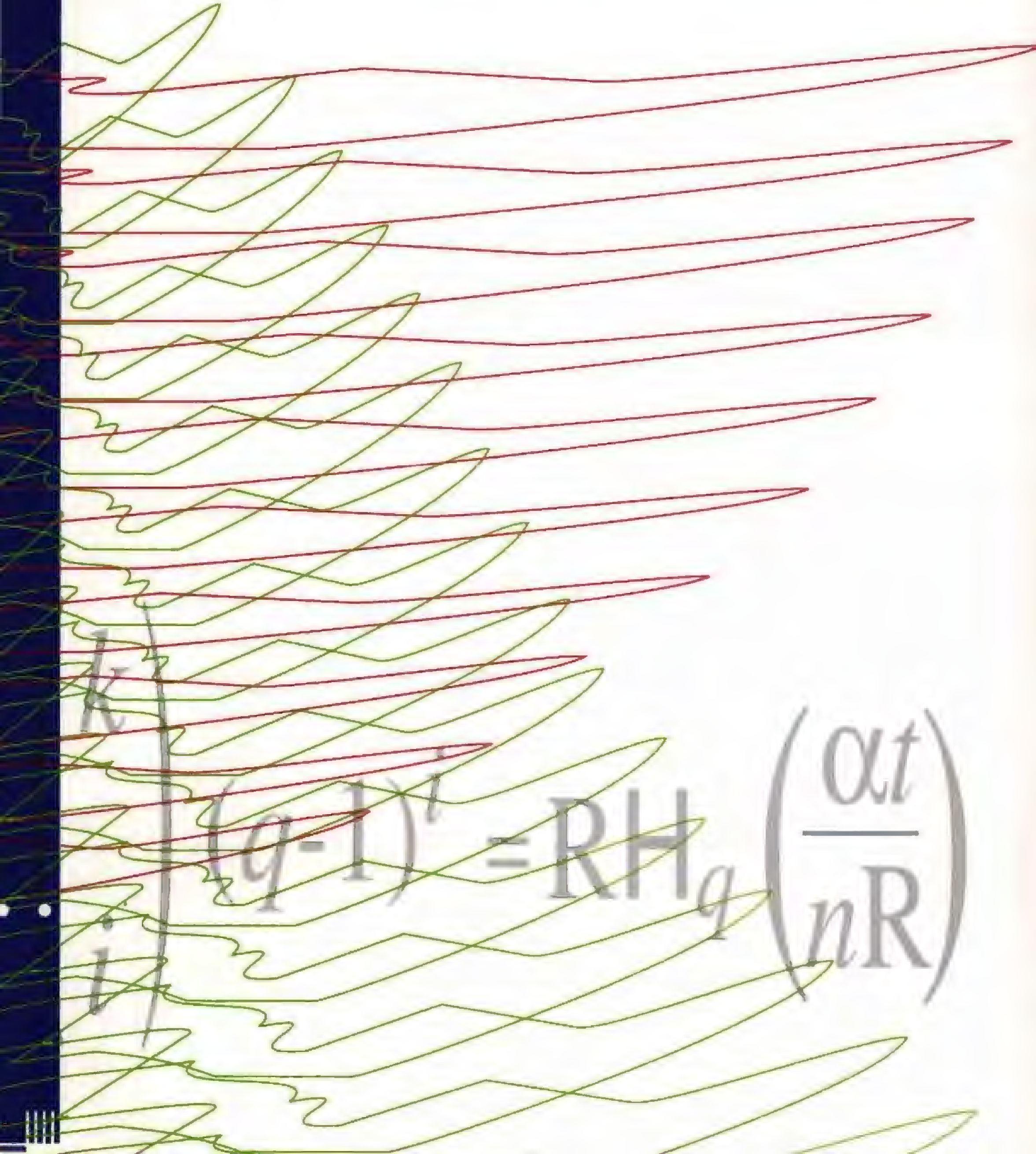
Security-analysis of a class of cryptosystems based on linear error-correcting codes

DOI:
10.6100/IR426904

Johan van Tilburg

Security-Analysis of a Class of Cryptosystems Based on Linear Error-Correcting Codes


$\binom{k}{i}(q-1)^i = RH_q\left(\frac{\alpha t}{nR}\right)$

Security-Analysis of a Class of Cryptosystems Based on Linear Error-Correcting Codes

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Tilburg, Johan van

Security-analysis of a class of cryptosystems based on
linear error-correcting codes / Johan van Tilburg. -
Leidschendam : PTT Research. - Ill.
Proefschrift Eindhoven. - Met lit. opg., reg.
ISBN 90-72125-45-2
Trefw.: geheimschrift / coderingstheorie.

Security-Analysis of a Class of Cryptosystems Based on Linear Error-Correcting Codes

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de Rector Magnificus, prof.dr. J.H. van Lint, voor een commissie aangewezen door het College van Dekanen in het openbaar te verdedigen op dinsdag 29 november 1994 om 16.00 uur

door

Johan van Tilburg

geboren te Voorburg (Z-H)

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr.ir. H.C.A. van Tilborg

en

prof.dr. A.E. Brouwer

To my Mother...

Abstract

For arbitrarily given linear encoding schemes, this dissertation describes a new universal probabilistic decoding algorithm. To date, it is the most efficient one in its class for small and moderate code parameter values. As proved herein, its decoding complexity is equal to Information Set Decoding. The work-factor, memory-factor and complexity of this new algorithm are evaluated and then its overall performance is compared to other general decoding algorithms. Furthermore, the new algorithm is rewritten as a syndrome decoding algorithm and then extended to simultaneously process several syndromes in a batch. Finally, a slight structure modification of the batch description yields a syndrome decoding algorithm with several simple, dedicated circuits.

This dissertation analyzes locally-randomized cryptosystems which make use of linear encoding schemes. In this class of cryptosystems, the message blocks are encoded into codewords and then locally randomized using random error patterns. This dissertation focuses on the McEliece locally-randomized public-key cryptosystem by first giving a detailed compilation and analysis of relevant literature describing its security. Then, this system's relation to the Niederreiter and the Stern public-key cryptosystems are briefly discussed. It is shown that the new decoding algorithms determine the range of values of the code parameters for which these locally-randomized public-key cryptosystems' security is vulnerable to cryptanalysis. Three insecure and related digital signature schemes are also identified.

In contrast to the McEliece public-key cryptosystem, the Rao-Nam and Li-Wang locally-randomized secret-key cryptosystems keep their linear encoding scheme secret. It is believed that this secrecy allows simpler and smaller codes to be used, which require less storage and enable faster processing. However, this dissertation shows that for this class of secret-key cryptosystems an equivalent encoding scheme can be obtained in an efficient way. Hence, it is concluded that the code parameter values for locally-randomized secret-key cryptosystems should be as large as those for locally-randomized public-key cryptosystems.

Acknowledgments

As I finish this Ph.D. dissertation, it is a pleasure to thank many people for their inspiration, contributions and assistance.

Special thanks go to Henk van Tilborg with whom I have had a fruitful corporation. His input and constructive feedback has been invaluable. I thank Dick Boekee for teaching me the basics of Information Theory along with many practical aspects of research and for introducing me to Cryptology. I also thank the other two members of the advisory committee: Andries Brouwer and Joos Vandewalle. I am particularly grateful to Paul Hin, René Struik, Joost Meijers, Raymond Doyen and Peter Roelse for their active participation, important contributions, and most of all, their friendship.

This work would not have been possible without the support of PTT Research. I wish to express my gratitude to Ruud Grünewald, Derk van der Houwen and Kees Stok for the many opportunities I have received to broaden my knowledge and experience. My appreciation also goes to Patrick Morley, Department Head, who has given me the flexibility to complete this work as scheduled. I thank my colleagues for their support and enthusiasm, especially Jean-Paul Boly, Johannes Braams, Andries Hekstra, Frank Muller, Gert Roelofsen and in particular Peter de Rooij. The PTT Research support staff has been extremely helpful, and a special thanks goes to the library and the photographic department for their dedicated service.

Finally, I am highly indebted to my wife Joie Rose for editing this work and for her patience and support. As I complete these acknowledgments, she affectionately reminds me of one slight problem—I am running out of excuses!

Again, to all a heartfelt thanks.

Leidschendam, September 1994 *Johan van Tilburg*

Contents

Chapter 1

Introduction

Cryptology, the science of secret writing, involves both *cryptography* and *cryptanalysis*. Cryptography relates to the science of designing cryptosystems, while cryptanalysis corresponds to the *security analysis* of cryptosystems.

Historical manuscripts confirm that cryptology has its origins among the Arabs [AK92]. In fact, ninth-century Arab scientist al-Kindī authored the oldest found study of cryptology. That manuscript reported that his predecessor al-Khalīl (718–786) had written *Kitāb al-Muʿammā* (*Book of Riddles*) about a century earlier. Unfortunately, the book was never found.

