspired Variation

Instead of guessing a superspace of $\mathcal{E}$, we will — analogously to ISD algorithms in the Hamming metric — guess a subspace that is "near-orthogonal" to $\mathcal{E}$.

Intuition Behind ISD in the Rank Metric

Instead of guessing a superspace of $\mathcal{E}$, we guess $\mathcal{V} \leq \mathbb{F}_q^m$ that is

1. "Near-orthogonal" to $\mathcal{E}$; and
2. Allows us to solve for each e_i.

Assume that

- $\mathcal{V}$ is ζ-dimensional.
- $\boldsymbol{P} \in \mathbb{F}_q^{\zeta \times m}$ projects from $\mathbb{F}_q^m$ to $\mathcal{V}$.
- $\tilde{\mathcal{V}}$ is a p-dimensional subspace of $\mathcal{V}$, for a parameter $p \in \{0, \ldots, \zeta\}$.
- $\langle \tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_p \rangle$ is a basis for $\tilde{\mathcal{V}}$.

Description of the Algorithm

Key ISD Equation

$$\{\boldsymbol{P}\overbrace{(y_i - (\boldsymbol{xG})_i)}^{e_i} = \sum_{j=1}^{p} \gamma_{i,j}\tilde{\boldsymbol{v}}_j : i \in \{1, \dots, n\}\} \tag{3}$$

is a linear system over $\mathbb{F}_q$ in ζn equations and $mk + np$ unknowns
$\implies \zeta = \lceil mk/n \rceil + p$.

Now:

1. Assume $\|\boldsymbol{e}\|_q \leq w$.
2. Solve for $\boldsymbol{x}$ in (3).
3. If $\|\boldsymbol{y} - \boldsymbol{xG}\|_q \leq w$, then we have a successful iteration. Otherwise, try a new $\mathcal{V}$ along with all p-dimensional $\tilde{\mathcal{V}}$'s.

Complexity Analysis of the Algorithm

Setting Up the Stage

Informally, we need the

- Probability that "$w - p$ dimensions" of $\mathrm{supp}(\boldsymbol{e})$ come from $\mathcal{V}^{\perp}$.
- Probability that "p dimensions" of $\mathrm{supp}(\boldsymbol{e})$ come from $\tilde{\mathcal{V}}$.
- The cost of iterating through all p-dimensional $\tilde{\mathcal{V}}$'s.

(We can ignore the cost of solving for $\boldsymbol{x}$ as it is a simple linear system.)

A Counting Argument

In order to evaluate the two probabilities, we:

1. Count all "good" choices; and
2. Divide that number by the total number of choices for $\mathrm{supp}(\boldsymbol{e})$.

q-Binomial Coefficients

Counting Vector Spaces

Definition (q-Binomial Coefficient)

The *q-binomial coefficient* (also called *Gaussian binomial coefficient*) is defined by

$$\begin{bmatrix} m \\ r \end{bmatrix}_q = \begin{cases} \frac{(1-q^m)(1-q^{m-1})\cdots(1-q^{m-r+1})}{(1-q)(1-q^2)\cdots(1-q^r)} & r \leq m \\ 0 & r > m. \end{cases}$$

Furthermore, it satisfies

$$\begin{bmatrix} m \\ r \end{bmatrix}_q \in \Theta(q^{r(m-r)}).$$

The q-binomial coefficient counts the number of r-dimensional subspaces of an m-dimensional vector space over $\mathbb{F}_q$.

Complexity Analysis of the Algorithm

Final Results

We can see that the probability of a successful iteration is

$$\begin{bmatrix} \zeta \\ p \end{bmatrix}_q \begin{bmatrix} m-\zeta \\ w-p \end{bmatrix}_q \begin{bmatrix} m \\ w \end{bmatrix}_q^{-1},$$

while the cost of iterating through all $\tilde{\mathcal{V}}$'s is

$$\begin{bmatrix} \zeta \\ p \end{bmatrix}_q .$$

This gives an average complexity of

$$\begin{bmatrix} m-\zeta \\ w-p \end{bmatrix}_q^{-1} \begin{bmatrix} m \\ w \end{bmatrix}_q \in \Theta(q^{w\zeta+p(p+m-\zeta)}).$$

$m > \zeta \implies$ minimized when $p = 0$ and becomes the same as in the support-trapping algorithm:

$$\Theta(q^{w\lceil mk/n \rceil}).$$

Contents

Closing Thoughts

We:

- Presented a simpler and more general common form of the GPT public key.
- Formulated a “framework” similar to Lee–Brickell’s ISD algorithm in the Hamming metric that already achieves state-of-the-art performance.

We hope to:

- Provide an implementation of the algorithm in SageMath.
- Adapt improvements in the spirit of Stern and Dumer to the rank metric. In the Hamming metric, such improvements allow ISD to work in time $\mathcal{O}(2^{n/20})$.

Thank you for your attention.