Public-key cryptosystems based on linear codes

Report 95-30

Ernst M. Gabidulin

ISSN 0922-5641

Public-Key Cryptosystems Based on Linear Codes

Ernst M. Gabidulin

January 22, 1995

Contents

1. Introduction

In [1], a revolutionary concept was proposed to make public the encipherment key as well as the enciphering algorithm. To describe a cryptosystem with *Private Keys*, one should introduce three main sets:

The set $\mathcal{M} = \{\mathbf{m}\}$ of *plaintext* messages to transmit;

The set $\mathcal{C} = \{\mathbf{c}\}$ of *ciphertexts*;

The set $\mathcal{K} = \{\mathbf{k}\}$ of *encipherment keys*.

An *encryption function* is a one-one map

$$\mathbf{c} = \mathbf{E}(\mathbf{m}, \mathbf{k})$$

which turns a plaintext $\mathbf{m}$ into a ciphertext $\mathbf{c}$ after the key setting $\mathbf{k}$ has been applied.

Let $\mathbf{m}, \mathbf{c}$ and $\mathbf{k}$ be sequences over a finite alphabet of length not greater than n. Define the complexity of encryption $W(\mathbf{E})$ as the number of calculations to get $\mathbf{c}$ from $\mathbf{m}$, if $\mathbf{k}$ is given.

The complexity of encryption $W(\mathbf{E} \mid \mathbf{k})$ should be a *polynomial* in n of small degree.

A *decryption function* is the inverse map

$$\mathbf{m} = \mathbf{D}(\mathbf{c}, \mathbf{k})$$

wich turns the ciphertext $\mathbf{c}$ into the plaintext $\mathbf{m}$.

The complexity of decryption $W(\mathbf{D} \mid \mathbf{k})$ should be a *polynomial* of small degree in n, if the key $\mathbf{k}$ is known.

On the other hand, if the key $\mathbf{k}$ is unknown then both the encryption and the decryption should be much harder problems. Usually, it is desirable that the number of calculations $W(\mathbf{D})$ would be *not polynomial* in n, i.e. it is asymptotically greater than any power of n. Another possibility would be the case when the problem of the decryption with unknown $\mathbf{k}$ is a so-called NP-complete problem. These are problems for which a proposed solution which somehow has been derived - possibly in a non-constructive manner - can be checked with polynomial efforts.

Hence, if two persons A and B want to communicate in a secret manner, they choose a key $\mathbf{k}$ and keep it in secret. They can easily encrypt the plaintexts and decrypt the ciphertexts.

Suppose that an *enemy party* E does not know the secret key $\mathbf{k}$ but can intercept ciphertexts $\mathbf{c}$. The person E may want to either obtain the corresponding plaintexts $\mathbf{m}$'s or get the secret key $\mathbf{k}$ from $\mathbf{c}$.

The cryptosystem seems to be rather good if the number of calculations, or, the *work function* $W(\mathbf{D})$, is *not polynomial* or is infeasible from the practical point of view.

Cryptosystems with private (secret) keys are said to use **symmetric ciphers** because either both the encipherment key and the decipherment key are the same or the decipherment key can easily be calculated provided that the encipherment key is known.

Public Key systems are based on the concept of the so-called *one way functions.*

Suppose from now on, that a key $\mathbf{k}$ defines an **asymmetric cipher**. This means that the complexity of the encryption $W(\mathbf{E} \mid \mathbf{k})$ remains polynomial but the complexity of decryption $W(\mathbf{D} \mid \mathbf{k})$ is *not polynomial even* if the key $\mathbf{k}$ is known. That is the new idea introduced by Diffie and Hellman.

We refer to an encryption function with this property as a *one way function.*

It may be that there exists an extra key $\mathbf{t}$ such that $W(\mathbf{D} \mid \mathbf{k})$ is *not* polynomial but $W(\mathbf{D} \mid \mathbf{k}, \mathbf{t})$ is *polynomial.* In this case, the key $\mathbf{t}$ is known as a *trapdoor* and the encryption function is referred to as a *trapdoor one way function.* We assume that the key $\mathbf{k}$ can be calculated with polynomial efforts if the trapdoor $\mathbf{t}$ is given. But calculating the trapdoor $\mathbf{t}$ using the known key $\mathbf{k}$ should be hard.

Note that, up to now, no function has been *proved* to be one way or trapdoor one way. But there are some candidats, for example, the discret logarithm or the problem of factorizing integers.

One can use the features of trapdoor one way functions to construct a cryptosystem as follows.

General case

- The legitimate user chooses a secret key (trapdoor) $\mathbf{t}$ (an easy problem).

- Calculates the *public* key **k** (an easy problem).
- Publishes **k**.
- The sending party is requested to use the *public* key **k** for Encrypting (an easy problem).
- The legitimate user gets the plaintext using the public key **k** and the secret key **t** (an easy problem).
- The *Enemy* party has to decrypt the intercepted ciphertext using only the public key **k** (a hard problem).

Many public key cryptosystems were proposed during the last two decades. First of all, the system due to Merkle and Hellman [2] should be mentioned. It is based on a famous NP-problem which is known as the *knapsack problem*. Another famous public key cryptosystem, the RSA System, was invented by Rivest, Shamir and Adleman [3]. Its security lies in the difficulty of the problem of factoring large integers.

We focus our attention on public key cryptosystems based on *linear codes*.

Prof. McEliece was the first who proposed to use linear codes for Public-Key Cryptosystems (PKC) [4]. Later on, it was shown in [5] that the the problem of decoding a general linear code is NP-complete.

If a family of linear codes is to be used in PKC, is should possess the following features:

- It is be rich enough to avoid an exhaustive search when an enemy party wants to break the cryptosystem;
- Encoding (encryption) and decoding (decryption) is easy when a full description of a code is known;
- A full description of a code is very hard to obtain from open keys (usually, a scrambled generator matrix or a scrambled parity check matrix).

In the **linear code case**, the PKC is implemented as follows.

- The legitimate user chooses a code with large distance having a fast decoding algorithm and chooses a scramble matrix (an easy problem).
- Calculates a scrambled parity or generator matrix (an easy problem)
- Publishes the above matrix as the public key.
- The sending party is requested to use the above matrix for encrypting (an easy problem).

- The legitimate user gets the plaintext using the fast decoding algorithm (an easy problem).
- The *Enemy* party has to decode an intercepted ciphertext as a general code (a hard problem).

In [4], the family of codes consisted of binary Goppa codes and a scrambled generator matrix $\mathbf{G_{cr}} = \mathbf{SG}$ 2was used as a public key. A few more or less unsuccessful attacks on this PKC were proposed [6, 7].

The PKC based on a family of Generalized Reed-Solomon codes was proposed by Niederreiter [8]. The open key is a scrambled parity check matrix $\mathbf{H_{cr}} = \mathbf{SH}$ of a GRS code. This is a Knapsack-Type cryptosystem. Therefore, one might hope that this PKC is secure. But in fact, many knapsack-type PKC including the *Niederreiter* PKC and some of its modifications have recently be shown to be insecure. See [9, 10].

The PKC based on a family of rank codes [11] was proposed in [12]. It looks like a McEliece type PKC. An important difference is that an open key $\mathbf{G_{cr}} = \mathbf{SG} + \mathbf{X}$ is a sum of a scrambled generator matrix $\mathbf{SG}$ and a hiding matrix $\mathbf{X}$. In [12], a hiding matrix $\mathbf{X}$ of rank 1 was used. Recently, Gibson showed that for such hiding matrices a PKC can be broken for practical values of the parameters [13].

In this report, the use of hiding matrices is proposed to modify all the above PKC. The modified open keys are as follows:

- for the *McEliece* PKC: $\mathbf{G_{cr}^{mod}} = \mathbf{SG} + \mathbf{X}$, where $\mathbf{X}$ is a specific matrix of rank 1;
- for the *Niederreiter* PKC: $\mathbf{H_{cr}^{mod}} = \mathbf{S}(\mathbf{H} + \mathbf{X})$, where $\mathbf{X}$ is a specific matrix of rank 1;
- for the *GPT* PKC: $\mathbf{G_{cr}^{mod}} = \mathbf{SG} + \mathbf{X}$, where $\mathbf{X}$ is a specific matrix of rank $t_1 > 1$, with t_1 a design parameter.

The main idea is as follows. The legal party chooses some random matrix $\mathbf{X}$ as an extra secret key and adds it to the original public key to produce a new modified public key. Thus any visible structure of the public key will be hidden. Although there are strong restrictions in the choice of the random matrix $\mathbf{X}$, it can be done in practical applications.

The report is organized as follows: in Section 2, the background of codes over large alphabets is given; in Section 3, the *Niederreiter* PKC is presented, and a modified version of the Sidel'nikov-Shestakov attack, which slightly differs from the original one, is described. A modification of the *Niederreiter* PKC is also considered; in Section 4, the *McEliece* PKC is described, together with attacks and modifications; in Section 5, the *GPT* PKC and Gibson's attack are presented;

in Section 6, a comparison of all three PKCs is made from the point of view of the size of the public keys and the work functions; in Section 7, concluding remarks are made.

2. Background of Algebraic Coding

All the public key cryptosystems under consideration are based on codes over large alphabets. In fact, these codes are generalized Reed-Solomon codes and rank codes. In this section, background information on these codes is given (for more detailed information, see [17] and [11]).

Let $GF(q)$ be a finite field with q elements. The nth Cartesian power $GF(q)^n$ of $GF(q)$ is a metric space with respect to the *Hamming distance function*

$$d(\mathbf{x},\mathbf{y}) := \left|\{i \mid 1 \le i \le n,\ x_i \ne y_i\}\right|, \tag{1}$$

where $\mathbf{x} = (x) \in GF(q)^n$ and $\mathbf{y}$ =$(y_1, y_2, \ldots, y_n) \in GF(q)^n$

The *Hamming weight* $w_H(x)$ of an element $\mathbf{x} \in GF(q)^n$ is the number of *nonzero* positions in $\mathbf{x}$, or, equivalently, the integer

$$w_H(x) := d(x), \tag{2}$$

where $\mathbf{o}$ is the zero-vector in $GF(q)^n$.

A *linear* code $\mathcal{C}$ of *length* n and *dimension* k over $GF(q)$ is a k-dimensional linear subspace of $GF9q)^n$.

The *(minimum) distance* of a *linear* code $\mathcal{C}$ is defined by

$$d := \min\left\{d(x) \mid \mathbf{x} \in \mathcal{C},\ \mathbf{y} \in \mathcal{C},\ \mathbf{x} \ne \mathbf{y}\right\}, \tag{3}$$

or, equivalently,

$$d := \min\left\{w_H(x) \mid \mathbf{x} \in \mathcal{C},\ \ \mathbf{x} \ne \mathbf{o}\right\}. \tag{4}$$

If a code $\mathcal{C}$ has distance d, then we can correct all the errors $\mathbf{e}$ with $w_H(\mathbf{e}) \le t = \lfloor (d-1)/2 \rfloor$. This means the following. Let us consider a vector $\mathbf{y} = \mathbf{g}_1 + \mathbf{e}$, where $\mathbf{g}_1$ is a code vector. Then y is strictly closer to $\mathbf{g}_1$ than to any other code vector $\mathbf{g}_2$ if $w_H(\mathbf{e}) \le t = \lfloor (d-1)/2 \rfloor$. Hence, we can uniquely recover $\mathbf{g}_1$ from $\mathbf{y}$.

A *linear* code $\mathcal{C}$ of dimension k and distance d is referred to as an (n,k,d)-code.

A *linear* code $\mathcal{C}$ can be given in terms of a *generator matrix* $\mathbf{G}$ or in terms of a *parity check matrix* $\mathbf{H}$.

Let $\{\mathbf{g}_i = (g_{i,1}, g_{i,2}, \ldots, g_{i,n}),\ i = 1, 2, \ldots, k,\ \mathbf{g}_i \in GF(q)^n\}$ be a set of vectors which are linearly independent over $GF(q)$. Define the $k \times n$ $\mathbf{G}$ of rank k by

$$\mathbf{G} = [g_{i,j}], \quad i = 1, 2, \ldots, k; \quad j = 1, 2, \ldots, n. \tag{5}$$

The *code vectors*, or *codewords* of the *linear* code $\mathcal{C}$ with *generator matrix* $\mathbf{G}$ are defined to be the linear combinations of rows of $\mathbf{G}$ with coefficients in $GF(q)$. In other words, let $\mathbf{m} = (m_1, m_2, \ldots, m_k)$ be a k-string of elements of $GF(q)$. Such an $\mathbf{m}$ is often called an *information sequence.* Then the corresponding codeword is given by

$$\mathbf{g}\,(\mathbf{m}) = \mathbf{mG}. \tag{6}$$

On the other hand, to any matrix $\mathbf{G}$ of rank k, there exists a $(n-k) \times n$ matrix $\mathbf{H}$ of rank $r = n - k$ such that

$$\mathbf{GH}^t = \mathbf{O}_k^{n-k}, \tag{7}$$

where $\mathbf{H}^t$ denotes transposed matrix and $\mathbf{O}_k^{n-k}$ means the all-zero matrix of size $k \times (n-k)$. The matrix $\mathbf{H}$ is known as a *parity check matrix* for $\mathcal{C}$. Each codeword $\mathbf{g} = (g_1, g_2 \ldots, g_n)$ can be obtained as a solution of a linear system of equations in the variables $g_1, g_2 \ldots, g_n$:

$$\mathbf{gH}^t = (g_1, g_2, \ldots, g_n)\mathbf{H}^t = \mathbf{O}_1^{n-k}. \tag{8}$$

The *linear* code $\mathcal{C}$ has distance d if and only if any $d-1$ columns of the Parity check matrix $\mathbf{H}$ are linearly independent over $GF(q)$ and there exists d columns which are linearly dependent.

The Singleton bound: for any *linear* code $\mathcal{C}$ we have

$$d \le r + 1, \tag{9}$$

where r denotes the rank of $\mathbf{H}$ over $GF(q)$.

2.1. Generalized Reed-Solomon codes. *Generalized Reed-Solomon* codes (GRS codes) are defined by a parity check matrix which is a generalized *Vandermonde* rectangular $r \times n$-matrix

$$\mathbf{H} = \mathbf{VZ} = \begin{bmatrix} z_1 & z_2 & \ldots & z_n \\ z_1x_1 & z_2x_2 & \ldots & z_nx_n \\ z_1x_1^2 & z_2x_2^2 & \ldots & z_nx_n^2 \\ \ldots & \ldots & \ldots & \ldots \\ z_1x_1^{r-1} & z_2x_2^{r-1} & \ldots & z_nx_n^{r-1} \end{bmatrix}, \tag{10}$$

where $\mathbf{Z} = \mathbf{diag}\,(z_1, z_2, \ldots, z_n)$ denotes a diagonal matrix with non-zero diagonal elements and $\mathbf{V}$ denotes a *Vandermonde* rectangular $r \times n$-matrix

$$\mathbf{V} = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ x_1 & x_2 & \ldots & x_n \\ x_1^2 & x_2^2 & \ldots & x_n^2 \\ \ldots & \ldots & \ldots & \ldots \\ x_1^{r-1} & x_2^{r-1} & \ldots & x_n^{r-1} \end{bmatrix}, \tag{11}$$

where the x_j's are different.

If the number of rows r is given, we can join, by convention, to $GF(q)$ a formal element x_∞. We shall operate with this element by the following agreement. Let

$$F(x) = a_{r-1}x^{r-1} + a_{r-2}x^{r-2} + \ldots + a_1 x + a_0$$

be a polynomial of degree not greater than $r-1$. Then the value of this polynomial at the point x_∞ is defined to be the leading coefficient of this polynomial:

$$F(x_\infty) = a_{r-1}.$$

Hence, $(x_\infty)^m = 0$, if $m = 0, 1, \ldots, r-2$, but $(x_\infty)^{r-1} = 1$. Thus, the *Vandermonde* matrix can contain a column $(0, 0, \ldots, 0, 1)^t$. From this, it follows that the maximal number of different columns of the *Vandermonde* matrix is equal to $|GF(q) \cup x_\infty| = q + 1$.

We have

Property 1. *Any square $r \times r$-submatrix of $\mathbf{V}$ is a non singular (Vandermonde) square matrix. An analogous statement is true for the generalized matrix $\mathbf{H}$.*

Conjecture 1. *If an $r \times n$ matrix with $n = q + 1$ possess* **Property 1**, *then this matrix is a generalized Vandermonde matrix (except for the cases $q = 2^m$, $r = 3$ or $r = n - 3$).*

All the code vectors $\mathbf{g}$ are solutions of the system

$$\mathbf{g}\mathbf{H}^t = (g_1, g_2 \ldots, g_n)\mathbf{H}^t = \mathbf{O}_1^{n-k}. \tag{12}$$

Evidently, the rank of $\mathbf{H}$ is equal to r and, by **Property 1**, any r columns are linearly independent. Thus, the Parity check matrix $\mathbf{H}$ defines an optimal linear code reaching the Singleton bound (9). This code has the following parameters:

1. code length $n \le q + 1$;
2. dimension $k = n - r$;
3. distance $d = r + 1 = n - k + 1$.

Such codes are known as *maximal distance separable*, or *MDS* codes.

The canonical row-reduced echelon form

Multiply the generalized *Vandermonde* rectangular $r \times n$-matrix $\mathbf{H}$ to the left by $\mathbf{F} = \mathbf{H}_1^{-1}$, where

$$\mathbf{H}_1 = \mathbf{V}_1\mathbf{Z}_1 = \begin{bmatrix} z_1 & z_2 & \ldots & z_r \\ z_1x_1 & z_2x_2 & \ldots & z_rx_r \\ z_1x_1^2 & z_2x_2^2 & \ldots & z_rx_r^2 \\ \ldots & \ldots & \ldots & \ldots \\ z_1x_1^{r-1} & z_2x_2^{r-1} & \ldots & z_rx_r^{r-1} \end{bmatrix}. \tag{13}$$

To obtain the inverse matrix $\mathbf{H}_1^{-1}$, define the *Lagrange* interpolation polynomials of degree $r-1$ by

$$f_i(x) = \prod\nolimits_{1\le s\le r, s\ne i} \frac{(x-x_s)}{(x_i-x_s)} = \sum\nolimits_{s=1}^{r} f_{is}x^{s-1}, \tag{14}$$

$$i = 1, 2, \cdots, r,$$

Note that

$$f_i(x_j) = \sum\nolimits_{s=1}^{r} f_{is}x_j^{s-1} = \delta_{ij} = \begin{cases} 1, & \text{if } j = i, \\ 0, & \text{if } j \ne i, \end{cases} \tag{15}$$

$$i, j = 1, 2, \cdots, r.$$

Define the square $r\times r$-matrix $\mathbf{F}$ by

$$\mathbf{F} = \left[\frac{f_{ij}}{z_i}\right], \ i, j = 1, 2, \ldots, r. \tag{16}$$

It follows from Eq. (15) that

$$\mathbf{F} = \mathbf{H}_1^{-1} \tag{17}$$

because

$$(\mathbf{FH}_1)_{ij} = \sum_{s=1}^{r} \left(\frac{f_{ij}}{z_i}\right)\left(z_jx_j^{s-1}\right) = \frac{z_j}{z_i}\sum_{s=1}^{r} f_{is}x_j^{s-1} = \delta_{ij}.$$

We obtain the *canonical row-reduced echelon form*

$$\mathbf{H}_{sys} = \mathbf{FH} = [\mathbf{E}_r\ \mathbf{R}], \tag{18}$$

where $\mathbf{E}_r$ denotes the $r\times r$ identity matrix, and $\mathbf{R}$ is the following $r\times(n-r)$-matrix

$$\mathbf{R} = [p_{ij}] = \left[\frac{z_j}{z_i}f_i(x_j)\right] =$$

$$\left[\frac{z_j}{z_i}\sum_{s=1}^{r} f_{is}x_j^{s-1}\right], \tag{19}$$

$$i = 1, 2, \ldots, r;\ j = r+1, r+2, \ldots, n.$$

On the other hand,

$$f_i(x_j) = \frac{1}{z_i}\prod_{1\le s\le r, s\ne i} z_s\frac{(x_j-x_s)}{(x_i-x_s)} = \frac{1}{z_i}\frac{\prod\limits_{s=1}^{r}(x_j-x_s)}{\prod\limits_{s=1, s\ne i}^{r}(x_i-x_s)}\frac{1}{x_i-x_j} = \frac{a_ib_j}{x_i-x_j}, \tag{20}$$

where

$$a_i = \left(\prod_{s=1, s\neq i}^{r} (x_i - x_s) \right)^{-1}, \ i = 1, 2, \ldots, r;$$
$$b_{j=} \prod_{s=1}^{r} (x_j - x_s), \ j = r+1, r+2, \ldots, n. \tag{21}$$

Hence,

$$\mathbf{R} = \left[\frac{\left(\frac{a_i}{z_i}\right)(z_j b_j)}{x_i - x_j} \right] = \left[\frac{\widetilde{a}_i \widetilde{b}_j}{x_i - x_j} \right], \ i = 1, 2, \ldots, r; \ j = r+1, r+2, \ldots, n, \tag{22}$$

which means that the matrix $\mathbf{R}$ in Eq. 18 - Eq. 20 is a *generalized Cauchy matrix.*

It is well known that a generalized *Cauchy* matrix with different the x_i's and x_j's has the

Property 2. *Any square submatrix of the matrix* $\mathbf{R}$ *of any order is nonsingular.*

The generalized *Cauchy* matrix $\mathbf{R}$ in (22) can be extended with one column $(\widetilde{a}_1, \widetilde{a}_2, \ldots, \widetilde{a}_r)^t$ keeping **Property 2**. Hence, for the extended Generalized *Cauchy* matrix the maximal possible value of the sum $r + k$ is equal to $q + 1$.

Consider the matrix

$$\mathbf{W} = \left[\left((\mathbf{R})_{i,j} \right)^{-1} \right] = \left[\frac{x_i - x_j}{\widetilde{a}_i \widetilde{b}_j} \right],$$
$$x_i \neq x_j, \ i = 1, 2, \ldots, r; \ j = 1, 2, \ldots, k. \tag{23}$$

It is easy to see, that all elements of this matrix are non-zero, all determinants of order 2 are non-zero but all determinants of order 3 or greater are equal to 0.