Cryptology has gone through many changes in its history. Its evolution is highlighted in such references as [Kah67], a historic overview; [DH76], an introduction to public-key cryptography; and [Sim92], a review of contemporary cryptology. The most recent change has been instigated by the advancement of computers. Because digital equipment is now affordable and fashionable, it is proliferating at an accelerating pace. With the digitization of information by means of analog-to-digital converters, the integration of voice, image and data is transforming voice-only transmissions into multi-media telecommunication networks. Based on fiber optics, and augmented by satellite and cellular transmission, the new telecommunication infrastructure resembles an electronic highway. The result is an international, integrated, information-processing industry based on digital technology, and the use of this advancement is becoming commonplace. Electronic mail is replacing traditional mail; electronic money is more and more replacing plastic money, which in turn is rapidly replacing paper money, and so forth. This growth of public data channels and the accessibility of databases has spawned an increased public demand for the type of security that only cryptology can provide. For more information on cryptology, the reader is referred to books on cryptology, such as [BP82, Den82, MM82, DP84, Kra86, Rue86, Kob88, Bra88, Til88, Men93, BPV94, Sch94] and [Sti95].

1.1 Cryptosystems

In the field of security, a *classical cryptosystem* conceals the *plaintext* by *encrypting* it into a *ciphertext* using a specific parameter called the *encryption key.* Any person who has the *decryption key* can decrypt the ciphertext into the original plaintext. However, in a *secret-key cryptosystem* both keys, which may be the same, are kept secret. In a *public-key cryptosystem*, one key is kept private while the other one is made public. This cryptosystem's security is based on the assumption that it is infeasible to deduce the *private key* from the *public key.*

In the mid-seventies, the invention of public-key cryptography by Diffie and Hellman [DH76] and, independently by Merkle [Mer78], provided a first solution to the needs of security in a modern information society. During the past two decades, this research has provided a catalyst for the explosive growth of public activity in modern cryptology and has led to a substantial change in its focus. The main realization has been that *privacy* and *authentication* are independent concepts. Privacy refers to secrecy and holds an important position in information security. Authentication, on the other hand, determines if a part of a message has been transmitted by the authorized sender and if it has been substituted, or altered. Authentication requirements are becoming much more sophisticated. For example, in a *zero-knowledge* proof system each party possesses some information they wish to keep private. The first party, *the prover*, convinces a second party, *the verifier*, that an assertion is valid without disclosing any private information. For details, the reader is referred to [Sim92].

Once a cryptosystem is designed, its security must be analyzed. A standard method of cryptanalysis reduces the description of the cryptosystem into a series of mathematical problems. For example, it may be proved that the security of the system relies on such difficult theoretical problems as factoring composite integers, taking discrete logarithms in finite fields and decoding random, linear codes. When such mathematical problems are shown to be easily solved, the cryptosystem is insecure. Through cryptanalysis, many cryptosystems are revealed to be vulnerable in this way.

1.2 Randomness

Just as cryptography has inspired more sophisticated cryptanalysis, cryptanalysis has inspired the design of more effective (secure) cryptosystems. A key-factor in this design is *randomness*, which relies on the following principle: The outcome (zero or one) of a *random* toss of a coin added (modulo 2) to the outcome of a *not random* (biased) toss results in a *random outcome.* Using this concept, Vernam [Ver26] invented a cryptosystem which adds (modulo 2) a random sequence of key bits to a structured sequence of message bits in order to yield a random sequence of crypto bits. In essence, the random sequence *globally randomizes* the message sequence. Shannon [Sha49] proved that this method hides the message *perfectly* if the random sequence of key bits is used only once and if it is at least as long as the sequence of message bits. This is why Vernam's original cryptosystem is commonly referred to as the *one-time pad.* The benefit of this cryptosystem is that even a cryptanalyst with unlimited resources will lack enough information to decipher the crypto sequence. On the other hand, if the cryptanalyst has limited computational resources, *conditional perfect security* may be sufficient and it can be obtained with a smaller sequence of random bits as shown by Maurer [Mau90].

This dissertation focuses on adding *local randomness* to the message by using a linear error-correcting code as follows. First, divide the message into short sequences of bits called *message blocks.* Then, transform each message block into a code block or *codeword* by using an *error-correcting code.* Next, randomly choose an *error pattern* (a block of error bits) from a set of correctable error patterns and add it to the codeword. The result is an encrypted sequence of encoded message bits. In essence, the original message is divided into blocks which are encoded into codewords and then *locally randomized* using random error patterns. The result is a *locally-randomized cryptosystem.* An authorized recipient with knowledge of the corresponding decoding algorithm can find the error pattern and recover the original message. Without this knowledge, an unauthorized recipient must solve a more complex, and hopefully unsolvable, decoding problem to recover the same message. In other words, the more complex the problem, the more secure the system.

1.3 Scope of Dissertation

This dissertation analyzes a class of locally-randomized cryptosystems based on linear error-correcting codes. The main objective is to devise, for arbitrarily given linear encoding schemes, a probabilistic decoding algorithm that is efficient for code lengths with small to moderate values. To evaluate the merits of this new algorithm its performance is compared to other general decoding algorithms using the following methodology:

- establish the performance and decoding complexity of the new algorithm as a benchmark for comparison;
- standardize the mathematical description of each relevant general decoding algorithm as it applies to cryptanalysis;
- provide a measure to evaluate the values of the security parameters of cryptosystems based on the decoding problem for linear error-correcting codes;
- compare the performance and decoding complexity of the new algorithm to that of the other algorithms.

As a result, a new probabilistic, general decoding algorithm is described which is, to date, the most efficient in its class. It is then proved that its decoding complexity is equal to the decoding complexity of the information set decoding method when applied to random codes.

For locally-randomized public-key cryptosystems, this new decoding algorithm determines the range of values for the code parameters for which the cryptosystem's security is vulnerable to cryptanalysis. In contrast to public-key cryptosystems, locally-randomized secret-key cryptosystems keep the linear encoding scheme secret. It is believed that this secrecy allows simpler and smaller codes to be used that require less storage and enable processing at higher speeds. This dissertation shows, however, that for this type of secret-key cryptosystems an equivalent encoding scheme can be obtained in an efficient way. Once it is obtained, the corresponding decoding problem can not be more difficult to solve than the decoding problem for a public-key cryptosystem. Otherwise, this equivalent encoding scheme would define a more efficient locally-randomized public-key cryptosystem which therefore contradicts the assumption that smaller codes can be effectively used in locally-randomized secret-key cryptosystems. In summary, the code parameter values for locally-randomized secret-key cryptosystems should be as large as those for locally-randomized public-key schemes.

This dissertation is structured as follows:

Chapter 2 Codes, Complexity and Decoding provides necessary background information for locally-randomized cryptosystems . It gives a brief overview of the theory of linear error-correcting codes and explains complexity theory terminology. The general decoding problem for linear error-correcting codes is described along with three types of (hard-decision) decoding techniques.

Chapter 3 A Class of Public-Key Cryptosystems focuses on the McEliece system, which is based on linear error-correcting codes. It includes a detailed compilation and analysis of relevant literature describing the security of the McEliece cryptosystem. In relation to this cryptosystem, the Niederreiter and the Stern public-key cryptosystems are also briefly discussed. All descriptions include a security discussion with respect to the decoding problems identified in Chapter 2. In conclusion three insecure, yet related, digital signature schemes are identified.

Chapter 4 Decoding of Random Linear Codes presents several techniques for solving the decoding problems identified in Chapter 2. In order to compare their merits and performances, they are described and analyzed in a standardized way. In particular, information set decoding is discussed in order to form a foundation for the decoding algorithms introduced in Chapter 5.

Chapter 5 Permutation Decoding discusses the class of probabilistic decoding algorithms as it appeared in a security analysis of the McEliece public-key cryptosystem. This chapter describes the implementation of three different algorithms in this class and then shows that their decoding complexity is equal to information set decoding.

Chapter 6 Syndrome Decoding takes the Permutation Decoding algorithm and rewrites it as a Syndrome Decoding algorithm. The work-factor of this new decoding algorithm is then established. Next, the algorithm is extended to simultaneously process several syndromes in a batch. Finally, a slight structure modification of the batch description yields a syndrome decoding algorithm with several simple, dedicated circuits (in parallel).

Chapter 7 Soft-Decision Decoding examines a channel-measurement decoding algorithm for random linear error-correcting codes. For locally-randomized cryptosystems soft-decision algorithms are important when the security of the system relies on the errors introduced by the channel. This chapter proves that the complexity of these decoding algorithms equals that of the underlying hard-decision decoding problem as discussed in Chapters 2 and 4.

Chapter 8 A Class of Secret-Key Cryptosystems discusses a secret-key variant of the McEliece public-key cryptosystem and then shows how it can be vulnerable to a majority voting analysis later refined in Chapter 9. Chapter 8 further shows that several algorithms exist that can obtain the encoding scheme in the Rao-Nam secret-key cryptosystem. Next, the Li-Wang joint secret-key and authentication cryptosystem is shown to be even less secure than the Rao-Nam scheme.

Chapter 9 Beyond Majority Voting gives three extensions of majority voting. In order to obtain local majority votes, the first simultaneously considers more than one coordinate. The second goes one step further and aligns these local majority votes and obtains global majority votes. The third uses the global majority votes to yield an equivalent encoding scheme which is then applied to some of the private-key cryptosystems discussed in Chapter 8.

Chapter 2

Codes, Complexity and Decoding

This chapter introduces the necessary background information needed for locally-randomized cryptosystems. First, Section 2.1 establishes an important relationship between the q-ary entropy function and the volume of a sphere. Section 2.2 gives a brief overview of the theory of linear codes. Section 2.3 explains complexity theory terminology. Section 2.4 defines a class of decoding problems. The relationship between these problems and locally-randomized cryptosystems is later discussed in Chapter 3.

2.1 Preliminaries

Before discussing the coding theory of linear codes, it is important to define two symbols often called *Landau symbols.* These symbols are practical when determining the *asymptotical* complexity of decoding algorithms, and useful in establishing a relationship between the q-ary entropy function and the volume of a sphere.

For two arbitrary real-valued functions f and g, the O-symbol notation $f(n) = O(g(n))$ when $n \to \infty$, means that, a value $\mathsf{K} > 0$ exists, such that, $|f(n)| \leq \mathsf{K}|g(n)|$ for sufficiently large n. Suppose that none of the subsequences of the $g(n)$ sequence tends to zero. Then the notation $f(n) = o(g(n))$ with $n \to \infty$ means that $\lim_{n\to\infty} f(n)/g(n) = 0$.

An important relationship between the O- and o-symbols is that $O(n^l)$ can be written as $q^{o(n)}$ for all integers l and $q > 1$.

Suppose that $\mathcal{Q}$ is an alphabet with q distinct symbols, and let $\mathcal{Q}^n$ denote the set of $\mathcal{Q}$-ary n-tuples. A ***block code*** C of length n is a non-empty subset of $\mathcal{Q}^n$, and an n-tuple of this subset is called a *codeword.* For every $\underline{x} := (x_1, x_2, \ldots, x_n)$ and $\underline{y} := (y_1, y_2, \ldots, y_n)$ in $\mathcal{Q}^n$, the *Hamming distance* $\mathsf{d}_{\mathrm{H}}(\underline{x}, \underline{y})$ equals the number of coordinates where $\underline{x}$ and $\underline{y}$ differ:

$$\mathsf{d}_{\mathrm{H}}(\underline{x}, \underline{y}) := |\{1 \leq i \leq n \mid x_i \neq y_i\}|.$$

The Hamming weight $\mathsf{w}_{\mathrm{H}}(\underline{x})$ equals the number of nonzero coordinates of $\underline{x}$ (i.e., $\mathsf{w}_{\mathrm{H}}(\underline{x}) = \mathsf{d}_{\mathrm{H}}(\underline{x}, \underline{0})$ where $\underline{0} := (0, 0, \ldots, 0) \in \mathcal{Q}^n$).

Let the q-ary function $\mathsf{H}_q(x)$ with $0 \leq x \leq 1$ be defined as

$$\begin{cases} 0, & \text{if } x = 0; \\ -x \log_q x - (1-x)\log_q(1-x) + x \log_q(q-1), & \text{if } 0 < x < 1; \\ \log_q(q-1), & \text{if } x = 1. \end{cases}$$

The function H_q is $\cap$-convex and invertible when restricted to $[0, 1 - \frac{1}{q}]$. Denote the inverse by H_q^{-1}. The function $\mathsf{H}_2(x)$ is often denoted by the binary entropy function $\mathsf{h}(x)$. For the q-ary alphabet, a function related to $\mathsf{h}(x)$ is $\widehat{\mathsf{H}}_q = \mathsf{h}(x)/\log_2 q$.

For any $r \in \mathbb{N}$ and $\underline{z} \in \mathcal{Q}^n$, the sphere $\mathsf{S}_r(\underline{z})$ with the radius r around $\underline{z}$ is given by

$$\mathsf{S}_r(\underline{z}) := \{\underline{y} \in \mathcal{Q}^n \mid \mathsf{d}_{\mathrm{H}}(\underline{y}, \underline{z}) \leq r\}.$$

The number of vectors contained in the volume of the sphere $\mathsf{S}_r(\underline{z})$ is

$$|\mathsf{S}_r(\underline{z})| = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i.$$

A relationship between the q-ary entropy function and the volume (cardinality) of a sphere with a radius $\lfloor \lambda n \rfloor$ is given by the following lemma:

Lemma 2.1.1 *Let $q > 1$ be an integer, and let $0 \leq \lambda \leq (q-1)/q$. Then*

$$\sum_{i=0}^{\lfloor \lambda n \rfloor} \binom{n}{i} (q-1)^i = q^{nH_q(\lambda) + o(n)}, \quad (n \to \infty). \tag{2.1}$$

This lemma, as proved in [Lin82], is used throughout this dissertation.

2.2 A Class of Codes

In locally-randomized cryptosystems, an important class of codes is that of *linear block codes.* These codes have simple encoding schemes and can be implemented efficiently. To construct these codes, let the alphabet $\mathcal{Q}$ be the Galois Field GF(q) where $q = p^r$ with p prime (i.e., the finite field with q elements). Now a linear (block) code $\mathcal{C}$ of length n over GF(q) is a linear subspace of GF$(q)^n$. The *minimum distance* d of $\mathcal{C}$ is equal to the *minimum non-zero weight* codeword in $\mathcal{C}$. From now on, let $\mathcal{C}$ be a linear code over GF(q) with code length n, dimension k and minimum distance d. This code will be referred to as a q-ary $[n, k, d]$-code. When the minimum distance is not known or, of no importance, then $\mathcal{C}$ is denoted as a q-ary $[n, k]$-code.

The maximum amount of *information* that can be transferred by code $\mathcal{C}$ is $\log_2 |\mathcal{C}|$ bits. In a q-ary alphabet, the maximum information contained in n symbols is $\log_2 q^n$ bits. The maximum *information rate* of $\mathcal{C}$ is defined by

$$\frac{\log_2 |\mathcal{C}|}{\log_2 q^n} = \frac{1}{n} \log_q |\mathcal{C}| = \frac{k}{n}. \tag{2.2}$$

The fraction k/n is called the *rate* R of $\mathcal{C}$. The set of all q-ary linear codes of length n and rate R is denoted as $\mathcal{C}(n, q, \mathrm{R})$. Since the code $\mathcal{C}$ has dimension k ($k \leq n$), there exist k linearly-independent codewords that form a basis of the linear subspace $\mathcal{C}$ of GF$(q)^n$. When these k codewords are the rows in the $k \times n$ matrix G that generates $\mathcal{C}$, then G is referred to as a *generator matrix.* An *information vector* $\underline{x}$ is encoded into the codeword $\underline{c}$ as follows: $\underline{c} = \underline{x}G$. The generator matrix of a linear code is not unique, the only exception is when $k = 1$ and $q = 2$.

A generator matrix G is in *systematic form* when $G = (I_k \mid A)$, where I_k is the $k \times k$ identity matrix and A is a $k \times (n-k)$ matrix. If G is in systematic form, then the first k symbols of a codeword $\underline{x}G$ are equal to the information vector $\underline{x}$ and are called *information symbols.* The remaining $r = n - k = n(1 - \mathrm{R})$ symbols are called *parity-check symbols* and form the *redundancy* of the codeword (note that r/n equals $1 - \mathrm{R}$).

Two linear codes $\mathcal{C}_1$ and $\mathcal{C}_2$ are called *equivalent* when a coordinate permutation exists, such that, each codeword of $\mathcal{C}_1$ is permuted into a codeword of $\mathcal{C}_2$ and vice versa (i.e., when the codes only differ in the order of the symbols). Let the matrices G_1 and G_2 generate two equivalent codes C_1 and C_2. Then there exists a non-singular matrix S and a permutation matrix P, such that $G_2 = SG_1P$. As a result, each linear code is equivalent to another linear code which can be generated by a generator matrix in systematic form.

Let H be an $(n-k) \times n$ matrix so that

$$C = \{\underline{c} \in \mathrm{GF}(q)^n \mid \underline{c}H^T = \underline{0}\}, \text{ and } GH^T = O_{k,n-k},$$

where $O_{k,n-k}$ is the $k \times (n-k)$ all-zero matrix (i.e., H is a basis of the null space of G). In other words, C is the solution space of the $n-k$ linearly-independent equations $\underline{c}H^T = \underline{0}$. The matrix H is called the *parity-check matrix*. When the generator matrix is in systematic form $(I_k|A)$, then a parity-check matrix is $(A^T|I_{n-k})$. The code C has a codeword $\underline{c}$ of weight w if, and only if, some w columns of H are linearly dependent. Therefore, a code has minimum distance d if, and only if, every combination of $d-1$ columns of H is linearly independent and at least one combination of d columns is linearly dependent.