Conjecture 2. *Let* $\mathbf{R} = [R_{i,j}]$ *be an* $r \times k$*-matrix possessing* **Property 2**. *Let* $r + k = q + 1$. *Let the matrix* $\mathbf{W} = \left[\left((\mathbf{R})_{i,j} \right)^{-1} \right]$ *have the property that all the* 1×1 *and* 2×2*-submatrices are non-singular but all the* 3×3 *submatrices are singular. Then the matrix* $\mathbf{R}$ *is an extended Generalized Cauchy matrix.*

2.2. Generalized Reed-Solomon codes: Fast Decoding Algorithms.

The main Coding Theory problems are as follows:

- The problem of finding *optimal* (n, k, d)-codes, i.e., codes with the maximal possible distance d, if n and k are given, or with maximal possible dimension k, if n and d are given;
- The problem of *Error Correcting Decoding:* given an (n, k, d)-code, to find a simple method to correct all errors $\mathbf{e}$ with $w_H(\mathbf{e}) \leq \lfloor (d-1)/2 \rfloor$;

- The more complicated type of Decoding known as the *Minimal Distance Decoding:* given an (n, k, d)-code, to find a simple method to identify for any received vector $\mathbf{y}$ a code vector $\mathbf{g}$ that has minimal distance to $\mathbf{y}$.

The Eq.'s (10) and (8) solve the first problem for the parameters listed above. We present a few solutions to the problem of the *Error Correcting Decoding,*. i.e., correcting of errors. The problem of the *Minimal Distance Decoding* is unsolved for GRS codes.

General remarks. Let $\mathbf{y} = \mathbf{g} + \mathbf{e}$ be a code vector $\mathbf{g}$ corrupted by an error vector $\mathbf{e}$. Calculate the product $s = y\mathbf{H}^t$ known as the *syndrome* vector of y:

$$\mathbf{s} = \mathbf{yH}^t = (g + e)\mathbf{H}^t = \mathbf{eH}^t. \tag{24}$$

Hence, the syndrome of $\mathbf{y}$ depends only on the error vector $\mathbf{e}$, not on the code vector $\mathbf{g}$.

First, consider the case when $z_1 = z_2 = \ldots = z_n = 1$. This is the case of the ordinary Reed-Solomon codes.

Let $\mathbf{e}$ be an error vector of weight $w_H(\mathbf{e}) \leq m \leq t = \lfloor (d-1)/2 \rfloor$ and let $\{i_1, i_2, \ldots, i_m\}$ be an index set that contains the error positions. Then the syndrome $\mathbf{s}$ can be represented as

$$\mathbf{s} = (s_0, s_1, \ldots, s_{r-1}) = \mathbf{eH}^t =$$

$$(e_{i_1}, e_{i_2}, \ldots, e_{i_m}) \begin{bmatrix} 1 & x_{i_1} & x_{i_1}^2 & \ldots & x_{i_1}^{r-1} \\ 1 & x_{i_2} & x_{i_2}^2 & \ldots & x_{i_2}^{r-1} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_{i_m} & x_{i_m}^2 & \ldots & x_{i_m}^{r-1} \end{bmatrix}, \tag{25}$$

where the e_{i_s}'s are the values of the errors. The positions $i_1, i_2, \ldots, i_m$ are unknown but there exists a one-one correspondence between i and x_i. So, if we know the value of x_i, then we know i. For this reason, the x_i's are known as the *error locators*. Denote, for any i_s, $e_{i_s} := u_s$, $x_{i_s} := Y_s$. By Eq. (25), we get the following system of r non-linear equations in the $2m$ variables $u_1, u_2, \ldots, u_m$ and $Y_1, Y_2, \ldots, Y_m$:

$$(u_1, u_2, \ldots, u_m) \begin{bmatrix} 1 & Y_1 & Y_1^2 & \ldots & Y_1^{r-1} \\ 1 & Y_2 & Y_2^2 & \ldots & Y_2^{r-1} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & Y_m & Y_m^2 & \ldots & Y_m^{r-1} \end{bmatrix} = (s_0, s_1, \ldots, s_{r-1}), \tag{26}$$

where the right hand side is known. Note that the integer $w_H(\mathbf{e})$ also is unknown.

The Problem of Error Correcting Decoding is finding a simple method of solving this system (if a solution exists). We describe two simple methods in the subsections below.

The general case of decoding a GRS code can be reduced to the above case. In general, we have instead of Eq. (25) the following equation

$$s = (s_0, s_1, \dots, s_{r-1}) = e\mathbf{H}^t =$$

$$(e_{i_1}, e_{i_2}, \dots, e_{i_m}) \begin{bmatrix} z_{i_1} & z_{i_1}x_{i_1} & z_{i_1}x_{i_1}^2 & \dots & z_{i_1}x_{i_1}^{r-1} \\ z_{i_2} & z_{i_2}x_{i_2} & z_{i_2}x_{i_2}^2 & \dots & z_{i_2}x_{i_2}^{r-1} \\ \dots & \dots & \dots & \dots & \dots \\ z_{i_m} & z_{i_m}x_{i_m} & z_{i_m}x_{i_m}^2 & \dots & z_{i_m}x_{i_m}^{r-1} \end{bmatrix}. \tag{27}$$

Denote, for any i_s, $e_{i_s}z_{i_s} := \widetilde{u}_s$, $x_{i_s} := Y_s$. Again, by Eq. (27), we obtain

$$(\widetilde{u}_1, \widetilde{u}_2, \dots, \widetilde{u}_m) \begin{bmatrix} 1 & Y_1 & Y_1^2 & \dots & Y_1^{r-1} \\ 1 & Y_2 & Y_2^2 & \dots & Y_2^{r-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & Y_m & Y_m^2 & \dots & Y_m^{r-1} \end{bmatrix} = (s_0, s_1, \dots, s_{r-1}). \tag{28}$$

When the $\widetilde{u}_s'$s and Y_s's are found, then the values of the errors are calculated by $e_{i_s} = \widetilde{u}_s / z_{i_s}$.

The Peterson Algorithm. The first method of solving (26) was found by Peterson. It is known as the *matrix method.* The Peterson algorithm is well-defined if the number of errors does not exceed $t = \lfloor (d-1)/2 \rfloor = \lfloor r/2 \rfloor$.

First, we describe how to find the actual number of errors $w_H(\mathbf{e})$.

Consider the successive m-strings of the s's. It follows from (26), that

$$\begin{aligned} (1, 1, \dots, 1)\,\mathbf{UY} &= (s_0, s_1, \dots, s_{m-1}) \\ (Y_1^1, Y_2^1, \dots, Y_m^1)\,\mathbf{UY} &= (s_1, s_2, \dots, s_m) \\ \dots \\ \left(Y_1^j, Y_2^j, \dots, Y_m^j\right)\mathbf{UY} &= (s_j, s_{j+1}, \dots, s_{j+m-1}), \\ \dots \\ \left(Y_1^{r-m}, Y_2^{r-m}, \dots, Y_m^{r-m}\right)\mathbf{UY} &= (s_{r-m}, s_{r-m+1}, \dots, s_{r-1}) \end{aligned} \tag{29}$$

where $\mathbf{U}$ denotes the diagonal matrix $\mathbf{U} = \mathbf{diag}\,(u_1, u_2, \dots, u_m)$ and

$$\mathbf{Y} = \begin{bmatrix} 1 & Y_1 & Y_1^2 & \dots & Y_1^{m-1} \\ 1 & Y_2 & Y_2^2 & \dots & Y_1^{m-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & Y_m & Y_m^2 & \dots & Y_m^{m-1} \end{bmatrix} \tag{30}$$

denotes the transposed square *Vandermonde* matrix.

Putting $\mathbf{u} := (u_1, u_2, \dots, u_m)$, we have the equivalent representation

$$\mathbf{u}\begin{bmatrix} 1 & Y_1 & Y_1^2 & \dots & Y_1^{m-1} \\ 1 & Y_2 & Y_2^2 & \dots & Y_1^{m-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & Y_m & Y_m^2 & \dots & Y_m^{m-1} \end{bmatrix} = \mathbf{uY} = (s_0, s_1, \dots, s_{m-1}),$$

$$\mathbf{u}\begin{bmatrix} Y_1 & Y_1^2 & Y_1^3 & \dots & Y_1^{m} \\ Y_2 & Y_2^2 & Y_2^3 & \dots & Y_1^{m} \\ \dots & \dots & \dots & \dots & \dots \\ Y_m & Y_m^2 & Y_m^3 & \dots & Y_m^{m} \end{bmatrix} = \mathbf{uDY} = (s_1, s_1, \dots, s_m),$$

$\dots$

$$\mathbf{u}\begin{bmatrix} Y_1^{m-1} & Y_1^{m} & Y_1^{m+1} & \dots & Y_1^{2m-1} \\ Y_2^{m-1} & Y_2^{m} & Y_2^{m+1} & \dots & Y_1^{2m-1} \\ \dots & \dots & \dots & \dots & \dots \\ Y_m^{m-1} & Y_m^{m} & Y_m^{m+1} & \dots & Y_m^{2m-1} \end{bmatrix} = \mathbf{uD}^{m-1}\mathbf{Y} = (s_{m-1}, s_m, \dots, s_{2m-1}),$$

$\dots$

$$\mathbf{u}\begin{bmatrix} Y_1^{r-m} & Y_1^{r-m+1} & Y_1^{r-m+2} & \dots & Y_1^{r-1} \\ Y_2^{r-m} & Y_2^{r-m+1} & Y_2^{r-m+2} & \dots & Y_1^{r-1} \\ \dots & \dots & \dots & \dots & \dots \\ Y_m^{r-m} & Y_m^{r-m+1} & Y_m^{r-m+2} & \dots & Y_m^{r-1} \end{bmatrix} = \mathbf{uD}^{r-m}\mathbf{Y} = (s_{r-m}, s_1, \dots, s_{r-1}), \tag{31}$$

where $\mathbf{Y}$ is defined above and $\mathbf{D} = \mathbf{diag}\,(Y_1, Y_2, \dots, Y_m)$.

Suppose that the actual number of errors is strictly less than m. Then the diagonal matrix $\mathbf{U}$ in (29) is singular because some diagonal elements are equal to 0. This means that the m-strings $(s_0, s_1, \dots, s_{m-1})$, $(s_1, s_2, \dots, s_m)$, ..., $(s_{m-1}, s_m, \dots, s_{2m-1})$ are linearly dependent over $GF(q)$ and that the determinant of the matrix

$$\mathbf{M}_m = \begin{bmatrix} s_0 & s_1 & \dots & s_{m-1} \\ s_1 & s_2 & \dots & s_m \\ \dots & \dots & \dots & \dots \\ s_{m-1} & s_m & \dots & s_{2m-1} \end{bmatrix} \tag{32}$$

is equal to 0. Thus the beginning of the Peterson algorithm is as follows.

1 After receiving the vector $\mathbf{y}$ calculate the syndrome $\mathbf{s}$. If $\mathbf{s} = \mathbf{o}$, there are no errors.

2 If $\mathbf{s} \neq \mathbf{o}$, then calculate $\det \mathbf{M}_m$ for $m = t = [(d-1)/2]$. If $\det \mathbf{M}_t \neq 0$, then the number of errors is equal to t.

3 If $\det \mathbf{M}_t = 0$, then calculate $\det \mathbf{M}_m$ for $m = t - 1$, and so on. Continue till the first time, when $\det \mathbf{M}_m \neq 0$. This value m will be the real number of errors.

Now suppose that m is the real value of the number of errors. Then both the matrix $\mathbf{D}$ and the matrix $\mathbf{Y}$ in the representation (31) are non-singular. Moreover, the matrices $\mathbf{Y}, \mathbf{DY}, \ldots, \mathbf{D^{m-1}Y}$ are linearly independent over $GF(q)$. This means that the m-strings $(s_0, s_1, \ldots, s_{m-1})$, $(s_1, s_2, \ldots, s_m)$, ..., $(s_{m-1}, s_m, \ldots, s_{2m-1})$ are also linearly independent and that

$$\det \mathbf{M}_m \neq 0. \tag{33}$$

On the other hand, the matrices $\mathbf{Y}, \mathbf{DY}, \ldots, \mathbf{D^{m-1}Y}, \mathbf{D^{m}Y}$ are linearly dependent over $GF(q)$ because the characteristic polynomial of the diagonal matrix $\mathbf{D}$ is given by

$$\varphi_D(x) = (Y_1 - x)(Y_2 - x)\ldots(Y_m - x) = \sigma_m - \sigma_{m-1}x + \ldots + (-1)^m \sigma_0 x^m, \tag{34}$$

where the σ_i denotes the ith symmetrical function of the roots $Y_1, Y_2, \ldots, Y_m$.

Let us multiply the first line of Eq. (31) by σ_m, the second line by $-\sigma_{m-1}$,..., the mth line by $(-1)^{m-1}\sigma_1$, the $(m+1)$th line by $(-1)^m \sigma_0 = (-1)^m$. Adding all lines, we get

$$\begin{aligned}
&\mathbf{u}\varphi_D(\mathbf{D})\mathbf{Y} = \mathbf{0}\\
&= \sigma_m (s_0, s_1, \ldots, s_{m-1})\\
&-\sigma_{m-1}(s_1, s_2, \ldots, s_m) + \ldots\\
&+(-1)^{m-1}\sigma_1 (s_{m-1}, s_m, \ldots, s_{2m-1}) +\\
&(-1)^m \sigma_0 (s_m, s_{m+1}, \ldots, s_{2m})
\end{aligned} \tag{35}$$

or, using the matrix $\mathbf{M}_m$ defined in (32), we obtain the following linear system in the variables $\sigma_m, \sigma_{m-1}, \ldots, \sigma_1$

$$(\sigma_m, \sigma_{m-1}, \ldots, \sigma_1)\mathbf{M}_m = -(-1)^m (s_m, s_{m+1}, \ldots, s_{2m}). \tag{36}$$

4 Solving this system, we get the σ_i's.

5 Solving Eq. (34)

$$\sigma_m - \sigma_{m-1}x + \ldots + (-1)^m \sigma_0 x^m = 0, \tag{37}$$

we get the roots Y_s's and, consequently, the i_s's.

6 Solving the linear system in the first line of Eq. (31), where the matrix $\mathbf{Y}$ is known now, we get the u_s's and the errors e_{i_s}.

This concludes the Peterson algorithm.

The number of calculations. The first three parts of the Peterson algorithm , namely, calculating determinants $\det \mathbf{M}_m$. $m = t, t-1, \ldots$, require at most $O(t^3)$ operations.

Solving the linear system Eq. (36) requires at most $O(t^3)$ operations.

Solving the algebraic equation Eq. (37) of order m requires at most $O(nt)$ operations. We simply can examine each of the elements of $GF(q)$ as a possible solution of Eq. (37).

Solving the linear system in the first line of Eq. (31), where the matrix $\mathbf{Y}$ is known, requires at most $O\left(t^3\right)$ operations.

Thus, the Peterson algorithm requires $O\left(t^3 + nt\right)$ operations. Usually, $t = cn$, where $c < 1 - \frac{1}{q}$ is the fraction of correctable errors. Hence, it requires at most $O\left(n^3\right)$ operations to solve the system (26) using the Peterson algorithm. Recent results on solving linear systems allow to reduce the number of calculations to $O\left(n^{2.4}\right)$ operations.

Fast Decoding Based on the Euclidean Division Algorithm - the Berlekamp-Massey Algorithm. There exists another algorithm of solving the system (26) based on the Euclidean division algorithm. This algorithm is due to Berlekamp, with a modification due to Massey. Therefore, this algorithm is known as the *Berlekamp-Massey* algorithm. Currently, there are a few versions of this algorithm which differ in the implementation of the Euclidean division algorithm.

We often shall consider rings of polynomials *mod* $g(x)$. From now on, we consider the case $g(x) = x^r$ and the quotient ring $GF(q)\left[x\right]/(x^r)$. In this ring, the polynomial

$$f(x) = 1 + x + x^2 + \ldots + x^{r-1} \tag{38}$$

has an inverse, namely, $h(x) = 1 - x$ because

$$(1-x)\left(1 + x + x^2 + \ldots + x^{r-1}\right) = 1 - x^r \equiv 1 \pmod{x^r}. \tag{39}$$

Thus, we can put

$$\frac{1}{1-x} := 1 + x + x^2 + \ldots + x^{r-1}. \tag{40}$$

Consider again the system (26). From now on, we assume that the integer $m \le t = \lfloor (d-1)/2 \rfloor$ is the actual number of errors. Introduce the syndrome polynomial

$$S(x) = s_0 + s_1 x + s_2 x^2 + \ldots + s_{r-1} x^{r-1}. \tag{41}$$

To get it, multiply both sides of Eq. (26) to the left by the vector $\left(1, x, x^2, \ldots, x^{r-1}\right)^t$. Then we obtain on the right hand side $S(x)$ and, using Eq. (40), on the left hand side the expression

$$\begin{gathered} u_1\left(1+(Y_1x)+(Y_1x)^2+\ldots+(Y_1x)^{r-1}\right)+ \\ u_2\left(1+(Y_2x)+(Y_2x)^2+\ldots+(Y_2x)^{r-1}\right)+ \\ \vdots \\ u_m\left(1+(Y_mx)+(Y_mx)^2+\ldots+(Y_mx)^{r-1}\right)\equiv \,, \end{gathered}$$

$$\sum_{s=1}^{m}\frac{u_s}{1-xY_s} \pmod{x^r}$$

Hence, instead of Eq. (26), we get the equivalent polynomial equation

$$\sum_{s=1}^{m}\frac{u_s}{1-xY_s}\equiv S(x) \pmod{x^r}. \tag{42}$$

Now introduce the *error locator polynomial*

$$\Lambda(x)=\prod_{s=1}^{m}(1-xY_s). \tag{43}$$

The degree of this polynomial is m. Its roots are the inverses of error locators Y_s's.

Multiply Eq. (42) by $\Lambda(x)$. Define the *error evaluator polynomial* $\Omega(x)$ as the product

$$\Omega(x)=\Lambda(x)\sum_{s=1}^{m}\frac{u_s}{1-xY_s}=\sum_{s=1}^{m}u_s\prod_{\substack{j=1,\\ j\neq s}}^{m}(1-xY_j). \tag{44}$$

The degree of $\Omega(x)$ is less than or equal to $m-1$. For any s, $s=1,2,\ldots,m$, this polynomial takes at the point $x=Y_s^{-1}$ the value

$$\Omega\left(Y_s^{-1}\right)=u_s\prod_{\substack{j=1,\\ j\neq s}}^{m}\left(1-Y_s^{-1}Y_j\right). \tag{45}$$

Hence, if we know $\Omega(x)$ and Y_j's, we can evaluate the values of the errors by

$$u_s=\frac{\Omega\left(Y_s^{-1}\right)}{\prod\limits_{\substack{j=1,\\ j\neq s}}^{m}\left(1-Y_s^{-1}Y_j\right)}. \tag{46}$$

Combining Eqs. (42)-(44), we get the famous Berlekamp *Key Equation*

$$\Omega(x)\equiv\Lambda(x)S(x) \pmod{x^r}, \tag{47}$$

where

$$\deg \Lambda(x) = m \leq t = \left\lfloor \frac{d-1}{2} \right\rfloor ,$$

$$\deg \Omega(x) \leq m-1, \tag{48}$$

$$\gcd(\Omega(x). \ \Lambda(x)) = 1.$$

The problem of decoding is reduced to solving the *Key Equation* (47) with the restrictions (48). This means that we have to find the unknown polynomials $\Lambda(x)$ and $\Omega(x)$, if the polynomial $S(x)$ is given. When the polynomial $\Lambda(x)$ is found, we get its roots Y_s^{-1}'s and, consequently, the error locators X_{i_s}'s. After that, we get the values of the errors $e_{i_s} = u_s$ from Eq. (46).

Solving the Key Equation is based on the Euclidean division algorithm. For reference, we give the description of this algorithm.

The Euclidean division algorithm. The Euclidean division algorithm is a procedure for finding the greatest common divisor of two polynomials. Consider two polynomials $f(x)$ and $g(x)$ over $GF(q)$. Suppose that

$$\deg g(x) \leq \deg f(x).$$

For convenience, denote $F_{-1}(x) := f(x)$, $g(x) := F_0(x)$.

1st step Divide $F_{-1}(x)$ $(= f(x))$ by $F_0(x)$ $(= g(x))$:

$$F_{-1}(x) = G_1(x)F_0(x) + F_1(x), \text{ where } \deg F_1 < \deg F_0. \tag{49}$$

2nd step Divide $F_0(x)$ $(= g(x))$ by $F_1(x)$:

$$F_0(x) = G_2(x)F_1(x) + F_2(x), \text{ where } \deg F_2 < \deg F_1. \tag{50}$$

3rd Divide $F_1(x)$ by $F_2(x)$:

$$F_1(x) = G_3(x)F_2(x) + F_3(x), \text{ where } \deg F_3 < \deg F_2. \tag{51}$$

i'th step Divide $F_{i-2}(x)$ by $F_{i-1}(x)$:

$$F_{i-2}(x) = G_i(x)F_{i-1}(x) + F_i(x), \text{ where } \deg F_i < \deg F_{i-1}. \tag{52}$$

next to the last step Divide $F_{p-2}(x)$ by $F_{p-1}(x)$:

$$F_{p-2}(x) = G_p(x)F_{p-1}(x) + F_p(x), \text{ where } \deg F_p < \deg F_{p-1}. \tag{53}$$

last step Divide $g(x) := F_{p-1}(x)$ by $F_p(x)$:

$$F_{p-1}(x) = G_{p+1}(x)F_p(x). \tag{54}$$

Then

$$\gcd\left(f(x),\ g(x)\right) = F_p(x). \tag{55}$$

At each step of the procedure, the current remainder $F_i(x)$ can be represented as a linear combination of the two previous remainders. Thus, it is possible to represent all remainders, including the last one $F_p(x)$, as the linear combinations of $f(x)$ $(= F_{-1}(x))$ and $g(x)$ $(= F_0(x))$:

$$F_i(x) = A_i(x)f(x) + B_i(x)g(x). \tag{56}$$

The polynomials $A_i(x)$ and $B_i(x)$ can be obtained from Eqs. (49)-(54) but there exists an inductive way of calculating them. Define two sequences of polynomials $\{A_i(x)\}$ and $\{B_i(x)\}$ by

$$A_i(x) = G_i(x)A_{i-1}(x) + A_{i-2}(x), \tag{57}$$

where, by definition, $A_{-1}(x) = 1$ and $A_0(x) = 0$, and

$$B_i(x) = G_i(x)B_{i-1}(x) + B_{i-2}(x), \tag{58}$$

where, by definition, $B_{-1}(x) = 0$ and $B_0(x) = 1$. Then Eq. (56) is true.