Let $\underline{y} = \underline{c}+\underline{z}$, where $\underline{c}$ is a codeword of C and where $\underline{z} \in \mathrm{GF}(q)^n$ is an error vector. The syndrome $\underline{s}$ of the vector $\underline{y}$ with respect to H of C is defined by $\underline{s} := \underline{y}H^T$. For example, if the received vector has syndrome $\underline{0}$, then it is a codeword, and most likely no errors have occurred during its transmission over a *q-ary symmetric channel* (QSC, see Section 4.1 on page 40). For any $\underline{z}$, the set

$$\underline{z} + C := \{\underline{z} + \underline{c} \mid \underline{c} \in C\}$$

is referred to as a coset of C. Each coset contains q^k vectors, and there are exactly q^{n-k} different cosets. Two vectors $\underline{x}$ and $\underline{y}$ are in the same coset if, and only if, their syndromes are equal (i.e., $\underline{y} - \underline{x} \in C$). Moreover,

$$\underline{s} = \underline{y}H^T = (\underline{c} + \underline{z})H^T = \underline{z}H^T.$$

For example, to find the codeword $\underline{c}$ that is closest to the received vector $\underline{y}$, search for a minimal weight vector $\underline{z}$ that satisfies $\underline{s} = \underline{z}H^T$. A *cosetleader* of a particular coset is an element of minimum weight in that coset. In general, a cosetleader is not unique. However, when C has minimum distance $d = 2t+1$, every vector of weight $\leq t$ is the unique cosetleader of some coset (two vectors with weight $\leq t$ have distance $\leq 2t$ and are therefore in different cosets).

The *sphere-packing radius* is the maximum radius t, such that, all the spheres $\mathsf{S}_t(\underline{c})$ around the codewords $\underline{c}$ in C are disjoint. Therefore, each q-ary $[n,k]$-code that corrects t errors satisfies

$$\sum_{i=0}^{t} \binom{n}{i} (q-1)^i \leq q^{n-k}. \tag{2.3}$$

This inequality is called the *Hamming bound* or *Sphere-packing bound.* When equality holds, the code is called *perfect.* In the set $C(n,q) = \cup_R\, C(n,q,\mathrm{R})$, only a very small fraction of codes are perfect. Substituting (2.1) into (2.3) results in the *asymptotical Hamming bound*:

$$1 - \mathrm{R} \geq \mathsf{H}_q(\frac{d}{2n}), \qquad (n \to \infty). \tag{2.4}$$

The relationship between the sphere-packing radius t and the minimum distance of a linear code is: $t = \lfloor (d-1)/2 \rfloor$.

To determine the maximum distance a received vector can be from the closest codeword, the concept of *covering radius* is introduced. The covering radius ρ is the minimum radius of the spheres around all codewords in C that is necessary to cover all elements of $\mathrm{GF}(q)^n$. Hence, every received vector lies at most at distance ρ from at least one codeword. For a linear code C, the covering radius equals the weight of the maximum weight cosetleader:

$$\rho = \max_{\underline{x} \in \mathrm{GF}(q)^n} \min_{\underline{c} \in C} \mathsf{d}_{\mathrm{H}}(\underline{x}, \underline{c}). \tag{2.5}$$

Note that the minimum distance of a code C corresponds to the largest radius $t = \lfloor (d-1)/2 \rfloor$, such that, the spheres $\mathsf{S}_t(\underline{c})$ with $\underline{c} \in C$ are disjoint. Also, the covering radius is the smallest ρ, such that, $\mathrm{GF}(q)^n$ is contained in the union of the spheres $\mathsf{S}_\rho(\underline{c})$ with $\underline{c} \in C$. Since the covering radius is an integer greater than or equal to $(d-1)/2$, it follows that $\rho \geq t$ (i.e., if a code's purpose is to correct all or fewer combinations of t errors, all vectors of weight t or less must be cosetleaders). Recall that for perfect codes equality must hold true.

The *Gilbert distance* [Gil52] is the minimum radius of a sphere necessary to contain q^{n-k} vectors. It does not belong to a specific code.

The next theorem, an improvement of Gilbert's result by Varshamov [Var57] (and independently by Sacks [Sac58]), is commonly referred to as the *Gilbert-Varshamov* or *Varshamov-Gilbert bound.* Although the improved bound gives a stronger result for the minimum distance, asymptotically it is equal to the *Gilbert bound.*

Theorem 2.2.1 (Gilbert-Varshamov Bound) *There exists a q-ary linear code C of length n, minimum distance $d \geq d_{GV}$ and dimension $k \geq n - r$, whenever*

$$\sum_{i=0}^{d_{GV}-2} \binom{n}{i} (q-1)^i \geq q^r, \tag{2.6}$$

where d_{GV} is the smallest integer that satisfies (2.6). It is referred to as the Gilbert-Varshamov distance. Moreover, it holds that

$$d_{GV} = nH_q^{-1}(1-R) + o(n), \quad (n \to \infty). \tag{2.7}$$

Proof: The objective is to construct an $r \times n$ parity-check matrix H for C, such that, no $d-1$ columns of H are dependent (i.e, C has at least a minimum distance of d). First, choose any nonzero vector as the first column. Second, choose any column that is a nonzero vector and is linearly independent from the first column. Suppose that i distinct nonzero columns are chosen, so that no $d-1$ columns are linearly dependent. As long as the set of all linear combinations of $d-2$ or fewer columns does not include all $q^r - 1$ nonzero vectors, that is

$$\sum_{j=1}^{d-2} \binom{i}{j} (q-1)^j < q^r - 1, \tag{2.8}$$

another column can be added, such that, the codewords of length i are at least distance d from each other. Let $n-1$ be the largest number i for which (2.8) holds. Then an $[n, k]$-code exists with minimum distance $d \geq d_{GV}$, where d_{GV} is the smallest integer that satisfies (2.6).

From Lemma (2.1.1) it follows that

$$q^{n-k} \leq \sum_{i=0}^{d-2} \binom{n}{i} (q-1)^i = q^{n\mathrm{H}_q(\frac{d-2}{n})+o(n)}, \quad (n \to \infty).$$

For sufficiently large d, it holds that

$$1 - \mathrm{R} \leq \mathsf{H}_q(\frac{d}{n}).$$

Then (2.7) follows since d_{GV} is the smallest integer d that satisfies this inequality. □

Pierce [Pie67] proved that the Gilbert-Varshamov (GV) bound is tight for virtually every (binary) linear code. The code in the proof of Theorem 2.2.1 has no specific structure except that none of the combinations of $d-1$ columns are linearly dependent (i.e., the code is almost random). Since most of the symbol sequences of a given length are virtually random, Coffey and Goodman [CG90a] used an approach based on the theory of Kolmogorov complexity [LV93] to prove that virtually every random q-ary linear code is *good* in the sense that it satisfies the Gilbert-Varshamov bound. They also proved that a weaker converse holds true: A q-ary linear code selected at random satisfies the GV-bound with probability asymptotically approaching one.

Important asymptotical relationships for the minimum distance d and covering radius ρ are given in Lemma 2.2.2. In this lemma, *Virtually all linear codes C in $C(n,q,R)$* refers to the fraction of codes that satisfy, $d+\epsilon$ in (2.9) tends to zero for any $|\epsilon| > 0$. This meaning also holds for $\rho + \epsilon$ in (2.10). A proof of the first relationship can, for example, be found in Coffey, Goodman and Farrell [CGF91]. Blinovskii [Bli87] proved the second one.

Lemma 2.2.2 *Virtually all linear codes C in $C(n,q,R)$ satisfy:*

$$\text{\textit{Minimum distance:} } d = nH_q^{-1}(1-R) + o(n), \quad (n \to \infty); \tag{2.9}$$

$$\text{\textit{Covering radius:} } \rho = nH_q^{-1}(1-R) + o(n), \quad (n \to \infty). \tag{2.10}$$

The two expressions in this lemma are important because they are code independent (i.e., they only depend on the code parameters n, q and R). Some exceptions to the Lemma 2.2.2 are perfect codes where $\rho = (d-1)/2$ and algebraic-geometric codes where $d > n\mathsf{H}_q^{-1}(1-\mathrm{R}) + o(n)$. Combining the expressions for the covering radius ρ in Lemma 2.2.2 and the Gilbert-Varshamov distance in Lemma 2.2.1 yields the following corollary for $n \to \infty$.

Corollary 2.2.3 *For virtually all codes C in $\mathcal{C}(n,q,R)$, it holds that*

$$\rho = d_{GV}(1+o(1)), \qquad n \to \infty.$$

The significance of this result in relationship to decoding linear codes is explained in Section 2.4. Additional information can be found in [MS77] and [Lin82].

2.3 A Complexity Class

A computational *problem* is defined by a description of its parameters (input) and of the properties that its solution is required to satisfy (output). An *instance* of a problem is obtained by specifying particular values for its parameters. The process that solves the problem is called an *algorithm.*

The complexity of a problem must be measured in order to evaluate its difficulty. Since modern technology can now place more than one processor on a single chip, hardware space and parallel time have become important complexity measurements. In complexity theory, the number of steps (*time*) and the memory units (*space*) on a *Turing machine* are also used as a measurement. In common practice, however, elementary bit operations or integer multiplications are more important and convenient and are therefore used in this dissertation. Also, these complexity measurements do not alter the problems in the complexity classes $\mathcal{P}$ and $\mathcal{NP}$.

A problem belongs to the class $\mathcal{P}$ if it can be solved on a *deterministic* machine in a number of steps bounded by a polynomial in the size of the input. Considering a problem *easy* or an (deterministic) algorithm *efficient* requires that the constants (degree, coefficient) of the polynomial are small or at least within reasonable range. A problem is referred to as *intractable* if it is not in the complexity class $\mathcal{P}$.