It is easy to obtain many relations concerning the polynomials defined above. The following properties are important for the decoding procedure:

1. For any i,

$$\gcd(A_i(x), B_i(x)) = 1. \tag{59}$$

2. For any i,

$$\begin{aligned} F_{-1}(x) &= B_{i+1}(x)F_i(x) + B_i(x)F_{i+1}(x), \\ F_0(x) &= A_{i+1}(x)F_i(x) + A_i(x)F_{i+1}(x). \end{aligned} \tag{60}$$

3. For any i,

$$F_i(x) = (-1)^{i-1}\left(A_i(x)F_{-1}(x) - B_i(x)F_0(x)\right). \tag{61}$$

In particular,

$$\gcd(f(x), g(x)) = F_p(x) = (-1)^{p-1}\left(A_p(x)F_{-1}(x) - B_p(x)F_0(x)\right). \tag{62}$$

4. For any i,

$$A_i(x)B_{i+1}(x) - A_{i+1}(x)B_i(x) = (-1)^{i-1}. \tag{63}$$

5. For any i,

$$\deg A_i(x) = \sum_{s=2}^{i} \deg G_i(x), \tag{64}$$

$$\deg B_i(x) = \sum_{s=1}^{i} \deg G_i(x) = \deg f(x) - \deg F_{i-1}(x), \tag{65}$$

$$\deg F_i(x) = \deg f(x) - \sum_{s=1}^{i+1} \deg G_i(x). \tag{66}$$

Solving the key equation. It follows from Eq. (47), that a polynomial $C(x)$ exists such that

$$\Omega(x) = C(x)x^r + \Lambda(x)S(x).$$

Hence, if we put $F_{-1}(x) = x$, $F_0(x) = S(x)$, we can apply the Euclidean division algorithm described in Eqs. (49)-(64), to get $\Omega(x)$ and $\Lambda(x)$:

1 Start with calculating the polynomials $F_i(x)$, $A_i(x)$, $B_i(x)$, $i = 1$.

2 (**The stop rule**) Continue the calculations till $i = m$ such that

$$\begin{aligned} &\deg F_{m-1}(x) \geq \left\lfloor \tfrac{d-1}{2} \right\rfloor \quad \text{but} \\ &\deg F_m(x) < \left\lfloor \tfrac{d-1}{2} \right\rfloor . \end{aligned} \tag{67}$$

3 Calculate

$$\begin{aligned} \Lambda(x) &= \tfrac{B_m(x)}{B_m(0)}, \\ & \qquad\qquad . \\ \Omega(x) &= \tfrac{F_m(x)}{B_m(0)} \end{aligned} \tag{68}$$

This is a solution of the *Key Equation* (47) with the restrictions (48) (see Eqs. (59), (61), (65)). (In fact, using (61), one can show that this solution is unique.)

We already explained how the knowledge of $\Lambda(x)$ and $\Omega(x)$ allows us to decode.

2.3. Subfield Subcodes: Alternant and Goppa Codes.

Alternant codes. Let $\mathcal{C}$ be an (n, k, d) GRS code over the field $GF(q)$ with $q = p^m$ elements. Consider the subcode $\mathcal{C}_a$ consisting of all code words with all coordinates in the base field $GF(p)$. This *subfield subcode* is called an *alternant* code.

In general, an alternant code $\mathcal{C}_a$ can be defined by the same parity check matrix (10) and the same linear system (8) with the restriction that all components g_j of a code vector $\mathbf{g}$ should belong to the base field $GF(p)$. But it is more convenient to rewrite the parity check matrix (10) over the large field $GF(p)$ as the equivalent Parity check matrix over the base field $GF(p)$. We consider a specific realization of the field $GF(p)$, say, by means of a basis $\{\omega_1, \omega_2, \ldots, \omega_m\}$. Hence, in this realization each element $\omega \in GF(p)$ can be represented in the form

$$\omega = a_1\omega_1 + a_2\omega_2 + \ldots + a_m\omega_m,$$

where all the coefficients $a_1, a_2, \ldots, a_m$ are in the base field $GF(p)$.

This induces a mapping

$$\mathcal{B} : \ GF(p^m) \ \rightarrow \ GF(p)^m \tag{69}$$

which maps each element ω of the large field into the column m-vector $(a_1, a_2, \ldots, a_m)^t$:

$$\mathcal{B}\left(\omega\right) = \left(a_1, a_2, \ldots, a_m\right)^t . \tag{70}$$

So a parity check matrix of the alternant code $\mathcal{C}_a$ over the base field $GF(p)$ can be obtained as

$$\widetilde{\mathbf{H}} = \mathcal{B}(\mathbf{H}) =$$

$$\begin{bmatrix} \mathcal{B}(z_1) & \mathcal{B}(z_2) & \dots & \mathcal{B}(z_n) \\ \mathcal{B}(z_1 x_1) & \mathcal{B}(z_2 x_2) & \dots & \mathcal{B}(z_n x_n) \\ \mathcal{B}(z_1 x_1^2) & \mathcal{B}(z_2 x_2^2) & \dots & \mathcal{B}(z_n x_n^2) \\ \dots & \dots & \dots & \dots \\ \mathcal{B}\left(z_1 x_1^{r-1}\right) & \mathcal{B}\left(z_2 x_2^{r-1}\right) & \dots & \mathcal{B}(z_n x_n^{r-1}) \end{bmatrix}. \tag{71}$$

It is clear that the number of columns in the matrix (71) is n and the number of rows is mr. If some columns of the original matrix $\mathbf{H}$ are linearly independent over the large field $GF(p^m)$, then the corresponding columns of $\widetilde{\mathbf{H}}$ still remain linearly independent over the base field $GF(p)$. Thus, any $d-1$ columns of $\widetilde{\mathbf{H}}$ are linearly independent over $GF(p)$, and the minimal distance of an alternant code $\mathcal{C}_a$ is at least $d = r+1$. On the other hand, the matrix $\widetilde{\mathbf{H}}$ may contain linearly dependent rows. So the rank r_a of $\widetilde{\mathbf{H}}$ is not greater than rm and sometimes may be strictly less.

Combining these features, we get the following parameters (n, k_a, d_a) of an alternant code $\mathcal{C}_a$:

- code length n (the same as that of $\mathcal{C}$);
- dimension $k_a = n - r_a \geq n - rm$;
- code distance $d_a \geq d = r + 1 = n - k + 1$.

In general, *alternant codes* form the rich family of quite good codes over the base field $GF(p)$. Some of them reach the Varshamov-Gilbert bound.

Fast decoding of *alternant codes* may be carried out just as for GRS codes (see the above Section). The only difference is that the code vector $\mathbf{g}$, the received vector $\mathbf{y}$, and the vector of errors $\mathbf{e}$ have coordinates in the base field $GF(p)$, not in the field $GF(p^m)$. It does not change any step of decoding procedure both in the Peterson algorithm and in the Berlekamp-Massey algorithm. Hence, *alternant codes* possess *fast decoding* algorithms.

Notice that in order to be able to carry out of a decoding algorithm, we should know not only the parity matrix $\widetilde{\mathbf{H}}$ over $GF(p)$ but also the parity check matrix $\mathbf{H}$ of the original GRS code over the large field $GF(p^m)$.

Goppa Codes. *Goppa* codes were proposed by Goppa about 30 years ago. Since then, they are very popular because *Goppa* codes have a quite good minimum distance (on the Varshamov-Gilbert bound) and possess fast decoding algorithms. It has been shown that *Goppa* codes are a particular type of *alternant* codes but the latter were proposed much later.

Again, consider a large field $GF(q)$ with $q = p^m$ elements and the ring of polynomials $GF(q)[x]$. Let $G(x) \in GF(q)[x]$. Introduce the quotient ring

$$R_G = GF(q)[x]/(G(x)) \tag{72}$$

of polynomials over $GF(q)$ mod $G(x)$. This ring R_G is not a field unless the polynomial $G(x)$ is irreducible.

However, if $\alpha \in GF(q)$ and $G(\alpha) \neq 0$, then the polynomial $x - \alpha$ is invertible in R_g (cf. Eqs. (38)-(39).) This can be shown by dividing the polynomial $G(x)$ by $x - \alpha$:

$$G(x) = F(x)(x) + G(\alpha). \tag{73}$$

It follows from Eq. (73) that

$$F(x)(x) + G(\alpha) \equiv 0 \bmod G(x), \tag{74}$$

or, equivalently,

$$\left[-G(\alpha)^{-1} F(x)\right](x) \equiv 1 \bmod G(x). \tag{75}$$

Hence, we can consider the expression

$$\frac{1}{x-\alpha} \tag{76}$$

as a polynomial:

$$\frac{1}{x-\alpha} := -G(\alpha)^{-1} F(x) = -G(\alpha)^{-1} \frac{G(x)-G(\alpha)}{x-\alpha} \qquad \bmod G(x). \tag{77}$$

Keeping this in mind, we define *Goppa* codes as follows.

Let

$$G(x) := G_0 + G_1 x + G_2 x^2 + \ldots + G_r x^r, \quad G_r \neq 0, \tag{78}$$

be a polynomial over $GF(p^m)$ of degree r and let $\alpha := (\alpha_1, \alpha_2, \ldots, \alpha_n)$, $n > r$, be a set of elements from $GF(p^m)$ such that $G(\alpha_j) \neq 0$. Let $\mathbf{g} := (g_1, g_2, \ldots, g_n)$ be a vector with components g_j from the *base* field $GF(p)$. The *Goppa* code $\mathcal{C}_G$ is a set of all the vectors $\mathbf{g}$ such that

$$\sum_{j=1}^{n} \frac{g_j}{x-\alpha_j} \equiv 0 \bmod G(x). \tag{79}$$

If the **Goppa polynomial** $G(x)$ is irreducible, the Goppa code $\mathcal{C}_G$ is called **irreducible**.

Goppa codes can be defined also in terms of GRS codes. To see this, let us consider the polynomial

$$G(\alpha)^{-1} \frac{G(x)-G(\alpha)}{x-\alpha} = \tag{80}$$
$$G(\alpha)^{-1}\left[G^{(1)}(\alpha) + \tfrac{G^{(2)}(\alpha)}{2!}(x) + \ldots + \tfrac{G^{(r)}(\alpha)}{r!}(x)^{r-1}\right],$$

where $r = \deg G(x)$ and $G^{(i)}(\alpha)$ denotes the ith derivative of $G(x)$ at the point $x = \alpha$. Since the polynomials $1, (x), (x)^2, \ldots, (x)^{r-1}$ are linearly independent in R_G, we get from Eq. (79):

A vector $\mathbf{g} = (g_1, g_2, \ldots, g_n)$ with components g_j from the *base* field $GF(p)$ is a code vector of the *Goppa* code $\mathcal{C}_G$ if and only if

$$\sum_{j=1}^{n} g_j G(\alpha_j)^{-1} \frac{G^{(r-i)}(\alpha_j)}{(r-i)!} = 0, \ i = 0, 1, \ldots, r-1. \tag{81}$$

On the other hand,

$$\begin{aligned} &\tfrac{G^{(r)}(\alpha_j)}{r!} = G_r \neq 0, \\ &\tfrac{G^{(r-1)}(\alpha_j)}{(r-1)!} = G_{r-1} + rG_r\alpha_j, \\ &\tfrac{G^{(r-2)}(\alpha_j)}{(r-2)!} = G_{r-2} + (r-1)G_{r-1}\alpha_j + \tfrac{r(r-1)}{2}G_r\alpha_j^2, \\ &\ldots \\ &G^{(1)}(\alpha_j) = G_1 + 2G_2\alpha_j + \ldots + rG_r\alpha_j^{r-1}. \end{aligned} \tag{82}$$

Hence, subtracting from the ith equation in Eq. (81) a suitable linear combination of the equations (82), we obtain the equivalent system

$$\sum_{j=1}^{n} g_j G(\alpha_j)^{-1} \alpha_j^i = 0, \ i = 0, 1, \ldots, r-1. \tag{83}$$

This is just the definition of the subfield subcode (an alternant code) of the GRS code with the parity check matrix $\mathbf{H} = \left[z_j x_j^i\right]$, where $z_j = G(\alpha_j)^{-1}$, $x_j = \alpha_j$, and $r = \deg G(x)$.

Therefore, *in general*, the parameters of *Goppa* codes are the same:

$$\begin{array}{ll} \text{Code length} & n \leq q + 1 = p^m + 1; \\ \text{Dimension} & k_G = n - r_G \geq n - rm; \\ \text{Code distance} & d_G \geq d = r + 1 = \deg G(x) + 1. \end{array} \tag{84}$$

But, in the binary case $p = 2$, *Goppa* codes have better parameters than general alternant codes.

To show this, consider again Eq. (79). In the binary case, $g_j = 0$ or 1. If $w_H(\mathbf{g}) = w$, $g_{j_1} = g_{j2} = \ldots = g_{j_w} = 1$, then we have for this code vector

$$\sum_{s=1}^{w} \frac{1}{x - \beta_s} \equiv 0 \qquad \mod G(x), \tag{85}$$

where $\beta_s := g_{j_s}$.

Note that the left hand side of Eq. (85) can be written as

$$\frac{\omega_{\mathbf{g}}^{(1)}(x)}{\omega_{\mathbf{g}}(x)}, \tag{86}$$

where the polynomial $\omega_{\mathbf{g}}^{(1)}(x)$ is the derivative of the polynomial $\omega_{\mathbf{g}}(x)$ given by

$$\omega_{\mathbf{g}}(x) = \prod_{s=1}^{w}(x). \tag{87}$$

Note that since $p = 2$, the polynomial $\omega_{\mathbf{g}}^{(1)}(x)$ has only even powers of x. Hence

$$\omega_{\mathbf{g}}^{(1)}(x) = \varphi_{\mathbf{g}}^{2}(x) \tag{88}$$

for some polynomial $\varphi_{\mathbf{g}}(x)$.

Note that $\omega_{\mathbf{g}}(x)$ and $G(x)$ have no common roots in any extension field.

Hence, by Eq. (85) we get

$$\varphi_{\mathbf{g}}^{2}(x) \equiv 0 \bmod G(x). \tag{89}$$

This means that

$$G(x) \mid \varphi_{\mathbf{g}}^{2}(x). \tag{90}$$

Assume from now on, that the Goppa polynomial $G(x)$ has no multiple roots in any extension field. Then, it follows from Eq. (90) that $G(x) \mid \varphi_{\mathbf{g}}^{2}(x)$ if and only if

$$G(x)^2 \mid \varphi_{\mathbf{g}}^{2}(x). \tag{91}$$

Thus, the parameters of binary Goppa codes are

$$\begin{array}{ll} \text{Code length} & n \leq q+1 = p^m + 1; \\ \text{Dimension} & k_G = n - r_G \geq n - rm;\ r = \deg G(x); \\ \text{Code distance} & d_G \geq d = 2\deg G(x) + 1, \end{array} \tag{92}$$

i.e. code distance is about twice more in comparison with the general alternant code for which the parity check matrix $\mathbf{H}$ has the same number of rows.

To obtain a generator matrix of a Goppa code, we can apply the mapping $\mathcal{B}$ (see Eqs. (69)-(71)) to the parity check matrix $\mathbf{H} = [\alpha_j^i / G(\alpha_j)], i = 0, 1, \ldots, t-1; j = 1, 2, \ldots, n$. Denote by $\mathbf{H}_{bin}$ the binary matrix obtained from the binary matrix $\widetilde{\mathbf{H}} = [\mathcal{B}\left(\alpha_j^i / G(\alpha_j)\right)]$ by deleting linearly dependent rows. Then a *generator* matrix $\mathbf{G}$ is a binary $k \times n$ matrix with maximal value of k such that $\mathbf{G}\mathbf{H}_{bin}^t = 0$.

Fast decoding of *Goppa codes* may be carried out just as for GRS codes.

Hence, *Goppa codes* possess *Fast decoding* algorithms.

Notice that in order to be able to carry out a decoding algorithm, we should know the ordering $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ and the Goppa polynomial $G(x)$.

2.4. Maximal Rank Distance Codes.

Rank Distance. Let the field $GF(q^N)$ be given and let $GF(q)$ be the base field, q is a power of a prime. Let

$$\omega_1, \omega_2, \ldots, \omega_N$$

be a basis of the field $GF(q^N)$ over the field $GF(q)$. Each element $x_j \in GF(q)$ can be uniquely represented in the form

$$x_j = a_{1,j}\omega_1 + a_{2,j}\omega_2 + \ldots + a_{N,j}\omega_N,$$

where $a_{i,j} \in GF(q)$.

Let A_N^n be the set of the $N \times n$ matrices with entries in $GF(q)$.

Let $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in GF(q^N)^n$ be a vector with coordinates in $GF(q^N)$.

Consider the bijective mapping

$$\mathcal{A}: \; GF(q)^n \to A_N^n \tag{93}$$

that maps the vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ onto the matrix

$$\mathbf{A}(\mathbf{x}) = \begin{bmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{N,1} & a_{N,2} & \ldots & a_{N,n} \end{bmatrix}.$$

Denote the rank of a matrix $\mathbf{A}$ over the field $GF(q)$ by $r(\mathbf{A}|q)$. Evidently, the rank depends on the field. In particular,

$$r(\mathbf{A}|q) \geq r(\mathbf{A}|q^N).$$

The *rank* (or *rank weight*) $r(\mathbf{x}|q)$ of a vector $\mathbf{x}$ is defined as the rank of $\mathbf{A}(\mathbf{x})$ over $GF(q)$:

$$r(\mathbf{x}|q) = r(\mathbf{A}(x)|q).$$

In fact, the rank function is a *norm* on $GF(q^N)^n$ because

1. for any $\mathbf{x}$, $r(\mathbf{x}|q) \geq 0$;
2. $r(\mathbf{x}|q) = 0 \Longleftrightarrow \mathbf{x} = 0$;
3. for any $\mathbf{x}, \mathbf{y} \in GF(q^N)^n$ we have, $r(\mathbf{x} + \mathbf{y}|q) \leq r(\mathbf{x}|q) + r(\mathbf{y}|q)$.

This allows us to define the *Rank distance function*

$$d_r(\mathbf{x}, \mathbf{y}) = r(\mathbf{x} - \mathbf{y}|q).$$

Equivalently, we can define the rank distance as follows.

The *rank weight* $r(\mathbf{x}|q)$ of $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in GF(q^N)^n$ is the *maximal* number of x_i that are linearly independent over the *base* field $GF(q)$.

The *rank distance* $d_r(\mathbf{x}, \mathbf{y})$ between $\mathbf{x}$ and $\mathbf{y}$ is the rank weight of the difference $\mathbf{x} - \mathbf{y}$.

It is clear that, for any $\mathbf{x}$,

$$r(\mathbf{x}|q) \leq \min(n, N). \tag{94}$$

The *rank distance* $d(\mathcal{C}) = d$ of a *linear* code $\mathcal{C}$ is defined by

$$d := \min\{d_r(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \in \mathcal{C},\ \mathbf{y} \in \mathcal{C},\ \mathbf{x} \neq \mathbf{y}\}, \tag{95}$$

or, equivalently,

$$d := \min\{r(\mathbf{x}|q) \mid \mathbf{x} \in \mathcal{C},\ \ \mathbf{x} \neq \mathbf{o}\}. \tag{96}$$

If a code $\mathcal{C}$ has distance d, then it can correct all errors $\mathbf{e}$ with $r(\mathbf{e}|q) \leq t = \lfloor (d-1)/2 \rfloor$.

Let a *linear* code $\mathcal{C} \subset GF(q^N)^n$ of *length* n, *dimension* k (or, equivalently, of *size* $M = q^{Nk}$), and (rank) distance d over $GF(q^N)$ be given. We construct a new code $\mathcal{C}^{tr} \subset GF(q^n)^N$, called the *transposed* code, of *length* N, of the *same* size $M = q^{Nk}$, and the same (rank) distance d over $GF(q^n)$. The construction is as follows.

Using the mapping (93), represent each codeword $\mathbf{x} \in \mathcal{C} \subset GF(q^N)^n$ as the corresponding matrix

$$\mathbf{A}(\mathbf{x}) = \begin{bmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{N,1} & a_{N,2} & \ldots & a_{N,n} \end{bmatrix}.$$

Transpose this matrix $\mathbf{A}(\mathbf{x})$:

$$\mathbf{A}(\mathbf{x})^t = \begin{bmatrix} a_{1,1} & a_{2,1} & \ldots & a_{N,1} \\ a_{1,2} & a_{2,2} & \ldots & a_{N,2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1,n} & a_{2,n} & \ldots & a_{N,n} \end{bmatrix}. \tag{97}$$

Choose a basis $\gamma_1, \gamma_2, \ldots, \gamma_n$ of the field $GF(q^n)$ and map the matrix $\mathbf{A}(\mathbf{x})^t$ onto the vector

$$\mathbf{y} = (y_1, y_2, \ldots, y_N) \in GF(q^n)^N,$$

where $y_i = a_{i,1}\gamma_1 + a_{i,2}\gamma_2 + \ldots a_{i,n}\gamma_n \in GF(q^n)$, $i = 1, 2, \ldots, N$.

All such vectors form a code $\mathcal{C}^{tr} \subset GF(q^n)^N$ of length N. It is also clear that it has the same number of codewords $M = q^{Nk}$ as $\mathcal{C}$ and the same rank distance d because the matrices $\mathbf{A}(x)$ and $\mathbf{A}(x)^t$ have the same rank.

Remark 1. *The code $\mathcal{C}^{tr}$ is a group code but not necessarily a linear code.*

Remark 2. *If $\mathcal{C}$ is an optimal code in the sense that its size is maximal for the given d, then $\mathcal{C}^{tr}$ is optimal as well.*

Lemma 1. *For any linear (n, k, d) code $\mathcal{C} \subset GF(q^N)^n$, we have the inequality*

$$d \leq \min(N, n) .$$

Proof. *See Eq (94).* □

Lemma 2. **(The Singleton-style bound.)** *Let $n \leq N$. For any linear (n, k, d) code $\mathcal{C} \subset GF(q^N)^n$,*

$$k \leq n - d + 1. \tag{98}$$

Proof. It is evident that, for any $\mathbf{x} \in GF(q^N)^n$, $r(\mathbf{x}|q) \leq w_H(\mathbf{x})$, where $w_H(\mathbf{x})$ denotes the Hamming weight of $\mathbf{x}$. Hence, any code of size M with the rank distance d is also a code of the same size with the Hamming distance $d^* \geq d$. This means that the size $M_r(n, d) = q^{Nk}$ of an optimal code for the rank metric is less then or equal to the size $M_H(n, d) = q^{Nk_1}$ of an optimal code for the Hamming metric with the same n and d. Thus,

$$k \leq k_1 \leq n - d + 1 \tag{99}$$

because the Singleton bound is valid for the Hamming distance. □

Lemma 3. *Let $n > N$. For any group code $\mathcal{G} \subset GF(q^N)^n$ of size $M = q^{Nk}$ and distance d,*

$$d \leq N,$$

$$Nk \leq n(N - d + 1) .$$

Proof. The first statement follows from Lemma 1. To obtain the second statement, construct a linear (N, k, d) code over the field $GF(q)$, transpose it and apply the Singleton-style bound. □

A linear (n, k, d) code is called a *maximal rank distance* (MRD) code if the Singleton bound (98) is reached. Such a code is optimal.

The theory of MRD codes is given in [11]. It is shown that, for any $n \leq N$, $1 \leq d \leq n$, an MRD code exists. Combining this with the transposed code construction, we obtain optimal rank codes for any length n and any admissible rank distance d.

General construction of optimal codes. Let $\mathcal{C} \subset GF(q)^n$ be a linear (n, k, d) code and let $\mathbf{H}$ be an $(n-k) \times n$ parity check matrix. For any $d-1$, let $\mathcal{Y}_{d-1}$ be the set of $(d-1) \times n$ matrices over the base field $GF(q)$ of full rank $d-1$.

Theorem 1. *Let $\mathbf{H}$ be an $(n-k) \times n$ parity check matrix of a linear (n, k) code $\mathcal{C} \subset GF(q^N)^n$. The code $\mathcal{C}$ has the rank distance d if and only if, for any matrix $\mathbf{Y} \in \mathcal{Y}_{d-1}$, we have*

$$r(\mathbf{Y}\mathbf{H}^t | q^N) = r(\mathbf{Y}|q) = d-1, \tag{100}$$

and there exists a matrix $\mathbf{Y}_0 \in \mathcal{Y}_d$ such that

$$r(\mathbf{Y}_0\mathbf{H}^t | q^N) < r(\mathbf{Y}_0|q) = d. \tag{101}$$

Proof. Necessity. Let $\mathcal{C}$ be a code of distance d. A vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ of rank $d-1$ or less can always be represented in the form

$$\mathbf{x} = (x_1, x_2, \ldots, x_n) = (u_1, u_2, \ldots, u_{d-1})\,\mathbf{Y},$$
$$u_i \in GF(q^N), \;\; \mathbf{Y} \in \mathcal{Y}_{d-1}.$$

Since $\mathbf{x}$ is *not* a codeword, the linear system

$$\mathbf{x}\mathbf{H}^t = (u_1, u_2, \ldots, u_{d-1})\,\mathbf{Y}\mathbf{H}^t = 0 \tag{102}$$

of equations in the unknowns $u_1, u_2, \ldots, u_{d-1}$ should have only the trivial zero solution. This will be the case if $r(\mathbf{Y}\mathbf{H}^t|q^N) = d-1$. On the other hand, a codeword $\mathbf{g}$ of rank d can be represented in the form

$$\mathbf{g} = (g_1, g_2, \ldots, g_n) = (v_1, v_2, \ldots, v_d)\mathbf{Y}_0, \tag{103}$$

where $\mathbf{Y}_0 \in \mathcal{Y}_d$ and $v_1, v_2, \ldots, v_d$ are linearly independent over $GF(q)$. Since $\mathbf{g}$ is a non-trivial solution of the linear system

$$\mathbf{g}\mathbf{H}^t = (v_1, v_2, \ldots, v_d)\,\mathbf{Y}_0\mathbf{H}^t = 0,$$

in the unknowns $v_1, v_2, \ldots, v_d$, we infer that $r(\mathbf{Y}_0\mathbf{H}^t|q^N) < d$.

Sufficiency. Suppose that the conditions (100)-(101) hold. Then the system (102) has only the trivial solution. Thus, vectors $\mathbf{x}$ of rank less or equal to $d-1$ can not be codewords. On the other hand, the system (103) has a solution of rank d. Hence, the distance of the code with the parity matrix $\mathbf{H}$ is d. □

We now present the general construction of an MRD code in terms of the parity check matrix.

Let $h_1, h_2, \ldots, h_n$ be a set of elements of $GF(g^N)$ which are *linearly independent* over $GF(q)$. Let us define the $(d-1) \times n$ matrix $\mathbf{H}$ by

$$\mathbf{H} = \begin{bmatrix} h_1 & h_2 & \cdots & h_n \\ h_1^q & h_2^q & \cdots & h_n^q \\ h_1^{q^2} & h_2^{q^2} & \cdots & h_n^{q^2} \\ \cdots & \cdots & \cdots & \cdots \\ h_1^{q^{d-2}} & h_2^{q^{d-2}} & \cdots & h_n^{q^{d-2}} \end{bmatrix}. \tag{104}$$

Theorem 2. *The code $\mathcal{C}$ defined by the parity check matrix* $\mathbf{H}$ *in (104) is an MRD code with the following parameters:*

- code length $n \leq N$;
- dimension $k = n - d + 1$;
- (Rank and Hamming) code distance $d = r + 1 = n - k + 1$.

Proof. Let $\mathbf{Y} \in \mathcal{Y}_{d-1}$. Then the square $(d-1) \times (d-1)$ matrix $\mathbf{YH}^t$ can be represented in the form

$$\mathbf{YH}^t = \begin{bmatrix} f_1 & f_1^q & \cdots & f_1^{q^{d-2}} \\ f_2 & f_2^q & \cdots & f_2^{q^{d-2}} \\ \vdots & \vdots & \vdots & \vdots \\ f_{d-1} & f_{d-1}^q & \cdots & f_{d-1}^{q^{d-2}} \end{bmatrix}, \tag{105}$$

where $(f_1, f_2, \ldots, f_{d-1})^t = \mathbf{Y}(h_1, h_2, \ldots, h_n)^t$. The elements $f_1, f_2, \ldots, f_{d-1} \in GF(q^N)$ are linearly independent over $GF(q)$ because if not, the elements $h_1, h_2, \ldots, h_n$ would also be linearly dependent over $GF(q)$, in contradiction with the assumption. It is known (see, for instance, [17]) that the matrix $\mathbf{YH}^t$ is non-singular. This means that $r(\mathbf{YH}^t \mid q^N) = d - 1$. Hence, by Theorem (1), a code $\mathcal{C}$ has rank distance d. The dimension of the code is $k = n - d + 1$. So this code reaches the Singleton bound for the rank distance as well as, by (99), for the Hamming distance. $\square$

The general construction of an MRD code can be also given in terms of its generator matrix. Let $g_1, g_2, \ldots, g_n$ be any set of elements of $GF(p^N)$ which are linearly independent over $GF(p)$. A matrix $\mathbf{G}$ is defined by

$$\mathbf{G} = \begin{bmatrix} g_1 & g_2 & \cdots & g_n \\ g_1^p & g_2^p & \cdots & g_n^p \\ g_1^{p^2} & g_2^{p^2} & \cdots & g_n^{p^2} \\ \cdots & \cdots & \cdots & \cdots \\ g_1^{p^{k-1}} & g_2^{p^{k-1}} & \cdots & g_n^{p^{k-1}} \end{bmatrix}. \tag{106}$$

It can be shown that there exists an orthogonal $(d-1) \times n$ matrix $\mathbf{H}$ of the form (104) such that

$$\mathbf{GH}^t = 0,$$

where $d = n - k + 1$. Hence the matrix (106) is a generator matrix of an MRD code.

MRD codes possess *Fast Decoding Algorithms* (see [11, 23].) We need some results of the theory of linearized polynomials to present these algorithms.

Linearized Polynomials. Let the field $GF(q^N)$ be given and let $GF(q)$ be the base field, where q is a power of a prime.

A *linearized polynomial* with coefficients in the field $GF(q^N)$ is a polynomial of the form

$$F(x) = \sum_{i=0}^{n} f_i x^{q^i}. \tag{107}$$

The largest i such that $f_i \neq 0$ will be called the *norm* $N(F)$ of the polynomial. By way of convention, the norm of the linearized polynomial 0 is taken to be $-\infty$. If $F(x) \neq 0,\ G(x) \neq 0$, then

$$N\left(F \otimes G\right) \geq \max\left(N\left(F\right),\ N\left(G\right)\right).$$

We write $R_N[x]$ to denote the set of all the linearized polynomials with coefficients in $GF(q^N)$.

Addition in $R_N[x]$ is defined by

$$F(x) + G(x) = \sum_{i=0}^{n} \left(f_i + g_i\right) x^{q^i}, \tag{108}$$

and *symbolic multiplication* by

$$F(x) \otimes G(x) = F\left(G(x)\right) = \sum_{i=0}^{n} \sum_{k+s=i} \left(f_s g_k^{q^s}\right) x^{q^i}. \tag{109}$$

In other words, symbolic multiplication is the composition (or the substitution) of two polynomials. It is important to note that symbolic multiplication is *not commutative*. But it is *associative* and *distributive*:

$$F(x) \otimes \left(G(x) \otimes H(x)\right) = \left(F(x) \otimes G(x)\right) \otimes H(x) = F(x) \otimes G(x) \otimes H(x),$$

$$\left(F(x) + G(x)\right) \otimes H(x) = F(x) \otimes H(x) + G(x) \otimes H(x),$$

$$F(x) \otimes \left(G(x) + H(x)\right) = F(x) \otimes G(x) + F(x) \otimes H(x). \tag{110}$$

Hence, the set $R_N[x]$ under these two operations becomes a *non-commutative* ring whose multiplicative identity is the polynomial x.

The Euclidean Right Division Algorithm. Consider two linearized polynomials $f(x)$ and $g(x)$ over $GF(q^N)$. If

$$\begin{aligned} f(x) &= Q(x) \otimes F(x), \\ g(x) &= P(x) \otimes F(x), \end{aligned}$$

we say that $F(x)$ is a *right* common divisor of polynomials $f(x)$ and $g(x)$. Left common divisors are defined in a similar manner. In $R_N[x]$, we have both Euclidean left division algorithms and right division algorithms. We describe the right division algorithm. The left division algorithm is similar.

The right Euclidean division algorithm is a procedure for finding the *right* greatest common divisor of two polynomials. We use the notation rgcd = the *right* greatest common divisor. Consider again two linearized polynomials $f(x)$ and $g(x)$ over $GF(q)$. Suppose that

$$N(g(x)) \leq N(f(x)).$$

For convenience, define $F_{-1}(x) := f(x)$, $F_0(x) := g(x)$.

1st step Divide to the right $F_{-1}(x)$ $(= f(x))$ by $F_0(x)$ $(= g(x))$:

$$F_{-1}(x) = G_1(x) \otimes F_0(x) + F_1(x), \text{ where } N(F_1) < N(F_0). \tag{111}$$

2nd step Divide $F_0(x)$ $(= g(x))$ by $F_1(x)$:

$$F_0(x) = G_2(x) \otimes F_1(x) + F_2(x), \text{ where } N(F_2) < N(F_1). \tag{112}$$

3rd step Divide $F_1(x)$ by $F_2(x)$:

$$F_1(x) = G_3(x) \otimes F_2(x) + F_3(x), \text{ where } N(F_3) < N(F_2). \tag{113}$$

i'th step Divide $F_{i-2}(x)$ by $F_{i-1}(x)$:

$$F_{i-2}(x) = G_i(x) \otimes F_{i-1}(x) + F_i(x), \text{ where } N(F_i) < N(F_{i-1}). \tag{114}$$

next to the last step Divide $F_{p-2}(x)$ by $F_{p-1}(x)$:

$$F_{p-2}(x) = G_p(x) \otimes F_{p-1}(x) + F_p(x), \text{ where } N(F_p) < N(F_{p-1}). \tag{115}$$

last step Divide $F_{p-1}(x)$ by $F_p(x)$:

$$F_{p-1}(x) = G_{p+1}(x) \otimes F_p(x). \tag{116}$$

Then

$$\operatorname{rgcd}(f(x),\ g(x)) = F_p(x). \tag{117}$$

If $F_p(x) = x$, then $f(x)$ and $g(x)$ are relatively prime.

At each step of the procedure, the current remainder $F_i(x)$ can be represented as a linear combination of the two previous remainders. Thus, it is possible to represent all remainders, including the last one $F_p(x)$, as linear combinations of $f(x)$ $(= F_{-1}(x))$ and $g(x)$ $(= F_0(x))$:

$$F_i(x) = (-1)^{i-1}(A_i(x) \otimes f(x) - B_i(x) \otimes g(x)). \tag{118}$$

The polynomials $A_i(x)$ and $B_i(x)$ can be obtained from Eqs. (49)-(54) but there exists an inductive way of calculating these polynomials. Define two sequences of polynomials $\{A_i(x)\}$ and $\{B_i(x)\}$ by

$$A_i(x) = G_i(x) \otimes A_{i-1}(x) + A_{i-2}(x), \tag{119}$$

where, by definition, $A_{-1}(x) = x$ and $A_0(x) = 0$, and

$$B_i(x) = G_i(x) \otimes B_{i-1}(x) + B_{i-2}(x), \tag{120}$$

where, by definition, $B_{-1}(x) = 0$ and $B_0(x) = x$. Then Eq. (118) is true.

It is easy to obtain many relations concerning the above polynomials. Define two sequences of polynomials $\{U_i(x)\}$ and $\{V_i(x)\}$ by

$$U_i(x) = U_{i-1}(x) \otimes G_i(x) + U_{i-2}(x), \tag{121}$$

where, by definition, $U_{-1}(x) = 0$ and $U_0(x) = x$, and

$$V_i(x) = V_{i-1}(x) \otimes G_i(x) + V_{i-2}(x), \tag{122}$$

where, by definition, $V_{-1}(x) = x$ and $V_0(x) = 0$.

The following properties are important for the decoding procedure:

1. For any i,
$$\begin{aligned} \mathrm{rgcd}(A_i(x), B_i(x)) = x, \\ \mathrm{rgcd}(U_i(x), V_i(x)) = x, \end{aligned} \tag{123}$$
i.e. both the polynomial pairs $\{A_i(x), B_i(x)\}$ and $\{U_i(x), V_i(x)\}$ are relatively prime.

2. For any i,
$$\begin{aligned} F_{-1}(x) &= U_{i+1}(x) \otimes F_i(x) + U_i(x) \otimes F_{i+1}(x), \\ F_0(x) &= V_{i+1}(x) \otimes F_i(x) + V_i(x) \otimes F_{i+1}(x). \end{aligned} \tag{124}$$

3. For any i,
$$F_i(x) = (-1)^{i-1} \left(A_i(x) \otimes F_{-1}(x) - B_i(x) \otimes F_0(x) \right). \tag{125}$$
In particular,
$$\mathrm{rgcd}(f(x), g(x)) = F_p(x) = (-1)^{p-1} \left(A_p(x) F_{-1}(x) - B_p(x) F_0(x) \right). \tag{126}$$

4. For any i,
$$\begin{aligned} A_i(x) \otimes U_{i+1}(x) - B_i(x) \otimes V_{i+1}(x) = (-1)^{i-1} x, \\ A_i(x) \otimes U_i(x) - B_i(x) \otimes V_i(x) = 0, \\ A_i(x) \otimes U_{i-1}(x) - B_i(x) \otimes V_{i-1}(x) = (-1)^{i-1} x. \end{aligned} \tag{127}$$

5. For any i,

$$N(A_i(x)) = \sum_{s=2}^{i} N(G_s(x)), \tag{128}$$

$$N(B_i(x)) = \sum_{s=1}^{i} N(G_s(x)) = N(f(x)) - N(F_{i-1}(x)), \tag{129}$$

$$N(F_i(x)0 = N(f(x)) - \sum_{s=1}^{i+1} N(G_s(x)). \tag{130}$$

Fast decoding of rank codes. We consider decoding algorithms for an MRD code given by the parity check matrix $\mathbf{H}$ (104). Let $\mathbf{g} = (g_1, g_2, \ldots, g_n)$ be a code vector and let $\mathbf{y} = \mathbf{g} + \mathbf{e}$ be the received vector, where $\mathbf{e} = (e_1, e_2, \ldots, e_n)$ is an error vector. Let us calculate the syndrome vector $\mathbf{s} = \mathbf{y}\mathbf{H}^t$:

$$\mathbf{y}\mathbf{H}^t = (\mathbf{g} + \mathbf{e})\mathbf{H}^t = \mathbf{g}\mathbf{H}^t + \mathbf{e}\mathbf{H}^t = \mathbf{e}\mathbf{H}^t = \mathbf{s}, \tag{131}$$

or

$$(e_1, e_2, \ldots, e_n)\mathbf{H}^t = \mathbf{s} = (s_0, s_1, \ldots, s_{d-2}). \tag{132}$$

The problem of decoding is to find the error vector $\mathbf{e} = (e_1, e_2, \ldots, e_n)$ if the matrix $\mathbf{H}$ and the syndrome vector $\mathbf{s} = (s_0, s_1, \ldots, s_{d-2})$ are given.

Assume that the rank weight of the error vector is equal to $r(\mathbf{e}|q) = m$. Then the vector $\mathbf{e}$ can be represented in the form

$$\mathbf{e} = \mathbf{E}\mathbf{Y} = (E_1, E_2, \ldots, E_m)\,\mathbf{Y}, \tag{133}$$

where $E_1, E_2, \ldots, E_m \in GF(q^N)$ are linearly independent over $GF(q)$, and $\mathbf{Y} \in \mathcal{Y}_m$ is an $m \times n$ matrix of rank exactly m with all the entries in the base field $GF(q)$. Hence, Eq (132) can be rewritten as

$$(E_1, E_2, \ldots, E_m)\,\mathbf{Y}\mathbf{H}^t = \mathbf{s}. \tag{134}$$

The matrix $\mathbf{Z}^t := \mathbf{Y}\mathbf{H}^t$ has the form

$$\mathbf{Z}^t = \mathbf{Y}\mathbf{H}^t = \begin{bmatrix} z_1 & z_1^q & \ldots & z_1^{q^{d-2}} \\ z_2 & z_2^q & \ldots & z_2^{q^{d-2}} \\ \vdots & \vdots & \vdots & \vdots \\ z_m & z_m^q & \ldots & z_m^{q^{d-2}} \end{bmatrix}, \tag{135}$$

where $\mathbf{z}^t := (z_1, z_2, \ldots, z_m)^t = \mathbf{Y}(h_1, h_2, \ldots, h_n)^t$. If $E_1, E_2, \ldots, E_m$ are linearly independent over $GF(q)$, then the elements $z_1, z_2, \ldots, z_m \in GF(q^N)$ are also linearly independent over $GF(q)$. From Eq. (134), we have the following system of $d-1$ equations in the $2m$ unknowns $E_1, E_2, \dots, E_m$, $z_1, z_2, \dots, z_m$:

$$(E_1, E_2, \dots, E_m) \begin{bmatrix} z_1 & z_1^q & \dots & z_1^{q^{d-2}} \\ z_2 & z_2^q & \dots & z_2^{q^{d-2}} \\ \vdots & \vdots & \vdots & \vdots \\ z_m & z_m^q & \dots & z_m^{q^{d-2}} \end{bmatrix} = (s_0, s_1, \dots, s_{d-2}), \tag{136}$$

or,

$$\sum_{j=1}^{m} E_j z_j^{q^i} = s_i, \; i = 0, 1, \dots, d-2. \tag{137}$$

Note that the integer m is also unknown.

If we, somehow, have obtained a solution of Eq (137), we can find the matrix $\mathbf{Y}$ from Eq. (135) and the error vector $\mathbf{e}$ from Eq (133). For a given m, there exist many solutions of Eq. (137). If $(E_1, E_2, \dots, E_m)$ and $(z_1, z_2, \dots, z_m)^t$ is a solution, then $(E_1, E_2, \dots, E_m)\mathbf{Q}$ and $\mathbf{Q}^{-1}(z_1, z_2, \dots, z_m)^t$ also is a solution for all $m \times m$ non-singular matrices $\mathbf{Q}$ with entries in $GF(q)$. Moreover, there are no other solutions. All these solutions are equivalent in that sense that the recovered error vector $\mathbf{e}$ is the same. So, we have to obtain any solution of Eq (137).

For the case $m \le t = \left\lfloor \frac{d-1}{2} \right\rfloor$, we present two algorithms of the fast solution of Eq (137): the *matrix algorithm* and the algorithm based on the *right Euclidean division algorithm.*

The Matrix Algorithm. The idea of this algorithm is similar to the Peterson algorithm for the decoding of alternant codes.

For any $i = 1, 2, \dots, t$, define the $i \times i$ matrix

$$\mathbf{M}_i = \begin{bmatrix} s_0 & s_1^{q^{-1}} & \dots & s_{i-1}^{q^{-i+1}} \\ s_1 & s_2^{q^{-1}} & \dots & s_i^{q^{-i+1}} \\ \vdots & \vdots & \vdots & \vdots \\ s_{i-1} & s_i^{q^{-1}} & \dots & s_{2i-2}^{q^{-i+1}} \end{bmatrix} \tag{138}$$

whose entries are obtained from the known syndrome vector $\mathbf{s}$.