A *nondeterministic* machine has several possibilities for its behavior when in a specific state. The complexity class $\mathcal{NP}$ is the set of problems solvable by a nondeterministic algorithm whose running time is bounded by a polynomial in the size of the input. Therefore, a problem in $\mathcal{NP}$ can be visualized as a tree structure where the depth of the tree is bounded by a polynomial in the size of the input. This can be described as a single machine making guesses and verifying them one at a time or as a maximum number of parallel machines verifying the guesses simultaneously. The introduction of randomness (guessing) in algorithms is useful in solving complex deterministic problems especially when the failure events are independent.

The class $\mathcal{NP}$ contains the class $\mathcal{P}$ by definition. Consequently, $\mathcal{P}$ is a subset of $\mathcal{NP}$. Whether or not this inclusion is proper is still an unsolved problem. Cook [Coo71] proved that a *satisfiability* (SAT) problem has the following property: All problems in $\mathcal{NP}$ can be reduced to the SAT problem in polynomial time. Karp [Kar72] showed that the SAT problem can be reduced to many other $\mathcal{NP}$ problems. Thus, if any of these $\mathcal{NP}$ problems can be solved in polynomial time, they all can. As a result, $\mathcal{NP}$ would be equal to $\mathcal{P}$. The class of $\mathcal{NP}$ problems with this property are called *$\mathcal{NP}$-complete problems.* In essence, this means that there is no known algorithm, or any polynomial p, which solves an $\mathcal{NP}$-complete problem in $p(s)$ steps (where s is the size of the input). If such an algorithm did exist, it could also be applied (with minor modifications) to solve a whole class of $\mathcal{NP}$ problems (for more information, refer to [AHU74, GJ79] and [PS82]).

In this dissertation, the performance of a decoding algorithm is evaluated by using the following measurements for time (*work-factor*) and space (*memory-factor*):

Definition 2.3.1 (Work-Factor and Memory-Factor) *Let $\mathcal{A}$ be an algorithm (perhaps probabilistic) that solves a given problem. The work-factor $\mathcal{W}_{\mathcal{A}}$ of $\mathcal{A}$ is the (expected) number of elementary operations, and the memory-factor $\mathcal{M}_{\mathcal{A}}$ of $\mathcal{A}$ is the (expected) number of memory units necessary to solve the problem.*

A relative complexity measurement is used to compare decoding algorithms to exhaustive-search decoding algorithms (Section 4.2). Therefore, the complexity coefficient is defined as the logarithm of the work-factor times the memory-factor normalized on the word length.

Definition 2.3.2 (Complexity and Complexity Coefficient) *Let $\mathcal{A}$ be an algorithm (perhaps probabilistic) that solves a given decoding* *problem for the class of linear codes $\mathcal{C}(n,q,R)$. The decoding complexity $DC_{\mathcal{A}}(n,q,R)$ of $\mathcal{A}$ is the (expected) number of elementary operations multiplied by the (expected) number of memory units necessary to solve the problem. The decoding complexity coefficient $dcc_{\mathcal{A}}(q,R)$ of $\mathcal{A}$ equals*

$$\lim_{n\to\infty} \frac{1}{n} \log_q DC_{\mathcal{A}}(n,q,R), \tag{2.11}$$

if this limit exists.

From this definition it follows that

$$\mathsf{dcc}_{\mathcal{A}}(q,\mathrm{R}) = \lim_{n\to\infty} \frac{1}{n} \log_q \mathsf{W}_{\mathcal{A}}(n,q,\mathrm{R}) + \lim_{n\to\infty} \frac{1}{n} \log_q \mathsf{M}_{\mathcal{A}}(n,q,\mathrm{R}).$$

When only the memory-factor is polynomially bounded, the complexity coefficient asymptotically depends only on the work-factor (i.e., time complexity). On the other hand, when only the work-factor is polynomially bounded, the complexity coefficient asymptotically depends only on the memory-factor (i.e., memory complexity). When both the memory-factor and the work-factor are polynomially bounded, the complexity coefficient is zero.

2.4 A Class of Decoding Problems

This section discusses the *general decoding problem* for linear codes and describe three types of hard-decision decoding techniques. For general decoding problems, recall that a linear code $\mathcal{C}$ consists of all codewords $\underline{c}$, such that $\underline{c}H^T = \underline{0}$. Suppose H is randomly selected and $\underline{y}$ is a given vector. The syndrome of the vector $\underline{y}$ is $\underline{s} = \underline{y}H^T$. Finding a codeword $\underline{c} = \underline{y} - \underline{z}$, where $\underline{z}$ is the minimum weight solution to the equation $\underline{s} = \underline{z}H^T$, is equivalent to the following problem:

Problem 2.4.1 (Coset Weight)
***Input** : An $r \times n$ matrix H over $GF(q)$, a vector $\underline{s} \in GF(q)^r$ and a nonnegative integer w.*
***Output** : A vector $\underline{y} \in GF(q)^n$ with $w_H(\underline{y}) \leq w$, such that, $\underline{y}H^T = \underline{s}$ if this vector exists.*

Berlekamp, McEliece and van Tilborg [BMT78] proved that this coset-weight problem belongs to the class $\mathcal{NP}$-complete. The dual description which asks for a vector $\underline{y}$ with Hamming weight exactly w is also $\mathcal{NP}$-complete. Therefore, the general decoding problem for linear codes is $\mathcal{NP}$-complete. For $\underline{s} = \underline{0}$ (i.e., output a codeword of Hamming weight w or less), the problem in conjectured to be $\mathcal{NP}$-complete [BMT78].

Since the parity-check matrix is normally known beforehand, it can be disputed whether Problem 2.4.1 reflects the actual difficulty of the decoding problem. For example, some form of preprocessing may have made the problem polynomial. Bruck and Naor [BN90] showed that when the code is known in advance the problem remains difficult even when it can be preprocessed indefinitely. In other words, Problem 2.4.1 remains $\mathcal{NP}$-complete.

For decoding algorithms that recover the original codeword from the received word by taking only advantage of the code's redundancy, three types of decoding algorithms are defined. The first class is defined as follows:

Definition 2.4.2 (Complete Hard-Decision Decoding) *Let C be a linear code of length n over $GF(q)$. A complete hard-decision decoding algorithm decodes the vector $\underline{y} \in GF(q)^n$ into a not-necessarily-unique codeword $\underline{c}$, such that*

$$d_H(\underline{y}, \underline{c}) = \min\{d_H(\underline{y}, \underline{x}) \mid \underline{x} \in C\}.$$

A CHDD algorithm solves an optimization problem, while a t-BDD algorithm as defined below solves the corresponding recognition problem:

Definition 2.4.3 (t-Bounded Distance Decoding) *Let C be a linear code of length n over $GF(q)$, and let t be an integer. A t-bounded distance algorithm decodes the vector $\underline{y} \in GF(q)^n$ into a not-necessarily-unique codeword $\underline{c}$, for which*

$$d_H(\underline{y}, \underline{c}) \leq t, \tag{2.12}$$

if such a codeword exists.

If more than one codeword satisfies (2.12), then a t-BDD algorithm does not necessarily decode to the codeword closest to the vector $\underline{y}$. A t-BDD algorithm that always yields a codeword closest to the vector $\underline{y}$ is called a *minimum* t-BDD algorithm. For every vector $\underline{y}$, a not-necessarily-unique codeword $\underline{c}$ exists, such that, $\mathrm{d_H}(\underline{y}, \underline{c}) \leq \rho$ where ρ is the covering radius of $\mathcal{C}$ as defined in (2.5). Therefore, a minimum t-BDD algorithm with $t = \rho$ solves the CHDD problem. Corollary 2.2.3 shows that for sufficiently large n and for virtually all codes $\mathcal{C}$ in $\mathcal{C}(n, q, \mathrm{R})$, a minimum d_{GV}-BDD algorithm solves the CHDD problem.

In case $t \leq \lfloor (d-1)/2 \rfloor$, a codeword found by a t-BDD algorithm is always unique. One subclass of t-BDD algorithms is defined as follows:

Definition 2.4.4 (Bounded Hard-Decision Decoding) *Let $\mathcal{C}$ be a linear $[n, k, d]$-code over $GF(q)$. A bounded hard-decision decoding algorithm decodes the vector $\underline{y} \in GF(q)^n$ into a unique codeword $\underline{c}$, for which*

$$d_H(\underline{y}, \underline{c}) \leq \lfloor (d-1)/2 \rfloor,$$

if such a codeword exists.

Just as the covering radius plays an important role in CHDD, the minimum distance is important in BHDD. The general problem of finding the true minimum distance of a code is conjectured to be $\mathcal{NP}$-complete [BMT78]. This seems to imply that the BHDD problem is $\mathcal{NP}$-complete, however, it is not necessarily true. For many classes of codes, BHDD algorithms do exist with work-factors which are polynomial in the code's length n. Hard-decision decoding techniques will be further discussed in Chapters 4, 5 and 6.