Lemma 4. *Let $m = r(\mathbf{e}|q) \le t$ be the rank weight of the error vector $\mathbf{e}$. If $i > m$, then*

$$\det(\mathbf{M}_i) = 0.$$

If $i = m$, then

$$\det(\mathbf{M}_i) \neq 0.$$

Proof. Note that even for $i > m$, we can represent the error vector $\mathbf{e}$ in a form similar to Eq (133):

$$\mathbf{e} = (\tilde{E}_1, \tilde{E}_2, \dots, \tilde{E}_i)\mathbf{Y}, \;\; \mathbf{Y} \in \mathcal{Y}_m,$$

but in this case the elements $\widetilde{E}_1, \widetilde{E}_2, \ldots, \widetilde{E}_i$ are *linearly dependent* over $GF(q)$. Let us express the syndrome components s_p in terms of $\widetilde{E}_1, \widetilde{E}_2, \ldots, \widetilde{E}_i$ and the corresponding $\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_i$:

$$s_p = \sum_{j=1}^{i} \widetilde{E}_j \tilde{z}_j^{q^p}, \; p = 0, 1, \ldots, d-2.$$

If we replace the s's by these expressions in Eq (138), we obtain after some manipulations

$$\mathbf{M}_i = \begin{bmatrix} \tilde{z}_1 & \tilde{z}_2 & \ldots & \tilde{z}_i \\ \tilde{z}_1^q & \tilde{z}_2^q & \ldots & \tilde{z}_i^q \\ \tilde{z}_1^{q^2} & \tilde{z}_2^{q^2} & \ldots & \tilde{z}_i^{q^2} \\ \ldots & \ldots & \ldots & \ldots \\ \tilde{z}_1^{q^{i-1}} & \tilde{z}_2^{q^{i-1}} & \ldots & \tilde{z}_i^{q^{i-1}} \end{bmatrix} \begin{bmatrix} \widetilde{E}_1 & \widetilde{E}_1^{q^{-1}} & \ldots & \widetilde{E}_1^{q^{-i+1}} \\ \widetilde{E}_2 & \widetilde{E}_2^{q^{-1}} & \ldots & \widetilde{E}_2^{q^{-i+1}} \\ \vdots & \vdots & \vdots & \vdots \\ \widetilde{E}_i & \widetilde{E}_i^{q^{-1}} & \ldots & \widetilde{E}_i^{q^{-i+1}} \end{bmatrix}.$$

Hence, $\det(\mathbf{M}_i) = 0$ because $\widetilde{E}_1, \widetilde{E}_2, \ldots, \widetilde{E}_i$ (and also $\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_i$) are *linearly dependent* over $GF(q)$.

On the other hand, if $i = m$, then $\widetilde{E} = E$ and $\tilde{z} = z$ in the above representation. Hence, $\det(\mathbf{M}_m) \neq 0$ because $E_1, E_2, \ldots, E_m$ (and also $z_1, z_2, \ldots, z_m$) are *linearly independent* over $GF(q)$. □

This lemma allows us to find the rank weight m of the error vector $\mathbf{e}$:

1. Calculate the syndrome $\mathbf{s}$ by Eq. (131).

2. For $i = t, t-1, \ldots$, calculate the matrices $\mathbf{M}_i$ and their determinant $\det \mathbf{M}_i$ till the first value $i = m$ for which $\det(\mathbf{M}_i) \neq 0$. This value is the rank weight of $\mathbf{e}$.

From now on, we have to solve Eq (137) for the known m.

Since the unknowns $z_1, z_2, \ldots, z_m$ are linearly independent over $GF(q)$, introduce the linearized polynomial

$$\sigma(x) = \sum_{i=0}^{m} \sigma_i x^{q^i}, \; \sigma_m = 1 \tag{139}$$

having as roots all the linear combinations of $z_1, z_2, \ldots, z_m$ with coefficients in the base field $GF(q)$.

Theorem 3.

$$(\sigma_0, \sigma_1, , \sigma_{m-1})\, \mathbf{M}_m = -\left(s_m, s_{m+1}^{q^{-1}}, \ldots, s_{2m-1}^{q^{-m+1}}\right). \tag{140}$$

Proof. Represent $\mathbf{M}_m$ in the form

$$\mathbf{M}_m = \begin{bmatrix} z_1 & z_2 & \dots & z_m \\ z_1^q & z_2^q & \dots & z_m^q \\ z_1^{q^2} & z_2^{q^2} & \dots & z_m^{q^2} \\ \dots & \dots & \dots & \dots \\ z_1^{q^{m-1}} & z_2^{q^{m-1}} & \dots & z_m^{q^{m-1}} \end{bmatrix} \begin{bmatrix} E_1 & E_1^{q^{-1}} & \dots & E_1^{q^{-m+1}} \\ E_2 & E_2^{q^{-1}} & \dots & E_2^{q^{-m+1}} \\ \vdots & \vdots & \vdots & \vdots \\ E_m & E_m^{q^{-1}} & \dots & E_m^{q^{-m+1}} \end{bmatrix}.$$

Since, by the definition of $\sigma(x)$,

$$\sigma_0 z_j + \sigma_1 z_j^q + \dots \sigma_{m-1} z_j^{q^{m-1}} = -z_j^{q^m}, \\ j = 1, 2, \dots, m,$$

we have

$$(\sigma_0, \sigma_1, , \sigma_{m-1})\, \mathbf{M}_m = -(z_1^{q^m}, z_2^{q^m}, \dots, z_m^{q^m}) \begin{bmatrix} E_1 & E_1^{q^{-1}} & \dots & E_1^{q^{-m+1}} \\ E_2 & E_2^{q^{-1}} & \dots & E_2^{q^{-m+1}} \\ \vdots & \vdots & \vdots & \vdots \\ E_m & E_m^{q^{-1}} & \dots & E_m^{q^{-m+1}} \end{bmatrix} =$$

$$\left(\sum_{s=1}^{m} E_s z_s^{q^m}, \left(\sum_{s=1}^{m} E_s z_s^{q^{m+1}} \right)^{q^{-1}}, \dots, \left(\sum_{s=1}^{m} E_s z_s^{q^{2m-1}} \right)^{q^{-m+1}} \right) =$$

$$-\left(s_m, s_{m+1}^{q^{-1}}, \dots, s_{2m-1}^{q^{-m+1}} \right).$$

□

The linear system of equations (140) in the m unknowns $\sigma_0, \sigma_1, , \sigma_{m-1}$ has a unique solution because the matrix $\mathbf{M}_m$ is non-singular. Hence, we can proceed with the algorithm as follows.

3. Solve the linear system (140) and calculate $\sigma(x)$.

4. Calculate the linearly independent roots $z_1, z_2, \dots, z_m$ of $\sigma(x)$. This problem can be reduced to the problem of solving a few linear systems over the base field $GF(q)$ (see [22]).

5. Solve the linear systems of the first m equations of Eq. (137) in the m unknowns $E_1, E_2, \dots, E_m$.

6. Calculate the matrix $\mathbf{Y}$ using Eq. (135).

7. Calculate the error vector $\mathbf{e}$ by Eq. (133).

We have to calculate determinants and to solve linear systems. Hence, the number of calculations is $O(n^3)$.

Fast Decoding Algorithm Based on the Right Euclidean Division Algorithm. This algorithm is an analogue of the Berlekamp-Massey algorithm for decoding generalized Reed-Solomon codes.

Introduce the linearized *syndrome polynomial*

$$S(x) = \sum_{p=0}^{d-2} s_p x^{q^p}. \tag{141}$$

Denote by $E(x)$ the linearized polynomial whose roots are all the possible linear combinations of $E_1, E_2, \ldots, E_m$ with coefficients in the base field $GF(q)$:

$$E(x) = \sum_{p=0}^{m} \Delta_p x^{q^p}, \ \Delta_m = 1. \tag{142}$$

Introduce the auxiliary linearized polynomial $F(x)$ by

$$F(x) = \sum_{p=0}^{m-1} F_p x^{q^p}, \tag{143}$$

where

$$F_p = \sum_{i=0}^{p} \Delta_i s_{p-i}^{q^i}, \ p = 0, 1, \ldots, m-1. \tag{144}$$

Note that the norm of $F(x)$ is strictly less than the norm of $E(x)$:

$$N(F(x)) \leq N(E(x)) - 1.$$

Theorem 4. (The key equation)

$$F(x) = E(x) \otimes S(x) \ \bmod \ x^{q^{d-1}}, \tag{145}$$

$$\begin{aligned} N(E(x)) &\leq \tfrac{d-1}{2}, \\ N(F(x)) &< \tfrac{d-1}{2}. \end{aligned} \tag{146}$$

Proof.

$$E(x) \otimes S(x) = \sum_{p=0}^{m} \Delta_p^{q^p} S(x) = \sum_{p=0}^{m+d-2} x^{q^p} \left(\sum_{j+i=p}^{m} \Delta_i s_j^{q^i} \right).$$

For $0 \leq p \leq m-1$, the inner sum is just F_p:

$$\sum_{j+i=p}^{m} \Delta_i s_j^{q^i} = \sum_{i=0}^{p} \Delta_i s_{p-i}^{q^i} = F_p.$$

For $m \le p \le d-2$ we have

$$\sum_{j+i=p}^{m} \Delta_i s_j^{q^i} = \sum_{i=0}^{m} \Delta_i s_{p-i}^{q^i} = \sum_{i=0}^{p} \Delta_i \left(\sum_{s=0}^{m} E_s z_s^{q^{p-i}} \right)^{q^i} =$$

$$\sum_{s=0}^{m} z_s^{q^p} \left(\sum_{i=0}^{p} \Delta_i E_s^{q^i} \right) = \sum_{s=0}^{m} z_s^{q^p} E\left(E_s\right) = 0$$

because, by definition of the polynomial $E(x)$, $\sum_{s=0}^{m} z_s^{q^p} E\left(E_s\right) = 0.$

For $p \ge d-1$ the coefficients vanish mod $x^{q^{d-1}}$. This proves the theorem. □

Decoding means finding the polynomials $E(x)$ and $F(x)$ in Eq (145) with the restrictions (146), if polynomials $S(x)$ and $x^{q^{d-1}}$ are given.

For $m \le t = \left\lfloor \frac{d-1}{2} \right\rfloor$, the equation (145) always has a solution. The problem is to find such polynomials $E(x)$ and $F(x)$ that the norm $N\left(E(x)\right)$ is minimal.

It follows from Eq (145), that there exists a polynomial $C(x)$ such that

$$F(x) = C(x) \otimes x^{q^{d-1}} + E(x) \otimes S(x).$$

It follows from Eq (125), that for the ith remainder $F_i(x)$ we have

$$F_i(x) = (-1)^i B_i(x) \otimes F_0(x) \mod F_{-1}(x). \tag{147}$$

Hence, if we put $F_{-1}(x) = x^{q^{d-1}}$, $F_0(x) = S(x)$, we can apply the right Euclidean division algorithm (Eqs. (111)-(116)) to obtain $E(x)$ and $F(x)$.

E1. Start with calculating the polynomials $F_i(x)$, $A_i(x)$, $B_i(x)$, $U_i(x)$, $V_i(x)$, $i = 1$.

E2. (**The Stop rule.**) Continue the calculations till the index $i = m$ satisfies the inequalities

$$N(F_{m-1}(x)) \ge \tfrac{d-1}{2}$$
but
$$N(F_m(x)) < t = \left\lfloor \tfrac{d-1}{2} \right\rfloor .$$

E3. Calculate

$$\begin{aligned} E(x) &= \mu B_m(x), \\ F(x) &= (-1)^{m-1} \mu F_m(x), \end{aligned} \tag{148}$$

where the constant μ is chosen such that $\Delta_m = 1$. These polynomials give the solution of Eq. (145). Indeed, it follows from Eqs. (129), (130), (147), that

$$\begin{aligned} N(E(x)) = N(B_m(x)) &\le \tfrac{d-1}{2}, \\ N(F(x)) &< \tfrac{d-1}{2}. \end{aligned} \tag{149}$$

It can be shown that any other solution of the key equation (145) that satisfies the restrictions (146) can differ from the obtained solution (148) only by a constant factor.

E4. Calculate the roots $E_1, E_2, \dots, E_m$ of $E(x)$ which are linear independent over $GF(q)$.

E5. From the first m equations of the system (137), get the linear system

$$\sum_{j=1}^{m} E_j^{q^{-i}} z_j = s_i^{q^{-i}},\ i = 0, 1, \dots, m-1$$

in the m unknowns $z_1, z_2, \dots, z_m$ and solve it.

E6. Calculate the matrix $\mathbf{Y}$ using Eq (135).

E7. Calculate the error vector $\mathbf{e}$ by Eq (133).

The complexity of the described algorithm depends on the complexity of the right Euclidean division algorithm. There exist an algorithm wich computes the rgcd of two linearized polynomials of norm n in $O(n \log^2 n)$ steps. Hence, the number of calculations is $O(d \log^2 d + dn)$.

3. The Niederreiter Cryptosystem Based On Generalized Reed-Solomon Codes

This cryptosystem was proposed by Prof. H. Niederreiter in 1986 [8]. It was broken by Prof. V.M. Sidelnikov and Dr. S.O. Shestakov in 1992 [9]

A modification has been proposed to avoid the Sidelnikov-Shestakov attack.

3.1. Description.

Private keys

The legitimate user A chooses as private keys:

The parity check matrix of a generalized Reed-Solomon code (see Section 2.1).

$$\mathbf{H} = \begin{bmatrix} z_1 & z_2 & \dots & z_n \\ z_1\alpha_1 & z_2\alpha_2 & \dots & z_n\alpha_n \\ z_1\alpha_1^2 & z_2\alpha^2 & \dots & z_n\alpha_n^2 \\ \dots & \dots & \dots & \dots \\ z_1\alpha_1^{r-1} & z_2\alpha_2^{r-1} & \dots & z_n\alpha_n^{r-1} \end{bmatrix}, \qquad (150)$$

$$z_j,\ \alpha_j \in GF(q),\ z_j \neq 0,\ \alpha_j \text{ are different},$$
$$i = 0, 1, \dots, r-1;\ j = 1, 2, \dots, n.$$

(As a consequence) a fast decoding algorithm (see Section 2.2).

A non-singular scrambling square matrix $\mathbf{S}$ of order r:

$$\mathbf{S} = \begin{bmatrix} s_{01} & s_{02} & \dots & s_{0n} \\ s_{11} & s_{12} & \dots & s_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ s_{r-1,1} & s_{r-1,2} & \dots & s_{r-1,n} \end{bmatrix}.$$

This matrix is used to scramble the parity check matrix, i.e. to destroy any evident structure of the parity check matrix.

Public key

The legitimate user A calculates the product

$$\mathbf{H_{cr}} = \mathbf{SH} =$$
$$\begin{bmatrix} z_1F_0(\alpha_1) & z_2F_0(\alpha_2) & \dots & z_nF_0(\alpha_n) \\ z_1F_1(\alpha_1) & z_2F_1(\alpha_2) & \dots & z_nF_1(\alpha_n) \\ \vdots & \vdots & \vdots & \vdots \\ z_1F_{r-1}(\alpha_1) & z_2F_{r-1}(\alpha_2) & \dots & z_nF_{r-1}(\alpha_n) \end{bmatrix},$$

where

$$F_i(x) = \sum_{k=0}^{r-1} s_{ik}x^k, \ i = 0, 1, \dots, r-1$$

are the polynomials defined by the scrambling matrix $\mathbf{S}$.

The legitimate user A publishes the matrix $\mathbf{H_{cr}}$ as a *public key* in some directory, in the hope that it is very difficult to get the secret matrices $\mathbf{S}$ and $\mathbf{H}$ separately from this product.

Encryption

In this cryptosystem, all the possible plaintexts (messages) are n-vectors

$$\mathbf{m} = (m_1, m_2, \dots, m_n)$$

of (Hamming) weight

$$w_H(\mathbf{m}) \le t = \left\lfloor \frac{d-1}{2} \right\rfloor = \left\lfloor \frac{r}{2} \right\rfloor$$

with components in $GF(q)$.

Note that messages are *not* codewords of the chosen GRS code but patterns of '*errors*' that can be corrected by the decoding algorithm.

If a person B wants to send the secret message to A, he chooses a plaintext to send $\mathbf{m} = (m_1, m_2, \ldots, m_n)$ and calculates the *ciphertext* as the syndrom $\mathbf{c}$ of a code with the Parity check matrix $\mathbf{H_{cr}}$:

$$\mathbf{c} = \mathbf{m}\mathbf{H}_{\mathbf{cr}}^{\mathbf{t}} = \mathbf{m}\mathbf{H}^{\mathbf{t}}\mathbf{S}^{\mathbf{t}}. \tag{151}$$

Hence, in this cryptosystem, the set of ciphertexts is the set of all the possible syndroms of correctable errors.

Decryption

Upon receiving the ciphertext $\mathbf{c}$, the legitimate user A multiplies it by $\left(\mathbf{S}^{\mathbf{t}}\right)^{-1}$:

$$\mathbf{c}\left(\mathbf{S}^{\mathbf{t}}\right)^{-1} = \mathbf{m}\mathbf{H}^{\mathbf{t}}$$

and gets the syndrome of the plaintext $\mathbf{m}$. Then he applies a fast decoding algorithm to obtain the plaintext $\mathbf{m}$.

3.2. Breaking the Niederreiter Cryptosystem

The Sidelnikov-Shestakov attack. Recently, the *Niederreiter* PKC was broken by Sidelnikov and Shestakov. We give a slightly different version of their attack.

Recall that everybody knows the public key

$$\mathbf{H_{cr}} = \mathbf{SH} = \mathbf{S}[z_j\alpha_j^i]$$

but not $\mathbf{S}$, $\{z_j\}$, $\{\alpha_j\}$ separately. Given the public key $\mathbf{H_{cr}} = \mathbf{SH}$, the breaking party tries to find trapdoors $\mathbf{H_{tr}}$ and $\mathbf{S_{tr}}$ such that

$$\mathbf{H_{cr}} = \mathbf{SH} = \mathbf{H_{tr}}\mathbf{S_{tr}},$$

where

$$\mathbf{H_{tr}} = [y_j\beta_j^i].$$

The elements $\{y_j\}$ and $\{\beta_j\}$ may differ from the elements $\{z_j\}$ and $\{\alpha_j\}$. Nevertheless, they allow to decrypt any ciphertext.

Lemma 5. *A matrix* $\mathbf{H_{tr}}$ *is a trapdoor if and only if both* $\mathbf{H}$ *and* $\mathbf{H_{tr}}$ *are parity check matrices (possibly different) of the* **same** *GRS code.*

Proof. The proof is trivial. □

Evidently, all matrices $\mathbf{H_{cr}}$, $\mathbf{H}$ and $\mathbf{H_{tr}}$ can be reduced to the same row-reduced echelon form.

The breaking party calculates the *canonical row-reduced echelon form* $\mathbf{H_{syst}}$ (see eq. (18)) using the known parity check matrix $\mathbf{H_{cr}}$:

$$\mathbf{H_{syst}} = [\mathbf{E_r}\ \mathbf{R}] = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & \cdots & R_{1,r+1} & R_{1,r+2} & \cdots & R_{1,n} \\ 0 & 1 & 0 & \cdots & 0 & \cdots & R_{2,r+1} & R_{2,r+2} & \cdots & R_{2,n} \\ 0 & 0 & 1 & \cdots & 0 & \cdots & R_{3,r+1} & R_{3,r+2} & \cdots & R_{3,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 & \cdots & R_{r,r+1} & R_{r,r+2} & \cdots & R_{r,n} \end{bmatrix}. \tag{152}$$

Reducing to the systematic form requires $O(n^3)$ calculations.

From eq's. (19)-(22), we have

$$R_{i,j} = \frac{z_j}{z_i} \prod_{k=1,k\neq i}^{r} \frac{\alpha_j - \alpha_k}{\alpha_i - \alpha_k}, \qquad i = 1, 2, \ldots, r; \qquad j = r+1, r+2, \ldots, n. \tag{153}$$

Since any trapdoor $\mathbf{H_{tr}}$ is reduced to the same matrix $\mathbf{H_{syst}}$, the breaking party has to solve the following nonlinear algebraic system:

$$\frac{y_j}{y_i} \prod_{k=1,k\neq i}^{r} \frac{\beta_j - \beta_k}{\beta_i - \beta_k} = R_{i,j}, \qquad i = 1, 2, \ldots, r; \qquad j = r+1, r+2, \ldots, n, \tag{154}$$

where y_j and β_j are unknowns but all $R_{i,j}$ are known.

Theorem 5. *Suppose that any three β_k, β_p, β_q are given and any y_s is given; then all others y_j and β_j can be obtained uniquely from eq. (154).*

Without loss of generality let β_1, β_2, β_{r+1}, y_1 be given. Apply the following procedure.

1. Calculate the ratio
$$\frac{R_{1,r+1}}{R_{2,r+1}} = c\frac{\beta_{r+1} - \beta_2}{\beta_{r+1} - \beta_1}.$$
Only c is unknown in this equation. Calculate c.

2. For $j = r+2, r+3, \ldots, n$, calculate the ratios
$$\frac{R_{1,j}}{R_{2,j}} = c\frac{\beta_j - \beta_2}{\beta_j - \beta_1}.$$
Since c is already known, only β_j is unknown in this equation. Calculate β_j.

3. For $i = 3, \ldots, r$, calculate the ratios
$$\frac{R_{1,r+1}}{R_{i,r+1}} \frac{R_{i,r+2}}{R_{1,r+2}} = \frac{\beta_{r+1} - \beta_i}{\beta_{r+2} - \beta_i} \frac{\beta_{r+2} - \beta_2}{\beta_{r+1} - \beta_1}.$$
Only β_i is unknown in this equation. Calculate β_i.

4. For $j = r+1, r+2, \ldots, n$, calculate

$$y_j = y_1 R_{1,j} \prod_{k=1, k\neq j}^{r} \frac{\beta_1 - \beta_k}{\beta_j - \beta_k}$$

(y_1 is given).