Chapter 3

A Class of Public-Key Cryptosystems

The public-key cryptosystems described in this chapter are based on linear error-correcting codes. Each system description includes a security discussion as it relates to the decoding problems identified in Section 2.4. Section 3.1 presents the McEliece system [McE78]. Its encoding scheme consists of transforming the plaintext into codewords using a t-error-correcting Goppa code and randomly adding error vectors of weight t to each codeword. The generator matrix disguises the corresponding decoding algorithm of the code, so that the security of the McEliece scheme is related to the general decoding problem for linear codes. Section 3.2 discusses how the Niederreiter system [Nie86] is related to the McEliece system. In contrast to the McEliece scheme, the encoding scheme for this system consists of a parity-check matrix and no error vectors are added. Its security is related to characteristics of the parity-check matrix when it is used as a (trapdoor) one-way function. Then, Section 3.3 discusses Stern's protocol set-up used in (interactive) identification protocols [Ste94]. It uses a randomly generated matrix as a one-way function instead of using a disguised parity-check matrix as in the Niederreiter system. Its security is related to characteristics of a randomly generated matrix used as a one-way function. In conclusion, Section 3.4 identifies three related, yet insecure, digital signature schemes.

3.1 The McEliece Scheme

In [McE78], McEliece proposed a public-key cryptosystem that assumes the public key is a random generator matrix of an error-correcting code. The public key conceals the code's efficient decoding algorithm. This locally-randomized cryptosystem is defined as follows:

Definition 3.1.1 (The McEliece Public-Key Cryptosystem) *Let C be a q-ary linear $[n, k, d]$-code. Define a set of correctable error vectors as*

$$\mathcal{Z} = \{\underline{z} \in GF(q)^n \mid w_H(\underline{z}) = \lfloor (d-1)/2 \rfloor\}.$$

Let G be a $k \times n$ generator matrix of the code C for which an efficient decoding algorithm exists. The encryption matrix is $E = SGP$, where S is a random $k \times k$ non-singular matrix over $GF(q)$, and P is a random $n \times n$ permutation matrix.

Encryption *A plaintext $\underline{x} \in GF(q)^k$ is encrypted into the ciphertext $\underline{y} \in GF(q)^n$ with $\underline{y} = \underline{x}E + \underline{z}$, where $\underline{z}$ is randomly chosen from $\mathcal{Z}$.*

Decryption *A ciphertext $\underline{y}$ is decrypted by $\underline{y}P^T = \underline{x}SG + \underline{z}P^T$ (the vector $\underline{z}P^T$ also belongs to $\mathcal{Z}$). Next, the vector $\underline{x}S$ is obtained by applying the decoding algorithm of C. The plaintext $\underline{x}$ is computed as $(\underline{x}S)S^{-1}$.*

Key *The public key is the set of correctable error vectors $\mathcal{Z}$ and the encryption matrix E. The secret key consists of: the two matrices S and P, and the code C's decoding algorithm.*

Each plaintext in the McEliece system corresponds to $|\mathcal{Z}|$ possible ciphertexts making a ciphertext search difficult. Heiman [Hei87] showed that, with overwhelming probability, a cryptanalyst can recognize a pair of ciphertexts within a large set if the plaintexts are identical.

When $\underline{x}_1$ and $\underline{x}_2$ are plaintexts encrypted by the McEliece system, then

$$\underline{y}_1 + \underline{y}_2 = (\underline{x}_1 + \underline{x}_2)E + \underline{z}_1 + \underline{z}_2.$$

Note that although the error-correcting code is linear, the encryption function in the McEliece scheme is not. The cryptanalyst can obtain the codeword $\underline{x}E$ by majority voting when the same plaintext $\underline{x}$ is encrypted several times. Then, the plaintext $\underline{x}$ can be computed as $(\underline{x}E)E^{-R}$ where E^{-R} is the right-inverse of E.

The key-storage space is dominated by the $k \times n$ encryption matrix. In [Til90], the author gave two methods for reducing the key-storage requirements. As a result, the public key can be reduced to a $k \times (n-k)$ matrix.

In his scheme, McEliece proposed using binary irreducible *Goppa codes.* These codes have useful error-correcting properties, efficient decoding algorithms, and are easy to generate. Also, the number of inequivalent Goppa codes increases rapidly as the parameters are expanded. For details about Goppa codes, the reader is referred to [MS77]. For this discussion, it suffices to know that a Goppa code can be described as follows. Let $g(x)$ (the *Goppa polynomial*) be a binary irreducible polynomial over GF(2^m) of degree t for some fixed m. Let $\mathsf{L} = \{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$ be an ordering of GF(2^m) with $g(\alpha_i) \neq 0$ for all $\alpha_i \in \mathsf{L}$. Then the binary irreducible Goppa code $\Gamma(g(x), \mathsf{L})$ is a t-error-correcting linear code of length $n = 2^m$, dimension k and

$$\underline{c} = (c_0, c_1, \ldots, c_{n-1}) \in \Gamma(g(x), \mathsf{L}) \quad \Longleftrightarrow \quad \sum_{i=0}^{n-1} \frac{c_i}{x - \alpha_i} \equiv 0 \bmod g(x). \tag{3.1}$$

Let G be a $k \times n$ generator matrix of the code $\Gamma(g(x), \mathsf{L})$. Then, in the McEliece scheme a plaintext $\underline{x}$ is encrypted as:

$$\underline{y} = \underline{x}E + \underline{z} = \underline{x}SGP + \underline{z} \text{ with } \operatorname{span}(G) = \Gamma(g(x), \mathsf{L}), \tag{3.2}$$

where S is a $k \times k$ invertible matrix, and P is a $n \times n$ permutation matrix (S, P, $g(x)$ and L are part of the private key). A cryptanalyst tries to find an efficient decoding algorithm for (3.2) when only G is given. From (3.1) it follows that the permutation matrix P only effects the ordering of L in (3.2) and is equivalent to the following decoding problem:

$$\underline{y} = \underline{x}G + \underline{z} \text{ with } \operatorname{span}(G) = \Gamma(g(x), \mathsf{L}P). \tag{3.3}$$

This illustrates that the matrix S is of no cryptographic importance in disguising the efficient decoding algorithm. To find an efficient decoding algorithm for (3.3), a parity-check matrix H is computed, so that

$GH^T = O_{k,n-k}$. The next objective is to find a Q-matrix, such that QH has a structure in GF(2^m) similar to

$$\begin{pmatrix} 1/g(\rho_0) & 1/g(\rho_1) & \dots & 1/g(\rho_{n-1}) \\ \rho_0/g(\rho_0) & \rho_1/g(\rho_1) & \dots & \rho_{n-1}/g(\rho_{n-1}) \\ \rho_0^2/g(\rho_0) & \rho_1^2/g(\rho_1) & \dots & \rho_{n-1}^2/g(\rho_{n-1}) \\ \vdots & \vdots & & \vdots \\ \rho_0^{t-1}/g(\rho_0) & \rho_1^{t-1}/g(\rho_1) & \dots & \rho_{n-1}^{t-1}/g(\rho_{n-1}) \end{pmatrix}, \qquad (3.4)$$

where $\{\rho_0, \rho_1, \dots, \rho_{n-1}\} = \{\alpha_0, \alpha_1, \dots, \alpha_{n-1}\}P = \mathsf{L}P$. From (3.4), the ordering of $\mathsf{L}P$ can be found by dividing any row by the previous row. Inverting the elements of the first row gives $g(\rho_i)$, $0 \le i \le n-1$. Then, the interpolation of $g(\rho_i)$ at ρ_i $(0 \le i \le n-1)$ reveals the Goppa polynomial $g(x)$. In this way, $\mathsf{L}P$ and $g(x)$ of the (equivalent) Goppa code are obtained. As a result, when an ordering L and an irreducible polynomial $g(x)$ can be efficiently obtained from the generator matrix G, any enciphered text in the McEliece scheme can be decrypted. In essence, the encryption matrix E that disguises the generator matrix G is used as a *trapdoor one-way function* in the McEliece system. This *one-way function* is easy to compute, yet unless the trapdoor is known, it is difficult to compute its inverse.

The security of the McEliece system can be examined by searching for this or any other trapdoor. The number of invertible binary matrices S is $\approx 0.29 \times 2^{k^2}$ [MS77]. There are approximately $2^{mt}/t$ different irreducible polynomials of degree t over GF(2^m). To each binary irreducible (Goppa) polynomial, there exists a binary irreducible Goppa code, of length $n = 2^m$ and dimension $k \ge n - mt$, capable of correcting at least t errors. This means, there exists a sufficient number of different generator matrices G which generate distinct, binary irreducible t-error-correcting Goppa codes. Also, there are $n!$ possible permutation matrices. It is concluded that, for sufficiently large parameters it is infeasible by an exhaustive search to obtain the private key in the McEliece scheme. As a result, a security analysis of the McEliece public-key cryptosystem often considers the following problem:

Problem 3.1.2 (The Trapdoor in the McEliece Scheme)
Input : The encryption matrix E.

Code parameters			Solution Space Dimension	
m	n	t	Heiman	Guessing
3	8	2	≤ 12	≤ 5
4	16	2	≤ 16	≤ 7
4	16	3	≤ 20	≤ 11
5	32	2	≤ 20	≤ 9
5	32	3	≤ 25	≤ 13
6	64	4	≤ 36	≤ 22
6	64	5	≤ 42	≤ 28

Table 3.1: A solution space comparison between *Heiman's algorithm* and a *Guessing algorithm*.