5. For $i = 2, \ldots, r$ and some $j \geq r+1$, calculate

$$y_i = y_1 R_{i,j} \prod_{k=1, k\neq j}^{r} \frac{\beta_i - \beta_k}{\beta_j - \beta_k}.$$

6. Calculate the trapdoor matrix $\mathbf{H_{tr}}$. □

The Sidelnikov-Shestakov algorithm of breaking the *Niederreiter* PKC follows immediately from Theorem 5:

- Reduce the Public key $\mathbf{H_{cr}}$ to the systematic form $\mathbf{H_{syst}}$. This requires $O(n^3)$ calculations.
- Calculate the trapdoor matrix $\mathbf{H_{tr}}$ using Theorem 5. This requires $O(n^3)$ calculations.
- Calculate the trapdoor matrix $\mathbf{S_{tr}}$ using $\mathbf{H_{cr}}$ and $\mathbf{H_{tr}}$. This requires $O(n^3)$ calculations.

The Work function of breaking is $O(n^3)$. This means that the Sidelnikov-Shestakov algorithm breaks the original Niederreiter PKC completely!

3.3. Modification of the Niederreiter PKC. The *Niederreiter* PKC public key is a scrambled parity-check matrix of some GRS code

$$\mathbf{H}_{cr} = \mathbf{SH}.$$

We change it to

$$\mathbf{H_{cr}^{mod}} = \mathbf{S}(\mathbf{H} + \mathbf{X}) = \mathbf{H}_{cr} + \mathbf{SX},$$

where $\mathbf{X}$ is a matrix of rank 1:

$$\mathbf{X} = \mathbf{a}^t\mathbf{b},$$

with $\mathbf{a}$=$(a_1, a_2, \ldots, a_r)$, $\mathbf{b}$=$(b_1, b_2, \ldots, b_n)$, where $a_i \in GF(q)$ and $b_i \in GF(q)$. The vector $\mathbf{a}$ must be a syndrom of a coset of the GRS code of weight $d-1$.

In the original system, the ciphertext $\mathbf{c}$ is the syndrome of the plaintext $\mathbf{m}$, which is considered as a correctable "error": $w_H(\mathbf{m}) \leq (r-1)/2 = t$ and

$$\mathbf{c} = \mathbf{m}(\mathbf{H}_{cr})^t.$$

Now the ciphertext is

$$\mathbf{c}^{mod} = \mathbf{m}(\mathbf{H}_{cr}^t + \mathbf{X}^t\mathbf{S}^t) = \mathbf{c} + \mathbf{m}\mathbf{X}^t\mathbf{S}^t,$$

where $w_H(\mathbf{m}) \leq t-1$ (and not t).

The legal user knows that $\mathbf{m}\mathbf{X}^t$ is either 0 or a vector $\lambda\mathbf{a}$, where λ depends on the value of $(m_1, ..., m_n)$ and $\mathbf{a}$ is fixed. Hence if the decoder cannot decode the ciphertext assuming that $\mathbf{m}\mathbf{X} = 0$, or if the detected error is greater than t, he has to try at most all the different $q-1$ possible values of λ. But the weight of the errors does not exceed $t-1$, so the problem he is confronted with has a solution.

Consider again reducing the matrix $\mathbf{H}_{\mathbf{cr}}^{\mathbf{mod}}$ to the systematic form. It can be shown that

$$\mathbf{H}_{\mathbf{syst}}^{\mathbf{mod}} = [\mathbf{E}_\mathbf{r}\ \mathbf{S}] = \left[\mathbf{E}_\mathbf{r}\quad \mathbf{R} + \lambda^\mathbf{t}\nu\right], \tag{155}$$

where $\mathbf{R}$ is a matrix given by eq. (153), λ=$(\lambda_1, \lambda_2, \ldots, \lambda_r)$ and $\nu = (\nu_{r+1}, \nu_{r+2}, \ldots, \nu_n)$ are some r- and $(n-r)$-vectors.

To break the cryptosystem, one has to solve the system of equations

$$R_{i,j} + \lambda_i\nu_j = S_{i,j}, \qquad i = 1, 2, \ldots, r; \qquad j = r+1, r+2, \ldots, n. \tag{156}$$

If the λ_i and ν_j are known, the Sidelnikov-Shestakov attack can be applied. To obtain λ_i and ν_j we need the following simple

Lemma 6. *The rank of the matrix* $[R_{i,j}^{-1}]$ *is equal to* 2 *(see Eq. (23)), or, in other words, the determinants of all square submatrices of order* 3 *are equal to* 0.

Using the $R_{i,j}$'s from eq. (156) and eq. (23), we get the equation

$$\det\begin{bmatrix} \frac{1}{S_{i_1,j_1}-\lambda_{i_1}\nu_{j_1}} & \frac{1}{S_{i_1,j_2}-\lambda_{i_1}\nu_{j_2}} & \frac{1}{S_{i_1,j_3}-\lambda_{i_1}\nu_{j_3}} \\ \frac{1}{S_{i_2,j_1}-\lambda_{i_2}\nu_{j_1}} & \frac{1}{S_{i_2,j_2}-\lambda_{i_2}\nu_{j_2}} & \frac{1}{S_{i_2,j_3}-\lambda_{i_2}\nu_{j_3}} \\ \frac{1}{S_{i_3,j_1}-\lambda_{i_3}\nu_{j_1}} & \frac{1}{S_{i_3,j_2}-\lambda_{i_3}\nu_{j_2}} & \frac{1}{S_{i_3,j_3}-\lambda_{i_3}\nu_{j_3}} \end{bmatrix} = 0, \tag{157}$$

in the 6 variables $\lambda_{i_1}, \lambda_{i_2}, \lambda_{i_3}, \nu_{j_1}, \nu_{j_2}, \nu_{j_3}$ of degree at most 12. The full number of equations is equal to $\binom{n-r}{3} / \binom{r}{3}$ and they contain $r(n-r)$ variables. Each variable λ_i is contained in $\binom{n-r}{3} / \binom{r-1}{2}$ equations, each variable ν_j is contained in $\binom{n-r-1}{2} / \binom{r}{3}$ equations.

It is unknown how to solve this system in appropriate time. Using the approach given in [14] we can only get the estimation $O(exp(r(n-r)))$.

Note that in specific cases breaking the modified *Niederreiter* PKC is still possible. Sidelnikov and Shestakov examined a hiding matrix X for which all rows except the last one are zero [10]. They showed that one can get a system of $n-r-1$ equations of order at most 4 in three variables. This system can be solved using $O(r^3+rn)$ calculations.

This example shows that the hiding matrix X should be chosen and examined very carefully.

4. The McEliece Cryptosystem based on Goppa Codes

In [5], the authors showed that the problem of decoding a general binary linear code is a NP-complete. Hence, if there exists a big set of codes of large distance having *fast decoding* algorithms, then someone can choose one of these codes as a public key with the fast decoding algorithm as a secret key, in the hope that the enemy party can decode intercepted ciphertexts only as corrupted codewords of a general linear code.

The first public-key cryptosystem based on linear codes was proposed by McEliece in 1978 [4]. The system is based on the family of *Goppa* codes. *Goppa* codes have better parameters than general alternant codes in the binary case only. Thus, a large binary field $GF(2^m)$ and the base field $GF(2)$ are used. Corresponding to each irreducible polynomial of degree t over $GF(2^m)$, there exists a binary irreducible Goppa code of length 2^m and dimension $k \geq n-mt$, capable of correcting any t or fewer errors. There exist fast algorithms for decoding these codes in $O(nt)$ [15], [17] or even in $O(n\log^2 n)$ arithmetic operations [16].

4.1. Description.

Private keys

The legal user A chooses the following private keys.

1. A monic primitive irreducible polynomial

$$g(x) = g_0 + g_1x + \ldots + g_{t-1}x^{t-1} + x^t$$

of degree t over the field $GF(2^m)$.

2. An ordering

$$\overline{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n), \quad n = 2^m,$$

of the elements of $GF(2^m)$.

3. A fast decoding Algorithm, say, based on the parity check matrix

$$\mathbf{H} = [\alpha_j^i/g(\alpha_j)],$$

$$i = 0, 1, \ldots, t-1; \; j = 1, 2, \ldots, n$$

(see Section 2.2).

4. A non-singular *binary* scramble matrix $\mathbf{S}$ of order k.

Public key

The legal user A chooses a bijective mapping

$$\mathcal{B}: \ GF(2^m) \ \rightarrow \ GF(2)^m, \tag{158}$$

which maps each element ω of the large field $GF(2^m)$ into the binary column m-vector $(a_1, a_2, \ldots, a_m)^t$:

$$\mathcal{B}(\omega) = (a_1, a_2, \ldots, a_m)^t. \tag{159}$$

Denote as $\mathbf{H}_{bin}$ the binary parity check matrix of the Goppa code obtained from the binary matrix

$$\widetilde{\mathbf{H}} = [\mathcal{B}(\alpha_j^i / g(\alpha_j))] \tag{160}$$

by deleting linearly dependent rows.

A *generator* matrix $\mathbf{G}$ is calculated as a binary $k \times n$ matrix of rank k of maximal size such that

$$\mathbf{G}\mathbf{H}_{bin}^t = 0.$$

The legal user A calculates the public key $\mathbf{G_{cr}}$ as the scrambled binary $k \times n$ generator matrix

$$\mathbf{G_{cr}} = \mathbf{SG}.$$

Encryption

Let $\mathbf{m} = (m_1, m_2, \ldots, m_k)$ be a k-bit plaintext. A ciphertext is given by

$$\mathbf{c} = \mathbf{m}\mathbf{G_{cr}} + \mathbf{e}, \tag{161}$$

where $\mathbf{e}$ is an artificial vector of errors of weight t or less, which is randomly chosen and added by the sending party.

Decryption

Upon receiving $\mathbf{c}$, the legitimate receiver applies a fast decoding algorithm (see Section 2.2) to obtain $\mathbf{mS}$ and then multiplies it by $\mathbf{S^{-1}}$ to obtain the plaintext $\mathbf{m}$.

The McEliece example

The chosen Goppa code has the following parameters:

$$\begin{aligned} &n = 1024, \\ &k = 524, \\ &t = (d-1)/2 = 50. \end{aligned}$$

There exist about 10^{149} possible Goppa polynomials, about 1000! orderings $\overline{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, and about 10^{750} choices for the scramble matrix $\mathbf{S}$. Hence, a brute-force attack based on comparing a ciphertext to each codeword seems to be infeasible.

4.2. Information sets and decoding general linear codes.

Let

$$\mathbf{G} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{bmatrix}$$

be a $k \times n$ generator binary matrix of an $(n,\ k,\ d)$ binary linear code. Let

$$\mathcal{J} = \{j_1, j_2, \ldots, j_k\}$$

be a k-subset of the set $\{1, 2, \ldots, n\}$, where $1 \leq j_1 < j_2 < \ldots < j_k \leq n$. Let

$$\mathbf{G}(\mathcal{J}) = \mathbf{G}(j_1, j_2, \ldots, j_k)$$

denote the square $k \times k$ submatrix consisting of the columns with the numbers $j_1, j_2, \ldots, j_k$. For any codeword $\mathbf{g} =(g_1, g_2, \ldots g_n 0$, let

$$\mathbf{g}(\mathcal{J})=(g_{j_1}, g_{j_2}, \ldots, g_{j_k})$$

be the $\mathcal{J}$-subword of length k associated with the subset $\mathcal{J}$.

If $\mathbf{m} = (m_1, m_2, \ldots, m_k)$ is a binary information vector, the corresponding codeword is calculated as

$$\mathbf{g} =(g_1, g_2, \ldots, g_n) = \mathbf{mG},$$

and the corresponding $\mathcal{J}$-subword can be calculated as

$$\mathbf{g}(\mathcal{J})=(g_{j_1}, g_{j_2}, \ldots, g_{j_k}) = \mathbf{mG}(\mathcal{J}) = \mathbf{mG}(j_1, j_2, \ldots, j_k).$$

Suppose that the matrix $\mathbf{G}(\mathcal{J}) = \mathbf{G}(j_1, j_2, \ldots, j_k)$ is non singular. Then we can obtain the *information vector* $\mathbf{m}$ from the $\mathcal{J}$-subword by

$$\mathbf{m} = \mathbf{g}(\mathcal{J})\mathbf{G}(\mathcal{J})^{-1} = \mathbf{g}(\mathcal{J})\mathbf{G}(j_1, j_2, \ldots, j_k)^{-1}. \tag{162}$$

A set $\mathcal{J} = \{j_1, j_2, \ldots, j_k\}$ is said to be an *information set* if the corresponding submatrix $\mathbf{G}(\mathcal{J}) = \mathbf{G}(j_1, j_2, \ldots, j_k)$ is non singular.

Information sets can be used for decoding a general linear code. Let

$$\mathbf{y} = \mathbf{g} + \mathbf{e}$$

be a received word with t or less errors.

For the given code, consider a set $\mathcal{I}$ of distinct information sets with the property:

$$\text{for any pattern } \mathbf{e} \text{ of } t \text{ errors, there exists at least one Information set } \mathcal{J} \in \mathcal{I}$$
$$\text{such that the } \mathcal{J}\text{-subword } \mathbf{y}(\mathcal{J}) = \mathbf{g}(\mathcal{J}) \text{ is free of errors.} \tag{163}$$

The procedure of decoding is as follows.

Main algorithm

1. Start with $\mathcal{J}_1 \in \mathcal{I}$.
2. Calculate by eq. (21) an estimation $\mathbf{m}_1 = \mathbf{y}(\mathcal{J}_1)\mathbf{G}_1(\mathcal{J})^{-1}$ of the information vector $\mathbf{m}$.
3. Calculate the codeword
$$\mathbf{g}_1 = \mathbf{m}_1\mathbf{G}.$$
4. Calculate the Hamming distance
$$d_1 = w_H(\mathbf{g}_1 - \mathbf{y})$$
between $\mathbf{g}_1$ and received vector $\mathbf{y}$.
5. If $d_1 \le t$, then $\mathbf{m}_1 = \mathbf{m}$. The decoding is finished.
6. If $d_1 > t$, then choose the next set $\mathcal{J}_2 \in \mathcal{I}$ and run the above procedure again.

Due to the property (163), the procedure gives a correct answer at some step.

The **Main algorithm** can be improved if for the case $d_1 > t$ we examine all error patterns of weight l or less inside the chosen set $\mathcal{J}_1$. It was found that the best choice is $l = 2$ [19].

We are interested in getting the set $\mathcal{I}$ as small as possible. The cardinality $|\mathcal{I}|$ of distinct information sets with property (163) depends on the code. It is clearly upperbounded by
$$|\mathcal{I}| \le \binom{n}{k}. \tag{164}$$
The (trivial) lower bound is given by
$$\frac{\binom{n}{k}}{\binom{n-t}{k}} = \frac{\binom{n}{t}}{\binom{n-k}{t}} \le |\mathcal{I}|. \tag{165}$$
This is because if a $\mathcal{J}$-subword is free of errors,all t errors are in the $n-k$ complementary positions. These positions can cover $\binom{n-k}{t}$ patterns $\mathbf{e}$ of t errors. Hence, we need at least
$$\frac{\binom{n}{t}}{\binom{n-k}{t}}$$
such complementary sets to cover all patterns $\mathbf{e}$ of t errors.

E.A.Kruk [20] showed that, for almost all linear binary (n,k,d) codes,
$$|\mathcal{I}| \le \gamma(n)\frac{\binom{n}{t}}{\binom{n-k}{t}}, \tag{166}$$

where $\gamma(n)$ is function which increases slowly with n ($\gamma(n) \sim n^2$ in Kruk's paper). This is the best known general bound. It can be improved for some codes of restricted lengths.

4.3. Work Functions for the McEliece PKC

Attempts of Breaking. Two kinds of attacks on a PKC can be considered. The first kind of attacks is based on getting a plaintext from an intercepted ciphertext. The second kind of attack is based on getting private keys from known public keys.

Getting a plaintext from a ciphertext. Let

$$\mathbf{c} = \mathbf{m}\mathbf{G}_{\mathbf{cr}} + \mathbf{e}$$

be the intercepted ciphertext. The enemy party knows the generator matrix $\mathbf{G}_{\mathbf{cr}}$ and uses decoding by means of information sets. He need not know the set $\mathcal{I}$ of information sets. He simply tries all k-subsets $\mathcal{J}$ and decodes $\mathbf{c}$ in accordance with the **main algorithm** (see Section 4.2). Sometimes a k-subset $\mathcal{J}$ is not an information set. In this case, it is impossible to invert the matrix $\mathbf{G}(\mathcal{J})$ at Step 2 of the **main algorithm**. So the next k-subset should be chosen. But discovering that the matrix $\mathbf{G}(\mathcal{J})$ is non-invertible takes about the same number of arithmetic operations as inverting a non-singular matrix.

McEliece [4] analyzed the *work function* W_1 of breaking the PKC for this case using the bound (164) and the following probabilistic approach.

Choose a k-subset $\mathcal{J}$ (not necessarily, an information set) with the uniform distribution:

$$P(\mathcal{J}) = \frac{1}{\binom{n}{k}}. \tag{167}$$

If t out of n positions are in error, the number of error-free sets $\mathcal{J}$ is $\binom{n-t}{k}$. Hence, the probability of successful decoding is

$$P_s = \frac{L}{\binom{n}{k}}, \tag{168}$$

where L is the number of error-free sets $\mathcal{J}$ such that the corresponding matrix $\mathbf{G}(\mathcal{J})$ is invertible. It can be shown that, for almost all binary linear codes,

$$L = c\binom{n-t}{k},$$

where

$$c = \prod_{i=1}^{k}\left(1 - \frac{1}{2^i}\right).$$

For large k, $c = 0.288788$.

Thus, the *average* number of attempts to get the plaintext $\mathbf{m}$ from the intercepted ciphertext $\mathbf{c}$ is

$$N = \frac{1}{P_s} = \frac{\binom{n}{k}}{L} \simeq \frac{\binom{n}{k}}{c\binom{n-t}{k}} \simeq 3.5\frac{\binom{n}{k}}{\binom{n-t}{k}}. \tag{169}$$

At each attempt, the cryptanalyst should invert a $k \times k$ matrix $\mathbf{G}(\mathcal{J})$. It takes about k^3 arithmetic operations. Hence, the *average work function* is

$$\overline{W_1} = 3.5 \cdot k^3 \frac{\binom{n}{k}}{\binom{n-t}{k}}. \tag{170}$$

Note that this is an average job. It requires sometimes much more attempts.

For the McEliece example, we obtain from Eq. (170)

$$\overline{W_1} = 3.5 \cdot 2 \cdot 10^{25} = 2^{82.5}.$$

Adams and Meijer [18] analyzed Eq. (170) for $n = 1024$ and found that W_1 is maximal if $k = 654$, $t = 37$. They obtained

$$\overline{W_1} = 2^{85.9}.$$

Lee and Brickell [19] used the improved **main algorithm** and obtained for $n = 1024$, $k = 654$, $t = 37$ that

$$\overline{W_1} = 2^{75.2}.$$

E.A. Kruk [20], [21] significantly improved the decoding algorithm and obtained that the *maximal* value of the *work function* is equal to

$$\max W_1 = (n-k)^3 \frac{\binom{n}{t}}{\binom{n-t}{t}} = (n-k)^3 \frac{\binom{n}{k}}{\binom{n-t}{k}}. \tag{171}$$

Thus, the maximal job for the Kruk algorithm is almost the same as the average job $\overline{W_1}$ for the probabilistic approach.

Kruk also found for the McEliece example $n = 1024$, $k = 524$, $t = 50$ that there exists a set $\mathcal{I}$ of information sets such that

$$W_1 = 2^{59}.$$

This is the best known attack of the first kind on the McEliece PKC.

Getting the secret key from the public key. The second kind of attacks was analyzed in [7, 6].

Gibson [7] showed that if the *enemy party knows not only the public key but also the ordering*

$$\overline{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$$

then the McEliece PKC can be broken very easily. Indeed, let

$$\mathbf{g} = (g_1, g_2, \ldots, g_n)$$

be any codeword of the chosen Goppa code with the binary generator matrix $\mathbf{G_{cr}}$. Let

$$g_{i_1} = g_{i_2} = \ldots = g_{i_m} = 1$$

be all non-zero components of $\mathbf{g}$. Let

$$p(x) = (x - \alpha_{i_1})(x - \alpha_{i_2}) \ldots (x - \alpha_{i_m}).$$

Then the formal derivative $p(x)'$ is a multiple of the square Goppa polynomial $g(x)^2$. For each row of the binary generator matrix $\mathbf{G_{cr}}$, the cryptanalyst can obtain polynomials the p and p' and calculate a non trivial greatest common divisor. This allows him to recover the Goppa polynomial $g(x)$. Hence, the cryptanalyst gets all the secret keys and can apply the fast decoding algorithm.

It is interesting to note that if the cryptanalyst knows $\mathbf{G_{cr}}$ *and* $g(x)$ then this still does not seem to permit an easy recovery of the ordering $\overline{\alpha}$, and hence does not permit to obtain the fast decoding algorithm.

Another idea is finding an overlying generalized Reed-Solomon code for the given Goppa code.

Shamir and Heiman [6] tried to break the McEliece PKC based on this idea. They found a partial set of equations to obtain the private keys, but these have too many solutions other than the private keys. Thus, they did not succeed in the full breaking, but a GRS code can be extracted from their equations.

We present a new method of embedding the Goppa code into a GRS code based on the Sidelnikov-Shestakov attack.

Recall that the parity check matrix of the Goppa code is given by the matrix

$$\mathbf{H} = \left[\alpha_j^i / g(\alpha_j)\right], \ i = 0, 1, \ldots, t-1, \ j = 1, 2, \ldots, n$$

over the large field $GF(2^m)$. It can be reduced to the systematic form $\mathbf{H}_{syst} = [\mathbf{E}_t \ \mathbf{R}]$, where $\mathbf{E}_t$ is the $t \times t$ identity matrix and $\mathbf{R} = [u_i v_j / (y_i - y_j)]$ is a generalized Cauchy matrix.

The enemy party calculates from the public key $\mathbf{G_{cr}}$ any binary $tm \times n$ parity check matrix $\mathbf{H}_{bin}$. To embed the Goppa code in a GRS code, he should find a non-singular square $mt \times mt$ matrix $\mathbf{A} = [a_{ij}]$ and unknowns u_i, v_j, y_i, y_j such that

$$\mathbf{A}\mathbf{H}_{bin} = \mathcal{B}(\mathbf{H}_{syst}) = [\mathcal{B}(\mathbf{E}_t) \ \mathcal{B}(\mathbf{R})]$$

for a bijective mapping $\mathcal{B} : \ GF(2^m) \ \rightarrow \ GF(2)^m$. Let $\mathbf{H}_{1bin}$ is the submatrix of $\mathbf{H}_{bin}$ consisting of the first t columns and $\mathbf{H}_{2bin}$ be the submatrix consisting of the rest of the columns.

First, he solves a *linear* system

$$\mathbf{AH}_{1bin} = \mathcal{B}(\mathbf{E}_t)$$

of mt^2 equations in the m^2t^2 unknowns a_{ij}.

Then, for any solution $\mathbf{A}$, he calculates $\mathbf{AH}_{2bin}$, $\mathcal{B}^{-1}(\mathbf{AH}_{1bin})$ and solves the system

$$[u_i v_j/(y_i - y_j)] = \mathcal{B}^{-1}(\mathbf{AH}_{1bin}).$$

(this is the Sidelnikov-Shestakov attack).

If he succeeds, then the problem is solved. If not, he tries another solution $\mathbf{A}$. It can be shown that *in average* the number of attempts is restricted.

Embedding the Goppa code into a GRS code does not solve the problem of breaking. It only allows to try to get the Goppa polynomial from $\mathbf{H}_{syst}$. A regular method for doing it has not yet been found.

Thus, there is still no progress in breaking the McEliece PKC.

4.4. Modification the McEliece PKC: Introducing a hiding matrix.

The principle of a hiding matrix is proposed to modify the McEliece PKC. The modified public key is as follows:

$$\mathbf{G}_{\mathbf{cr}}^{\mathbf{mod}} = \mathbf{SG} + \mathbf{X},$$

where $\mathbf{X}$ is a specific matrix of rank 1.

The main idea is as follows. A legal party chooses some random matrix $\mathbf{X}$ as an extra secret key and adds it to the original public key to produce a new modified public key. Thus any visible structure of the public key will be hidden.

The problem in adding a hiding matrix $\mathbf{X}$ to the McEliece PKC is to keep the overall error lower than t, the error-correcting capacity of the legal users decoder, for all possible messages. On the other hand, the hiding matrix should have as many free parameters as possible. This dilemma can be solved at the expense of having to lower the error-correcting capacity needed by the decoder from t to $t-1$. So if the cryptanalyst can guess (or knows) the correct secret $\mathbf{X}$, his job is only slightly easier than for the original PKC.

The hiding matrix $\mathbf{X}$ is a secret $k \times n$ matrix of the form

$$\mathbf{X} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{pmatrix} \begin{pmatrix} x_1 & x_2 & \dots & x_n \end{pmatrix}, \qquad (172)$$

where $(a_1, a_2, \dots, a_k)$ is a randomly chosen binary k-vector. The binary n-vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$ should be a leader of some coset of the Goppa code of weight $w_H(\mathbf{x}) = d - 1 = 2t$.

Let $\mathbf{m} = (m_1, ..., m_k)$ be an information vector to be encrypted. A message is then encrypted as

$$\mathbf{c} = \mathbf{m}\mathbf{G}_{\mathbf{cr}}^{\mathbf{mod}} + \mathbf{e} = \mathbf{m}\mathbf{G}_{\mathbf{cr}} + \mathbf{m}\mathbf{X} + \mathbf{e}$$

where $w_H(\mathbf{e}) \leq t - 1$ and not t, and $\mathbf{x} + \mathbf{e}$ should not be a codeword. The weight $w_H(\mathbf{m}\mathbf{X})$ can either be equal to 0, and then the overall error is $t - 1$, or $w_H(\mathbf{m}\mathbf{X}) = d - 1 = 2t$, and then, with $d = 2t + 1$, we see that

$$w_H(\mathbf{m}\mathbf{X} + \mathbf{e}) \geq d - 1 - (t - 1) = t + 1,$$

so *even* if the enemy party knows a decoder correcting t errors for the chosen Goppa code, he cannot eliminate all the errors.

The legal user knows that $\mathbf{m}\mathbf{X}$ is either 0 or $\mathbf{x} =(x_1, ..., x_n)$. So he first assumes that $\mathbf{m}\mathbf{X} = 0$, and if the weight of the computed error is greater than t for some possible code vector, he adds $\mathbf{x} =(x_1, ..., x_n)$ to the ciphertext and decodes once again.

The advantages of this scheme lie in the greater difficulty for the cryptanalyst to find the generator matrix $\mathbf{G}_{\mathbf{cr}}$ for the chosen Goppa code in virtue of the hiding matrix $\mathbf{X}$, for which a large choice exists. A regular method for determining and eliminating the correct $\mathbf{X}$ has not yet been found. Moreover, even if the cryptanalyst possesses an equivalent t-error-correcting decoder for $\mathbf{G}_{\mathbf{cr}}$, it will fail to correct all the $t + 1$ errors.

The disadvantages of the proposed system are, first, that the enemy cryptanalyst need only find a $t - 1$ error-correcting decoder if he disposes of the hiding matrix $\mathbf{X}$ (in McElieces proposal [4], $t = 49$ instead of 50), and, second, that the legal user sometimes has to decode twice.

5. The Public-Key Cryptosystem based on Rank codes

This cryptosystem is proposed in [12] and is known as GPT PKC.

5.1. Description.

Private keys

The legitimate user A chooses as private keys:

1. The generator $k \times n$ matrix $\mathbf{G}$ of an MRD code (see Section 2.4) of rank distance $d = 2t + 1$:

$$\mathbf{G} = \begin{bmatrix} g_1 & g_2 & \cdots & g_n \\ g_1^q & g_2^q & \cdots & g_n^q \\ g_1^{q^2} & g_2^{q^2} & \cdots & g_n^{q^2} \\ \dots & \dots & \dots & \dots \\ g_1^{q^{k-1}} & g_2^{q^{k-1}} & \cdots & g_n^{q^{k-1}} \end{bmatrix}. \qquad (173)$$

Remark 3. *Let* $\mathbf{U}$ *be an* $k \times k$ *diagonal matrix*

$$\mathbf{U} = \mathrm{diag}[u, u^q, u^{q^2}, \ldots, u^{q^{k-1}}], \tag{174}$$

where $u \in GF(q^N)$, $u \neq 0$. *Then the generator matrix*

$$\widetilde{\mathbf{G}} = \mathbf{UG} = \begin{bmatrix} \widetilde{g}_1 & \widetilde{g}_2 & \cdots & \widetilde{g}_n \\ \widetilde{g}_1^q & \widetilde{g}_2^q & \cdots & \widetilde{g}_n^q \\ \widetilde{g}_1^{q^2} & \widetilde{g}_2^{q^2} & \cdots & \widetilde{g}_n^{q^2} \\ \cdots & \cdots & \cdots & \cdots \\ \widetilde{g}_1^{q^{k-1}} & \widetilde{g}_2^{q^{k-1}} & \cdots & \widetilde{g}_n^{q^{k-1}} \end{bmatrix}, \tag{175}$$

with $\tilde{g}_j = ug_j$, $j = 1, 2, \ldots, n$, *defines the same MRD code* $\mathcal{C}$ *and has the same structure. There are no other generator matrices of* $\mathcal{C}$ *of the same structure. In particular, for some* j, *one can choose* $u = g_j^{-1}, \tilde{g}_j = 1$. *In this case, the elements of the* j*th column are equal to* 1.

2. A Fast Decoding Algorithm (see Section 2.4).

3. A non-singular scrambling matrix $\mathbf{S}$ of order k. This matrix is used to scramble the generator matrix, i.e. to destroy any evident structure of the generator matrix.

4. A randomly chosen $k \times n$ matrix $\mathbf{X}$ such that for any k-vector $\mathbf{m}$ the vector $\mathbf{mX}$ has rank norm not greater than $t_1 < t$, where t_1 is a *design* parameter.

Such a matrix can be chosen as follows. Let $\mathbf{A}$ be a $k \times l$ matrix of rank l over the extended field $GF(q^N)$, $1 \leq l \leq t_1$. Let $\mathbf{B}$ be an $l \times n$ matrix of rank t_1 over the *base* field $GF(q)$. Then

$$\mathbf{X} = \mathbf{AB}. \tag{176}$$

Indeed, for any k-vector $\mathbf{m}$, we have

$$r(\mathbf{mX}|q) = r(\mathbf{mAB}|q) \leq \min\{r(\mathbf{m}|q), r(\mathbf{A}|q), r(\mathbf{B}|q)\} \leq t_1$$

because $r(\mathbf{B}|q) = t_1$.

The matrix $\mathbf{B}$ can be constructed as follows. First, choose a matrix $\mathbf{B}_0 = [\mathbf{Q}\ \mathbf{O}]$, where $\mathbf{Q}$ is an $l \times t_1$ matrix with entries in the extended $GF(q^N)$ and of rank t_1 over the base field $GF(q)$, and $\mathbf{O}$ is a $l \times (n - t_1)$ matrix of 0's. Then multiply $\mathbf{B}_0$ to the right by an $n \times n$ non-singular matrix $\mathbf{P}$ with entries in the *base* field $GF(q)$. It is clear, that $r(\mathbf{B}|q) = t_1$.

Public key

The legitimate user A calculates the matrix

$$\mathbf{C_{cr}} = \mathbf{SG} + \mathbf{X}. \tag{177}$$

The legitimate user A publishes the matrix $\mathbf{C_{cr}}$ as a *public key* in the directory, in the hope that it is very difficult to get the secret matrices $\mathbf{S}$, $\mathbf{G}$ and $\mathbf{X}$ separately from eq.177.

Encryption

In this Cryptosystem, all the possible plaintexts (messages) are k-vectors

$$\mathbf{m} = (m_1, m_2, \ldots, m_k)$$

with components in $GF(q^N)$.

If B wants to send a secret message to A, he chooses a plaintext $\mathbf{m} = (m_1, m_2, \ldots, m_k)$ and calculates the *ciphertext* $\mathbf{c}$ as

$$\mathbf{c} = \mathbf{m}\mathbf{G_{cr}} + \mathbf{e} = \mathbf{mSG} + (\mathbf{mX} + \mathbf{e}), \tag{178}$$

where $\mathbf{e}$ is an artificial vector of errors of the rank $t_2 = t - t_1$ or less, randomly chosen and added by the sending party. Note that, for any plaintext $\mathbf{m}$, we have

$$r(\mathbf{mX} + \mathbf{e}|\mathbf{q}) \leq r(\mathbf{mX}|q) + r(\mathbf{e}|q) \leq t_1 + t_2 = t.$$

Hence, in this cryptosystem, the set of ciphertexts is the set of all the possible codewords of the chosen (n, k, d) MRD code corrupted by errors $\mathbf{e}$ of rank not greater than $t = \left\lfloor \frac{d-1}{2} \right\rfloor$.

Decryption

Upon receiving the ciphertext $\mathbf{c}$,the legitimate receiver applies a fast decoding algorithm (see Section 2.4) to obtain $\mathbf{mS}$ and then multiplies it by $\mathbf{S^{-1}}$ to obtain the plaintext $\mathbf{m}$.

5.2. Attacks on the GPT PKC. Two kinds of attacks on any PKC can be considered. The first kind of attacks is based on getting a plaintext from an intercepted ciphertext. The second kind of attack is based on getting private keys from known public keys.

Getting a plaintext from a ciphertext. The enemy party, after interception of the ciphertext $\mathbf{c}$, can try something like decoding by means of information sets.

The chosen MRD code has dimension $k = n - d + 1$. Hence, if *Hamming* weight of an error $\mathbf{e}$ is greater than or equal to d, then there is no error free information set at all!

But most errors $\mathbf{e}$ of rank t have Hamming weight greater than $2t$ and, therefore, this attack is absolutely inefficient. Indeed, it can be shown (see, for instance, [11]) that the number $A_s(n)$ of n-vectors having rank s is given by

$$A_s(n) = \frac{(q^N-1)(q^N-q)\dots(q^N-q^{s-1})(q^n-1)(q^n-q)\dots(q^n-q^{s-1})}{(q^s-1)(q^s-q)\dots(q^s-q^{s-1})} \tag{179}$$

and the number $A_s(n,i)$ of n-vectors having simultaneously rank s and Hamming weight $i \geq s$ is given by

$$A_s(n,i) = \binom{n}{i}\sum_{k=0}^{i-s}(-1)^k\binom{i}{k}A_s(i-k). \tag{180}$$

Using eqs 179-180, one can show that the fraction of vectors of rank s and having the Hamming weight greater than $2s$ is closed to 1 for large n.

For example, let $q = 2$, $n = N$. Consider vectors of rank $t = 1$. We have, $A_1(n) = (2^N-1)^2$. The number $A_1(n, i \geq 3)$ of vectors of rank 1 having Hamming weight greater than or equal to $2t+1 = 3$ is equal to $A_1(n)-(2^N-1)(\binom{N}{1}+\binom{N}{2})$. Hence, the fraction γ of non-correctable errors of rank 1 is

$$\gamma = \frac{A_1(n, i \geq 3)}{A_1(n)} = 1 - \frac{N(N+1)}{2(2^N-1)}.$$

If $N = 10$, then $\gamma \simeq 0.95$.

The only possibility to get a plaintext from the intercepted ciphertext is an exhaustive search for all possible errors of rank t_1 in a chosen information set of lenght k. It requires about

$$W \simeq A_{t_1}(k) > q^{(N+k-t_1)t_1}$$

arithmetical operations in the extended field $GF(q^N)$.

An illustrative example

Let $q = 2$, $N = n = 20$. Let us choose an (n,k,d) MRD code with parameters $n = 20$, $k = 12$, $d = 9$. This code allows to correct errors of rank less than or equal to $t = 4$. We can choose the design parameter $t_1 = 2$ or 3. The performance of the designed cryptosystem is as follows:

- Rate $R = \frac{k}{n} = 0.6$.
- Size of the Public key $n^2k = 4800$ bits.
- Work function $W \simeq 2^{61}$, if $t_1 = 2$, and $W \simeq 2^{88}$, if $t_1 = 3$.

This example shows that we can get the same parameters as in the McEliece cryptosystem using much a smaller size of the public key.

Getting the private key from the public key:
The Gibson Attack on the GPT PKC. In [12], a GPT PKC was proposed with a special type of hiding matrix $\mathbf{X}$. Namely, we considered the PKC with the public key $\mathbf{G_{cr}} = \mathbf{SG} + \mathbf{X}$, where $\mathbf{G}$ is given by eq. 173. As for the hiding matrix $\mathbf{X}$, it was proposed to be of the form

$$\mathbf{X} = \mathbf{a}^t\mathbf{b},$$

where $\mathbf{a} = (a_1, a_2, \ldots, a_k)$ is a k-vector with nonzero components in $GF(q^N)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_n)$ is an n-vector of rank t_1 over $GF(q)$.

Remark 4. *Note that, for any $c \in GF(q^N)$, $c \neq 0$,*

$$\mathbf{a}^t\mathbf{b} = (c\mathbf{a}^t)(c^{-1}\mathbf{b}).$$

Hence, if the expression $d = a_1b_1 + a_2b_2 + \ldots + a_nb_n$ is nonzero, then without loss of generality one can put $d = 1$.

J.K. Gibson [13] investigated this type of the PKC carefully and found that it can be broken from the practical point of view. We present the Gibson attack in the new form based on the ideas of the Sidelnikov-Shestakov attack on the Niederreiter cryptosystem.

Rewrite $\mathbf{G_{cr}} = \mathbf{SG} + \mathbf{X}$ as

$$\mathbf{G_{cr}} = \mathbf{S}(\mathbf{G} + \mathbf{Y}), \tag{181}$$

where $\mathbf{Y} = \mathbf{S}^{-1}\mathbf{X} = \mathbf{S}^{-1}\mathbf{a}^t\mathbf{b}$.

Denote

$$\mathbf{c}^t = (c_0, c_1, \ldots, c_{k-1})^t = \mathbf{S}^{-1}\mathbf{a}^t. \tag{182}$$

Put $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2)$, where $\mathbf{b}_1 = (b_1, b_2, \ldots, b_k)$ and $\mathbf{b}_2 = (b_{k+1}, b_{k+2}, \ldots, b_n)$.

Denote

$$\mathbf{G} = [\mathbf{G_1}, \mathbf{G_2}],$$

where $\mathbf{G_1}$ is a square submatrix of $\mathbf{G}$ of order k,

$$\mathbf{G_1} = \begin{bmatrix} g_1 & g_2 & \cdots & g_k \\ g_1^q & g_2^q & \cdots & g_k^q \\ g_1^{q^2} & g_2^{q^2} & \cdots & g_k^{q^2} \\ \cdots & \cdots & \cdots & \cdots \\ g_1^{q^{k-1}} & g_2^{q^{k-1}} & \cdots & g_k^{q^{k-1}} \end{bmatrix}, \tag{183}$$

and $\mathbf{G_2}$ is the rest of $\mathbf{G}$,

$$\mathbf{G_2}=\begin{bmatrix} g_{k+1} & g_{k+2} & \cdots & g_n \\ g_{k+1}^q & g_{k+2}^q & \cdots & g_n^q \\ g_{k+1}^{q^2} & g_{k+2}^{q^2} & \cdots & g_n^{q^2} \\ \cdots & \cdots & \cdots & \cdots \\ g_{k+1}^{q^{k-1}} & g_{k+2}^{q^{k-1}} & \cdots & g_n^{q^{k-1}} \end{bmatrix} \tag{184}$$

In a similar manner, we denote

$$\mathbf{X} = [\mathbf{X_1}, \mathbf{X_2}],$$

$$\mathbf{Y} = [\mathbf{Y_1}, \mathbf{Y_2}],$$

where $\mathbf{X_1}, \mathbf{Y_1}$ are square matrices of order k, and

$$\begin{aligned} \mathbf{Y_1} &= \mathbf{S}^{-1}\mathbf{X_1} = \mathbf{c}^t\mathbf{b}_1, \\ \mathbf{Y_2} &= \mathbf{S}^{-1}\mathbf{X_2} = \mathbf{c}^t\mathbf{b}_2. \end{aligned} \tag{185}$$

Reduce the public key $\mathbf{G_{cr}}$ to the systematic form

$$\mathbf{G_{syst}} = (\mathbf{S}\mathbf{G}_1 + \mathbf{X}_1)^{-1}\mathbf{G_{cr}} = [\mathbf{E}_k\ \mathbf{R}],$$

where $\mathbf{E}_k$ is the identity matrix of order k, and

$$\mathbf{R} = [R_{i,j}],$$

$$i = 0, 1, \ldots, k-1;\ j = k+1, k+2, \ldots, n.$$

Calculate $(\mathbf{G_2} + \mathbf{Y_2}) = (\mathbf{G_1} + \mathbf{Y_1})\mathbf{R}$, or,

$$\mathbf{G_1R} = \mathbf{G_2} + \mathbf{Y_2} - \mathbf{Y_1R}. \tag{186}$$

In eq. 186, the matrix $\mathbf{R}$ is known but other matrices $\mathbf{G_1}, \mathbf{G_2}, \mathbf{Y_1}, \mathbf{Y_2}$ are unknown. If one somehow gets a solution of eq. 186, where $\mathbf{G_1}, \mathbf{G_2}$ are of the form (183)-(184) and where $\mathbf{Y_1}, \mathbf{Y_2}$ are of the form (185), one can also obtain the matrix $\mathbf{S}$ from Eq (181) as well as the matrix $\mathbf{X} = \mathbf{SY}$. This means that one gets all the private keys.

We rewrite eq. 186 in the different form.

By definition, put

$$\mathbf{f}^t = (f_0, f_1, \ldots, f_{k-1})^t = \left(c_0, c_1^{q^{N-1}}, \ldots, c_{k-1}^{q^{N-k+1}}\right)^t. \tag{187}$$

Put

$$\mathbf{d} = (d_{k+1}, d_{k+2}, \ldots, d_n) = \mathbf{b}_2 - \mathbf{b}_1\mathbf{R}. \tag{188}$$

From the matrix equation (186), we obtain

$$(\mathbf{G_1R})_{i,j} = (\mathbf{G_2} + \mathbf{Y_2} - \mathbf{Y_1R})_{i,j},$$

$$i = 0, 1, \ldots, k-1;\ j = k+1, k+2, \ldots, n,$$

or, using the notations (183), (184), (182), (185), (188),

$$\begin{gathered} g_1^{q^i} R_{0,j} + g_2^{q^i} R_{1,j} + \ldots + g_k^{q^i} R_{k-1,j} = g_j^{q^i} + c_i\left(\mathbf{b}_2 - \mathbf{b}_1\mathbf{R}\right)_j = g_j^{q^i} + c_i d_j, \\ i = 0, 1, \ldots, k-1;\ j = k+1, k+2, \ldots, n. \end{gathered} \tag{189}$$

For any i, $0 \le i \le k-1$, raise eq. 189 to the power q^{N-i}. Since $g^{q^N} = g$, if $g \in GF(q^N)$ and $c_i^{q^{N-i}} = f_i$ by (187), we obtain the following system of the $k(n-k)$ algebraic equations

$$\begin{gathered} g_1 R_{0,j}^{q^{N-i}} + g_2 R_{1,j}^{q^{N-i}} + \ldots + g_k R_{k-1,j}^{q^{N-i}} = g_j + f_i d_j^{q^{N-i}}, \\ i = 0, 1, \ldots, k-1;\ j = k+1, k+2, \ldots, n, \end{gathered} \tag{190}$$

where $\mathbf{g} = (g_1, g_2, \ldots, g_n)$, $\mathbf{f} = (f_0, f_1, \ldots, f_{k-1})$, $\mathbf{d} = (d_{k+1}, d_{k+2}, \ldots, d_n)$ are the $2n$ unknowns. Using the matrix notations, we can also rewrite this system as

$$\begin{aligned} \mathbf{g_1R_{k+1}} &= g_{k+1}\mathbf{a} + \mathbf{f\,D_{k+1}}, \\ \mathbf{g_1R_{k+2}} &= g_{k+2}\mathbf{a} + \mathbf{f\,D_{k+2}}, \\ \mathbf{g_1R_{k+3}} &= g_{k+3}\mathbf{a} + \mathbf{f\,D_{k+3}}, \\ &\vdots \\ \mathbf{g_1R_n} &= g_n\mathbf{a} + \mathbf{f\,D_n}, \end{aligned} \tag{191}$$

where

$$\mathbf{R_j} = \begin{bmatrix} R_{0,j}^{q^N} & R_{0,j}^{q^{N-1}} & \ldots & R_{0,j}^{q^{N-k+1}} \\ R_{1,j}^{q^N} & R_{1,j}^{q^{N-1}} & \ldots & R_{1,j}^{q^{N-k+1}} \\ \vdots & \vdots & \vdots & \vdots \\ R_{k-1,j}^{q^N} & R_{k-1,j}^{q^{N-1}} & \ldots & R_{k-1,j}^{q^{N-k+1}} \end{bmatrix},$$

$$j = k+1, k+2, \ldots, n$$

are the known matrices, $\mathbf{g_1}=(g_1, g_2, \ldots, g_k)$, $\mathbf{a} = (1, 1, \ldots, 1)$ is a k-vector of 1's, and $\mathbf{D_j} = \mathrm{diag}[d_j^{q^N}, d_j^{q^{N-1}}, \ldots, d_j^{q^{N-k+1}}]$, $j = k+1, k+2, \ldots, n$, are diagonal matrices.