Output : An invertible matrix S, a permutation matrix P and a generator matrix G, for which an efficient decoding algorithm is known, such that $E = SGP$.

Adams and Meijer [AM89] showed that the likelihood of finding such a trapdoor for Goppa codes is small. They concluded that there is usually only one trapdoor. Gibson [Gib91b] challenged this conclusion and showed that the McEliece system for Goppa codes has many trapdoors. This may be true, however, it is important to note that the number of Goppa codes with the same parameters is exponentially large, therefore the probability of finding one is negligible.

When the ordering of L is known, Heiman [Hei87] gave a construction that derives a decoding algorithm for $\Gamma(g(x), \mathsf{L})$ from the generator matrix G. It was expected to succeed for any alternant code [MS77]. Although Goppa codes are a special subclass of alternant codes, the binary irreducible Goppa code can correct twice as many errors as a general alternant code. Therefore, the number of errors in the McEliece system should be greater than $\lfloor t/2 \rfloor$ when the Goppa code can correct t errors. A *Guessing algorithm* is based on randomly selecting polynomials and then verifying if they are irreducible [Rab80]. Considering the solution space of *Heiman's algorithm*, an Guessing algorithm is more efficient as illustrated in Table 3.1.

Gibson [Gib91a, Gib91b] showed that an (equivalent) efficient decoding algorithm for $\Gamma(g(x), \mathsf{L}P)$ can be found by considering the code $\Gamma(g(x), \mathsf{L})$ and by repeatedly choosing new permutations P. To verify if an irreducible polynomial $g(x)$ is found, he applied a procedure described on page 341 of [MS77]. The algorithm requires at most $n!$ iterations. He showed that several permutations (at least $mn(n-1)$) results in equivalent Goppa codes.

In [Gib93], Gibson gave an analysis of the Gabidulin version of the McEliece system [GPT91] with maximum rank distance codes. He showed that in most instances, the Gabidulin system appeared to have only one trapdoor which seemed to make it easier to find. In particular, the subfield code over GF(2) used by Gabidulin contained only the zero codeword. For the McEliece system, the subfield code introduced a much richer structure that seems to make finding a trapdoor much more difficult. For example, Heiman [Hei87] found a partial set of equations, that had so many solutions other than the secret key that he was not able to use them to find the secret key. However, the trapdoor can be found in polynomial time when generalized Reed-Solomon codes [MS77] are used in the McEliece system, as proved by Sidelnikov and Shetstakov [SS92].

Then in [Til94], the author presented the following related problem:

Problem 3.1.3 (Code Equivalence)
Input : *Two codes $\mathcal{C}$ and $\mathcal{E}$ of the same cardinality.*
Output: *A coordinate permutation, such that every codeword of $\mathcal{C}$ is permuted into a codeword of $\mathcal{E}$, if such a permutation exists.*

For linear codes, *code equivalence* reduces the problem to a search for a nonsingular matrix S and a permutation matrix P, such that $S^{-1}E = GP$ with $\mathcal{E} = \text{span}(E)$ and $\mathcal{C} = \text{span}(G)$.

Suppose Problem 3.1.3 can be solved in polynomial time. Then, the McEliece system's security is upper-bounded by a search through the (exponential) number of different Goppa codes (for given parameters). The number of different BCH codes of a given length is much lower than the value of the code's length squared. In this case, the use of BCH codes, instead of Goppa codes, would render the McEliece system insecure.

Let $E_{\mathsf{U}} = S_{\mathsf{U}}GP_{\mathsf{U}}$ be the encryption matrix of user U (i.e., all participants make use of the same Goppa code). When Problem 3.1.3 is difficult, user B's knowledge of E_{A}, E_{B}, S_{B}, P_{B} and G does not reveal user A's private information. This means, the same code can be utilized by all users, so that there exists an ID-based cryptosystem [TQ93] based on the McEliece system.

McEliece stated that the most promising attack on his system consists of decoding an arbitrary linear code containing correctable errors. Therefore, the security of the McEliece system seems to be based on the difficulty of solving the corresponding BHDD problem. Based on this assertion, McEliece computed for length 1024 a Goppa code, such that the BHDD complexity was at a maximum. This analysis used an (probabilistic) *Information Set Decoding* algorithm (ISD, see Section 4.3 on page 43). McEliece substituted the approximation $(1-t/n)^k$ for p_k with $\gamma = 1$, $\mathsf{V}_k = 0$ and $\mathsf{D}_k = 0$ in Equation (4.7) and found that, without plaintext validation, the maximal work-factor was reached at $t = 50$ with code dimension $k = 512$.

Adams and Meijer [AM89], and Hin [Hin86b] and Jorissen [Jor86], independently, observed that when the exact value for p_k as given by (4.5) was substituted in (4.7) (with $\gamma = 1$, $\mathsf{V}_k = 0$ and $\mathsf{D}_k = 0$), the maximum work-factor was reached at $t = 37$ with code dimension $k = 654$. As a positive consequence, the work-factor increased from $2^{64.9}$ to $2^{84.1}$ and the information rate R increased from 0.51 to 0.64.

Lee and Brickell [LB88] observed that an error search in each round reduced the work-factor. They also included a systematic method for checking if the obtained plaintext was correct. Lee and Brickell found that for a two-error search, assuming that $\gamma = 1$ and $\mathsf{V}_k = 0$ in (4.7), the maximum work-factor was reached at $t = 38$ with code dimension $k = 644$. Li, Deng and Wang [LDW94] expanded Lee's and Brickell's analysis in terms of the parity-check matrix (i.e., Syndrome Decoding). Based on the same assumptions as in [LB88], they found that the maximum work-factor was reached at $t = 41$.

In [Til90] based on an idea of Hin [Hin86a], the author proposed a probabilistic decoding algorithm where $\mathsf{V}_k = 0$ per definition. This algorithm, uses *bit swapping* and includes a systematic method for checking the solution. The author observed that finding the last error location dominated the work-factor. Therefore, in each step of the

Reference	n	k	t	Work-factor
[McE78]	1024	524	50	$2^{64.9}$
[AM89]	1024	654	37	$2^{84.1}$
[LB88]	1024	644	38	$2^{73.4}$
[LDW94]	1024	614	41	$2^{71.8}$
[Til90]	1024	634	39	$2^{71.1}$
1SSD$^{(1)}$ (p 86)	1024	634	39	$2^{70.8}$
BSD$^{(1)}$ (p 95)	1024	624	40	$2^{66.5}$
SBSD (p 103)	1024	614	41	$2^{57.0}$

Table 3.2: For each analysis in the reference, the code parameters' "optimal" values when the work-factor is given in bit operations.

bit swapping algorithm an error search was added, its performance analysis is given in [Til90]. A generalized version of this algorithm is discussed in Chapter 5. In Chapter 6 it is rewritten in terms of the parity-check matrix and three distinct implementations are described. It is also shown that these algorithms are highly suitable for parallel and pipelined implementations.

The results of the research described above is summarized in Table 3.2. This table indicates that the values of the work-factor for the McEliece system with binary irreducible Goppa codes of length $n = 1024$ are low, and for this case its security is disputable.

Another way to analyze the security of the McEliece scheme is to search for a codeword of minimum Hamming weight. Let $\underline{y} = \underline{x}G + \underline{z}$ be the received vector, where G is a generator matrix of a linear $[n, k, d]$-code and $\underline{z}$ is an error vector with weight t. Then, the $(n, k + 1)$ matrix

$$\left(\frac{G}{\underline{y} = \underline{x}G + \underline{z}}\right) \text{ or } \left(\frac{G}{\underline{z}}\right)$$

is the generator matrix of a linear $[n, k + 1, t]$-code (note that $d - t \geq \lfloor (d-1)/2 \rfloor = t$). Therefore, finding the minimum weight codeword in this code corresponds to finding the error vector $\underline{z}$. As a result, the (probabilistic) algorithms described by Leon [Leo88] and Stern [Ste89] for finding vectors of low weight in a code can be applied to the McEliece scheme. As also observed by Chabaud [Cha94], these algorithms require a lot of memory (his analysis of the McEliece assumes that $q_k = 1$ and $D_k = 0$). Finding a codeword of minimum weight is known to be an $\mathcal{NP}$-complete problem. Therefore, in contrast to solving the BHDD problem, which is conjectured to be $\mathcal{NP}$-complete [BMT78], this class of algorithms seems to be less promising.

Korzhik and Turkin [KT91] claimed to have found a polynomial-time algorithm for BHDD. This alleged algorithm is based on an iterative optimization algorithm with $\approx 20n^3$ operations. If it were possible, this result would be considered a major advancement. However, the description and the analysis of the algorithm are not sufficiently precise, and its correct functioning within the claimed time-bound has never be confirmed.