Solving the system (191) leads to breaking the PKC as mentioned above.

The solution proceeds in two stages. In the first stage, we obtain $\mathbf{g_1} = (g_1, g_2, \ldots, g_k)$ and $g_{k+1}, g_{k+2}, d_{k+1}, d_{k+2}, d_{k+3}$. In the second stage, we get the rest of the unknowns.

Stage 1

Consider the three first matrix equations of (191):

$$\mathbf{g_1 R_{k+1}} = g_{k+1}\mathbf{a} + \mathbf{f\,D_{k+1}}, \tag{192}$$

$$\mathbf{g_1 R_{k+2}} = g_{k+2}\mathbf{a} + \mathbf{f\,D_{k+2}}, \tag{193}$$

$$\mathbf{g_1 R_{k+3}} = g_{k+3}\mathbf{a} + \mathbf{f\,D_{k+3}}. \tag{194}$$

Assume without loss of generality that $g_{k+3} = 1$ (see Remark 3) and $d_{k+3} = 1$, $\mathbf{D_{k+3}} = \mathbf{E}$ (see Remark 4). It may happen that $d_{k+3} = 0$ in (188). Then we can find another non-zero d_j and put $d_j = 1$.

We can also *guess* a value of d_{k+2}, respectively, $\mathbf{D_{k+2}}$. If d_{k+2} is guessed correctly then the other unknowns can be obtained rather easily as follows.

Using (194) to eliminate $\mathbf{f}$ from (193) yields

$$\begin{aligned} \mathbf{g_1} &= g_{k+2}\mathbf{z} + \mathbf{w}, \\ \mathbf{f} &= g_{k+2}\mathbf{r} + \mathbf{s} \end{aligned} \tag{195}$$

where

$$\begin{aligned} \mathbf{z} &= \mathbf{a}\left(\mathbf{R_{k+2}} - \mathbf{R_{k+3}D_{k+2}}\right), \\ \mathbf{r} &= \mathbf{zR_{k+3}}, \\ \mathbf{w} &= -\mathbf{aD_{k+2}}, \\ \mathbf{s} &= \mathbf{wR_{k+3}} - \mathbf{a} \end{aligned}$$

are the known vectors (provided that the matrix $\mathbf{R_{k+2}} - \mathbf{R_{k+3}D_{k+2}}$ is non-singular.)

Using (195) to eliminate $\mathbf{f}$ and $\mathbf{g_1}$ from (192) yields

$$g_{k+2}\left(\mathbf{z} - \mathbf{rD_{k+1}}\right) = g_{k+1}\mathbf{a} + \mathbf{sD_{k+1}} + \mathbf{t}, \tag{196}$$

where $\mathbf{t} = -\mathbf{wR_{k+1}}$ is the known vector. Hence, we obtain a system of k equations in the three unkowns $g_{k+1}, g_{k+2}, d_{k+1}$. Since this system is a linear one with respect to the unknowns g_{k+1}, g_{k+2}, it has a solution if the k-vectors

$$\begin{aligned} &\mathbf{a}, \\ &\mathbf{z} - \mathbf{rD_{k+1}}, \\ &\mathbf{sD_{k+1}} + \mathbf{t} \end{aligned}$$

are *linearly dependent*. This means that all the successive 3-determinants must be equal to zero. For instance, the first of such a determinant has the form

$$\begin{vmatrix} 1 & 1 & 1 \\ z_1 - r_1 d_{k+1}^{q^N} & z_2 - r_2 d_{k+1}^{q^{N-1}} & z_3 - r_3 d_{k+1}^{q^{N-2}} \\ t_1 + s_1 d_{k+1}^{q^N} & t_2 + s_2 d_{k+1}^{q^{N-1}} & t_3 + s_3 d_{k+1}^{q^{N-2}} \end{vmatrix} = 0. \tag{197}$$

If we denote $y := d_{k+1}^{q^{N-2}}$, we obtain from eq. 197 the first algebraic equation of degree $q^2 + q$ in the unknown y. Similarly, the second determinant is

$$\begin{vmatrix} 1 & 1 & 1 \\ z_2 - r_2 d_{k+1}^{q^{N-1}} & z_3 - r_3 d_{k+1}^{q^{N-2}} & z_4 - r_4 d_{k+1}^{q^{N-3}} \\ t_2 + s_2 d_{k+1}^{q^{N-1}} & t_3 + s_3 d_{k+1}^{q^{N-2}} & t_4 + s_4 d_{k+1}^{q^{N-3}} \end{vmatrix} = 0. \qquad (198)$$

If we raise this equation to the power q, we can obtain a second equation of degree $q^2 + q$ in the unknown y, and so on. Hence, we get the system of $k - 2$ equations of degree $q^2 + q$ in the only unknown y. One can use the Euclidean division algorithm or another method, to solve this system. If $y = d_{k+1}^{q^{N-2}}$ is a solution, we get the unknowns g_{k+1}, g_{k+2} from eq. 196.

Gibson's procedure includes the *guessing* of a value of d_{k+2}, the solving of linear equations and the solving of polynomial equations of degree $q^2 + q$. In the binary case, $q = 2$, the complexity of his algorithm is at worst $O(n^6 2^n)$.

Stage 2

Now, consider the fourth equation of the system (191):

$$\mathbf{g_1 R_{k+4}} = g_{k+4}\mathbf{a} + \mathbf{f\, D_{k+4}}. \qquad (199)$$

Since $\mathbf{g_1}$ and $\mathbf{f}$ are already known from **Stage 1**, we can get from eq. 199 the unknowns g_{k+4} and d_{k+4} using the same method as for getting d_{k+1} at **Stage 1**. In a similar manner, we obtain the remaining unknowns $g_j, d_j,\ j = k + 5, \ldots, n$.

In general, the number of calculations to solve the system (191) increases exponentially with length n. *But for the practical range* $20 \le n \le 30$, *the Gibson attack breaks the GPT PKC with a hiding matrix* $\mathbf{X}$ *of rank* 1 *over* $GF(q)$.

5.3. How to Avoid the Gibson Attack. The only modification needed is more a careful choice of the hiding matrix $\mathbf{X}$. Generally, a matrix $\mathbf{X}$ must be of rank t_1 over the *base* field $GF(q)$ but it may be of rank 1 to t_1 over the extended field $GF(q^N)$. Recall that t_1 is a *design* parameter of the PKC.

Now, we have to choose this matrix having rank t_1 also over the extended field $GF(q^N)$. Namely let $\mathbf{X} = \mathbf{AB}$, where $\mathbf{A}$ is a $k \times t_1$ matrix of full rank t_1 over $GF(q^N)$, and $\mathbf{B}$ is a $t_1 \times n$ matrix of full rank *both* over the extended field $GF(q^N)$ *and* over the base field $GF(q)$.

It is clear that the matrix $\mathbf{X}$ also is a matrix of full rank t_1 *both* over the extended field $GF(q^N)$ *and* over the base field $GF(q)$.

The matrix $\mathbf{B}$ can be constructed as follows. First, choose a matrix $\mathbf{B}_0 = [\mathbf{Q}\ \mathbf{O}]$, where $\mathbf{Q}$ is a nonsingular square matrix of order t_1 with entries in $GF(q^N)$, and $\mathbf{O}$ is a $t_1 \times (n - t_1)$ matrix of 0's. Then multiply $\mathbf{B}_0$ to the right by an $n \times n$ nonsingular matrix $\mathbf{P}$ with entries in the *base* field $GF(p)$.

Gibson's approach can be also applied to this case.

By definition, put $\mathbf{C} = [C_{ij}] = \mathbf{S^{-1}A}$, $\mathbf{F} = [f_{ij}] = [C_{i,j}^{q^{N-i}}]$. Put $\mathbf{B} = [\mathbf{B_1}\ \mathbf{B_2}]$, where $\mathbf{B_1}$ consists of the first k columns of $\mathbf{B}$. Put

$$\mathbf{d} = [d_{i,j}] = \mathbf{B_2} - \mathbf{B_1 R},$$

$$i = 1, 2, \ldots, t_1;\ j = k+1, k+2, \ldots, n.$$

Then from eq. 186 one can obtain, after some manipulations, the following system of equations

$$\begin{gathered} g_1 R_{0,j}^{q^{N-i}} + g_2 R_{1,j}^{q^{N-i}} + \ldots + g_k R_{k-1,j}^{q^{N-i}} = g_j + \textstyle\sum_{s=1}^{t_1} f_{is} d_{s,j}^{q^{N-i}}, \\ i = 0, 1, \ldots, k-1;\ j = k+1, k+2, \ldots, n, \end{gathered} \tag{200}$$

where g_i, f_{is}, d_{sj} are the unknowns. In the matrix notations, we have

$$\begin{aligned} \mathbf{g_1 R_{k+1}} &= g_{k+1}\mathbf{a} + \sum_{s=1}^{t_1} \mathbf{F}_s\, \mathbf{D}_{s,\mathbf{k+1}}, \\ \mathbf{g_1 R_{k+2}} &= g_{k+2}\mathbf{a} + \sum_{s=1}^{t_1} \mathbf{F}_s\, \mathbf{D}_{s,\mathbf{k+2}}, \\ \mathbf{g_1 R_{k+3}} &= g_{k+3}\mathbf{a} + \sum_{s=1}^{t_1} \mathbf{F}_s\, \mathbf{D}_{s,\mathbf{k+3}}, \\ &\vdots \\ \mathbf{g_1 R_n} &= g_n\mathbf{a} + \sum_{s=1}^{t_1} \mathbf{F}_s \mathbf{D}_{s,\mathbf{n}}, \end{aligned} \tag{201}$$

where $\mathbf{D}_{s,\mathbf{j}} = \mathrm{diag}[d_{s,j}^{q^N}, d_{s,j}^{q^{N-1}}, \ldots, d_{s,j}^{q^{N-k+1}}]$, $j = k+1, k+2, \ldots, n$, and $\mathbf{F}_s = (f_{0,s}, f_{1,s}, \ldots, f_{k-1,s})$.

Again, one can assume without loss of generality that $g_{k+3} = 1$ and that $\mathbf{D}_{s,\mathbf{k+3}} = \mathrm{diag}[1, 1, \ldots, 1]$, $s = 1, 2, \ldots, t_1$. But to solve eq. 201, one has to *guess* the values of t_1 the unknowns $\mathbf{D}_{s,\mathbf{k+2}}$, $s = 1, 2, \ldots, t_1$. In this case, Gibson's procedure includes the solving of linear equations and the solving of systems of polynomial equations each in t_1 variables of degree $q^{t_1+1} + q^{t_1} + \ldots + q$. In the binary case, $q = 2$, the complexity of his algorithm is at least $O(n^6 2^{t_1 n})$. This seems to be infeasible even for $t_1 = 2$ and for the practical range $20 \leq n \leq 30$.

6. Comparison of the Three Public-Key Cryptosystems

For purpose of comparison. we choose the McEliece example and examples of the Niederreiter PKC and the GPT PKC with comparable performance.

PKC	**Parameters**	**Size of Public keys (bits)**	**Workfunction**
McEliece	binary, $n = 1024$, $k = 524,\ t = 50$	Large: 5×10^5	$> 2^{59}$
Niederreiter	q-ary, $q = n = 128$ $d = 64,\ r = 63$	Reasonable: $32,000$	Poor: $O(n^3)$
Niederreiter modified	The same	The same	$> 2^{75}$
GPT	$q = 2^{20},\ n = 20,$ $d = 9,\ k = 12,\ t_1 = 1$	Good: 4800	Poor: $O(n^6 2^n)$ $> 2^{46}$
GPT modified 1	$q = 2^{20},\ n = 20,$ $d = 9,\ k = 12,\ t_1 = 2$	The same	Good: $O(n^6 2^{2n})$ $> 2^{66}$
GPT modified 2	$q = 2^{20},\ n = 20,$ $d = 11,\ k = 10,\ t_1 = 3$	The same	Good: $O(n^6 2^{3n})$ $> 2^{86}$

The McEliece PKC seems to be secure even without any modification. The reason is that Goppa codes are subfield subcodes over $GF(2)$ and there exist too many GRS codes containig them as subcodes. Thus, there is no evident way to find a Goppa polynomial or a suitable GRS code from a scrambled generator matrix. A modification of the McEliece cryptosystem is useful if the breaking party has extra information on the ordering $\overline{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$. A disadvantage of this PKC is the large size of the Public ley.

The Niederreiter and GPT PKC's are defined over large alphabets. The weakness of such a kind of PKC is due the very regular structure of the generator or parity check matrices, even if scrambled . A hiding of the public keys by means of adding carefully chosen matrices prevents known attacks and provides security of this PKC.

The GPT PKC based on rank codes seems to be the best one due to the relatively small size of the public key. However, it is large compared to the ones used in the Rivest-Shamir-Adleman PKC [3]. Nevertheless, the GPT PKC can be used in *practical applications* because of the very easy procedure of key generation.

7. Conclusion

The security of the public-key cryptosystems based on linear codes depends on a few open problems. Only if these have been solved, a final conclusion will be possible. These problems are listed below.

Main problem

Let $\mathcal{C}$ be a general linear (n, k, d)-code of lenght n, dimension k and Hamming distance d.

Problem: For the given n-vector $\mathbf{y}$, find a code vector $\mathbf{g} \in \mathcal{C}$ such that

$$d_H(\mathbf{y}, \mathbf{g}) \le t = \left\lfloor \frac{d-1}{2} \right\rfloor .$$

If this problem belongs to the class P, then the McEliece- and the Niederreiter PKC would be insecure.

It is known, that the **problem** "Find a code vector $\mathbf{g}$ such that $d_H(\mathbf{y}, \mathbf{g}) = \min$" is hard for the Hamming metric [5].

Breaking the McEliece PKC

Let $\mathbf{G}$ be a binary generator matrix of a Goppa code of length $n = 2^m$ and let $g(x)$ be the corresponding Goppa polynomial of degree t. This means that each row of the generator matrix satisfies the condition

$$\sum_{s=1}^{n} \frac{g_s}{x - \alpha_s} \equiv 0 \mod g(x),$$

for some ordering

$$\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$$

of the elements of the field $GF(2^m)$.

Problem: Using $\mathbf{G}$, find the ordering $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ and the polynomial $g(x)$.

Breaking the Niederreiter PKC

Let the field $GF(q)$ be given. Consider the system of algebraic equations in the n unknowns $\lambda_1, \lambda_2, \ldots, \lambda_k, \nu_1, \nu_2, \ldots, \nu_{n-k}$:

$$\det \begin{bmatrix} \frac{1}{S_{i_1,j_1} - \lambda_{i_1}\nu_{j_1}} & \frac{1}{S_{i_1,j_2} - \lambda_{i_1}\nu_{j_2}} & \frac{1}{S_{i_1,j_3} - \lambda_{i_1}\nu_{j_3}} \\ \frac{1}{S_{i_2,j_1} - \lambda_{i_2}\nu_{j_1}} & \frac{1}{S_{i_2,j_2} - \lambda_{i_2}\nu_{j_2}} & \frac{1}{S_{i_2,j_3} - \lambda_{i_2}\nu_{j_3}} \\ \frac{1}{S_{i_3,j_1} - \lambda_{i_3}\nu_{j_1}} & \frac{1}{S_{i_3,j_2} - \lambda_{i_3}\nu_{j_2}} & \frac{1}{S_{i_3,j_3} - \lambda_{i_3}\nu_{j_3}} \end{bmatrix} = 0, \tag{202}$$

$$1 \le i_1 < i_2 < i_3 \le k; \; 1 \le j_1 < j_2 < j_3 \le n - k$$

Problem: For the given $S_{i,j}$, $1 \le i \le k$, $1 \le j \le n-k$, find a solution of this system.

Breaking the GPT PKC

Let the field $GF(q)$ be given. Consider the system of algebraic equations in the $n + kt_1 + (n-k)\,t_1 = n\,(1+t_1)$ unknowns g_i, f_{is}, d_{sj}

$$g_1 R_{0,j}^{q^{N-i}} + g_2 R_{1,j}^{q^{N-i}} + \ldots + g_k R_{k-1,j}^{q^{N-i}} = g_j + \textstyle\sum_{s=1}^{t_1} f_{is} d_{s\ell,j}^{q^{N-i}},$$

$$i = 0, 1, \ldots, k-1; \;\; j = k+1, k+2, \ldots, n.$$

Problem: For the given $R_{i,j}$, $0 \le i \le k-1$, $k+1 \le j \le n$, find a solution of this system.

References

[1] W. Diffie and M.E. Hellman, "New Directions in Cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, 6, pp. 644-654, 1976.

[2] R.C. Merkle and M.E. Hellman, "Hiding Information and Signatures in Trapdoor Knapsacks," *IEEE Trans. Inform. Theory*, vol. IT-24, 5, pp. 525-530, 1978.

[3] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM*, vol. 21, Feb. 1978.

[4] R. J. McEliece, "A Public Key Cryptosystem Based on Algebraic Coding Theory," *JPL DSN Progress Rep. 42-44*, pp. 114-116, Jan.-Feb. 1978.

[5] E.R. Berlekamp, R.J. McEliece, and H.C.A. van Tilborg, "On inherent intractability of certain coding problems," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 384-386, 1978.

[6] R. Heiman and A. Shamir, "On the Security of Cryptosystems Based on Linear Error Correcting Codes," *Applied Mathematics*, Weizmann Institute of Science, Rehovot, Israel, 1987.

[7] J. K. Gibson, "Equivalent Goppa Codes and Trapdoors to McEliece's Public Key Cryptosystem," pp.517-521 in *Advances in Cryptology* - EUROCRYPT'91 (Ed. D.W. Davies), Lecture Notes in Computer Science No. 547. Berlin and Heidelberg: Springer, 1991.

[8] H. Niederreiter, "Knapsack-Type Cryptosystem and Algebraic Coding Theory," *Probl.Control and Inform.Theory*, vol.15, pp.19-34, 1986.

[9] V. M. Sidelnikov, S. O. Shestakov, "On the Insecurity of Cryptosystems Based on Generalized Reed-Solomon Codes," *Discrete Math* ., vol. 1, no. 4, pp. 439-444, 1992.

[10] V. M. Sidelnikov, S. O. Shestakov, "On Cryptosystems Based on Generalized Reed-Solomon Codes," in: *Prospective Methods for Telecommunication and Integrated Communication Systems,* (Ed. V. V. Zyablov), pp. 48-51, Moscow, 1992 (in Russian.)

[11] E. M. Gabidulin, "Theory of Codes with Maximum Rank Distance," *Probl. Inform. Transm.*, vol.21, No. 1, pp.1-12, July 1985.

[12] E. M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov, "Ideals over a Non-Commutative Ring and their Application in Cryptology," pp. 482-489, in: *Advances in Cryptology - EUROCRYPT'91* (Ed. D.W. Davies), Lecture Notes in Computer Science No. 547. Berlin and Heidelberg: Springer, 1991.

[13] J. K. Gibson, "Severely Denting the Gabidulin Version of the McEliece Public Key Cryptosystem," submitted to *Designs, Codes, and Cryptography*, 1993.

[14] D. Lazard, "Resolution des systems d'equations algebriques," *Theor. Comp. Sci.*, vol. 15, pp. 77-110, 1981.

[15] N.J. Patterson, "The Algebraic Decoding of Goppa Codes," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 203-207, 1975.

[16] D.V. Sarwate, "On the Complexity of Decoding Goppa codes," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 515-516, 1977.

[17] F. J. MacWilliams, N. J. A. Sloane, "The Theory of Error-Correcting Codes," Amsterdam, North-Holland, 1977.

[18] C. M. Adams, H. Meijer, "Security-Related Comments Regarding McEliece Public Key Cryptosystem." In *Advances in Cryptology - EUROCRYPT'87.*

[19] R. I. Lee, E. F. Brickell, "An Observation of the McEliece Public Key Cryptosystem." In *Advances in Cryptology - EUROCRYPT'88.*

[20] E. A. Kruk, "Bounds for Decoding Complexity of Any Linear Block Code," *Probl. Inform. Transm.*, vol.25, No. 3, pp.103-107, 1989.

[21] E. A. Kruk, "Code Based Public Key Cryptosystems," in: *Prospective Methods for Telecommunication and Integrated Communication Systems,* (Ed. V. V. Zyablov), pp. 62-69, Moscow, 1992 (in Russian.)

[22] E.R. Berlekamp, "Algebraic coding theory," McGraw-Hill, New York, 1968.

[23] E. M. Gabidulin, "A Fast Matrix Decoding Algorithm For Rank-Error-Correcting Codes." In G. Cohen, S. Litsyn, A. Lobstein, G. Zemor),*Algebraic coding* , pp. 126-132, Lecture Notes in Computer Science No. 573, Springer-Verlag, Berlin, 1992.