As a result of the linearity of the error-correcting code, a security risk might arise when the message contains redundancy. For example, suppose that the 8-th bit in each byte of the message is a parity, such that

$$m_{8j+8} = \sum_{i=1}^{7} m_{8j+i} \text{ for } j = 0, 1, 2, \ldots$$

Depending on its value, the addition of the j-th rows to the generator matrix can be removed from the ciphertext. As a result, the dimension of the effective generator matrix is reduced by factor eight (i.e., $k' = k(1 - r_p)$ where r_p is the redundancy of the plaintext). For the $1\mathrm{SSD}^{(1)}$ algorithm, this means that an $[n, k]$-code reduces to an $[(n - \lfloor kr_p \rfloor), (k - \lfloor kr_p \rfloor)]$-code with the expected number of errors being $(1 - \lfloor kr_p \rfloor / n)t$. For example when $r_p = 1/8$, the $[1024, 634]$-code with 39 errors is reduced to an $[945, 555]$-code with an expected number of 36 errors. In this case, the work-factor $\mathsf{W}^{(1)}_{1\mathrm{SSD}}$ is reduced from $2^{70.8}$ to $2^{63.3}$.

The linearity of the error-correcting code can also be used to disguise the plaintext with a blinding factor. A random codeword is added to the enciphered text, and since the number of errors remains the same, the security of the system is not affected. A cryptanalyst can take advantage of the blinding property by disguising the unknown plaintext and then distributing the corresponding ciphertext over several (untrusted) code-breaking machines. For example, such a protocol is proposed in [QD91] where the Chinese lotto is used as a code-breaking machine. When one machine returns a solution (i.e., the error vector of weight t), then the cryptanalyst can remove the error vector from the original (unblinded) ciphertext and compute the plaintext. In this example, the successful machine (unaware that it was breaking a code) collects the prize without knowledge of the plaintext, so codebreaking becomes a lucrative game of chance.

Davies and Price [DP84] observed that when the error vector is chosen according to a strict rule, it provides an additional information channel referred to as a *concealed channel* or a *subliminal channel.* Park [Par89], and Lin, Chang and Fu [LCF90] used the concealed channel to improve the information rate of the cryptosystem. The total number of error vectors of length n whose weight is t equals $\binom{n}{t}(q-1)$ and can be uniquely represented by $\lceil \log_q \binom{n}{t}(q-1)\rceil$ symbols. Otherwise, when $\lfloor \log_q \binom{n}{t}(q-1)\rfloor$ additional message symbols are mapped on these error vectors, the maximum information rate becomes

$$\begin{aligned} \mathsf{R_M} &= \mathsf{R} + \frac{\lfloor \log_q \binom{n}{t}(q-1)\rfloor}{n} \\ &\to \mathsf{R} + \mathsf{H}_q(\frac{t}{n}), \qquad (n \to \infty). \end{aligned}$$

For a binary [1024, 634, 79]-code, the concealed channel contains at most 235 bits of information, therefore $\mathsf{R_M} = (634 + 235)/1024 = 0.85$ and asymptotically $\mathsf{R_M} \to 0.852$. Note that the security is not affected when the concealed channel contains random information (i.e., original situation). In all other cases, the cryptanalyst might obtain an important advantage through information leakage.

Struik [Str93] proposed a generalization of the McEliece scheme that used a covering code as an efficient data-reduction algorithm (note that using a code in the reverse order yields a code rate greater than 1). First an k-bit plaintext is enciphered into an n-bit locally-randomized codeword in the usual way. Then, this n-bit vector is mapped into an l-bit ciphertext by using a covering code. The rate of his system equals $\mathsf{R_S} = k/l$, and results in an increase of the rate by over 20 percent without affecting the system's security or the implementation's efficiency. Struik's work also illustrates a weakness of locally-randomized cryptosystems: Due to the local randomization process, the ciphertext in the McEliece is not as random as expected.

Jorissen [Jor86] proposed adding only $t < \lfloor (d-1)/2 \rfloor$ errors intentionally, so that $\lfloor (d-1)/2 \rfloor - t$ additional errors introduced by a noisy channel are correctable. Under worst-case conditions, no additional errors occurred or the intentional errors were canceled. As a result, the work-factor to obtain the resulting error vector decreases significantly. For some applications, however, this solution may be attractive and acceptable.

When the plaintext is known, then the Hamming weight of the plaintext can be computed in an efficient way. In [Hei87], Heiman showed that when the Hamming weight of the plaintext can be computed in an efficient way, the plaintext can also be obtained in an efficient way. In other words, the following two problems are equivalent:

- Given a ciphertext $\underline{y} = \underline{x}G + \underline{z}$, compute the plaintext $\underline{x}$;
- Given a ciphertext $\underline{y} = \underline{x}G + \underline{z}$, compute the Hamming weight of the plaintext $\mathsf{w_H}(\underline{x})$.

Let the error-correcting code in the McEliece system be cyclic, and let the set of error vectors $\mathcal{Z}$ be closed under cyclic shifts. In this case, Heiman proved that the security of each of the error vector's bits is equivalent to the security of the plaintext.

Let $\underline{y}$ be a randomly selected vector from $\mathrm{GF}(q)^n$. The probability that this vector can not be decoded into a vector $\underline{x} \in \mathrm{GF}(q)^k$ is

$$\begin{aligned}\Pr\{\text{Decoding failure}\} &= 1 - q^{-n} \times q^k \sum_{i=0}^{\lfloor (d-1)/2 \rfloor} \binom{n}{i} (q-1)^i \\ &\rightarrow 1 - q^{-n(1-\mathrm{R}-\mathsf{H}_q(d/2n))} \rightarrow 1, \quad (n \rightarrow \infty),\end{aligned}$$

for codes where $1 - \mathrm{R} > \mathsf{H}_q(\frac{d}{2n})$ (i.e., for codes that are beneath the asymptotical Hamming bound (2.4)). Note that for a binary 39-error-correcting [1024, 634]-code the probability of decoding failure is $1 - 2^{-154.8}$. As a result, the McEliece system using Goppa codes can not be used to authenticate messages (even if it was possible to authenticate messages, this type of authentication is of no use due to the linearity of the encryption matrix). Therefore, as McEliece stated in [McE78], the decryption algorithm cannot be used as an encryption algorithm, so that, the McEliece cryptosystem does not include a signature property.

In [PBGV92], Preneel, Bosselaers, Govaerts and Vandewalle described a software implementation of the McEliece system. For $n = 1024$ and $t = 39$, they obtained on a 16 MHz IBM PS/2 model 80 an encryption speed of 6 kbps and a decryption speed of 1.7 kbps (no use was made of the 32-bit 80386 instructions). For fixed values of the code parameters, they concluded that doubling the speed was possible. (Notice that the difference between encryption and decryption speed is only a factor of three.) Instead of the irreducible Goppa polynomials, they proposed using polynomials which are the product of non-repeating factors and have a degree of at least two. The advantage stated was that the key-generating process was quicker. A similar modification was also proposed by Jordan [Jor83].

3.2 The Niederreiter Scheme

Closely related to the McEliece system is the Niederreiter system [Nie86]. Niederreiter proposed a public-key cryptosystem which assumes that the public key is a random parity-check matrix of an error-correcting code. In contrast to the McEliece system, it conceals the code's equivalent parity-check matrix for which an efficient decoding algorithm exists. This system is defined as:

Definition 3.2.1 (The Niederreiter Public-Key Cryptosystem) *Let C be a q-ary linear $[n, k, d]$-code for which an efficient decoding algorithm exists. Let H be a $(n-k) \times n$ parity-check matrix of C. The encryption matrix is $E = SHP$, where S is a random $(n-k) \times (n-k)$ non-singular matrix over $GF(q)$, and P is a random $n \times n$ permutation matrix.*

Encryption *A plaintext $\underline{x} \in GF(q)^n$ with $w_H(\underline{x}) \leq \lfloor (d-1)/2 \rfloor$ is encrypted into the ciphertext $\underline{y} \in GF(q)^{n-k}$ as $\underline{y} = \underline{x}E^T$.*

Decryption *A ciphertext $\underline{y}$ is decrypted by $\underline{y}(S^T)^{-1} = \underline{x}P^T H^T$. Next, $\underline{x}P^T$ is obtained by applying the decoding algorithm of C. The plaintext $\underline{x}$ is computed as $(\underline{x}P^T)P$.*

Key *The public key is the value of the minimum distance d and the encryption matrix E. The secret key consists of: the two matrices S and P, and the code $\mathcal{C}$'s decoding algorithm.*

The size of the key-storage space, as in the McEliece system, is large and it is dominated by the $(n-k) \times n$ encryption matrix. The key storage requirements can be reduced to an $(n-k) \times k$ matrix in a similar way as shown in [Til90].

In contrast to the McEliece system, where each plaintext corresponds to $|\mathcal{Z}|$ ciphertexts, in the Niederreiter system each plaintext corresponds to only one ciphertext. Moreover, since no errors can be added intentionally, the system does not have a *concealed channel* and can not be used for joint encryption and error-correction. The maximum information rate of the system is

$$\begin{aligned} \mathrm{R_N} &= \frac{\lfloor \log_q \sum_{i=0}^{\lfloor (d-1)/2 \rfloor} \binom{n}{i} (q-1)^i \rfloor}{n-k} \\ &\to \frac{\mathsf{H}_q(\frac{d}{2n})}{1-\mathrm{R}}, \qquad (n \to \infty). \end{aligned}$$

For a binary $[1024, 634, 79]$-code, this becomes $\mathrm{R_N} = 235/(1024-634) = 0.6$ and $\mathrm{R_N} \to 0.625$ asymptotically. Note that this is a small improvement over the McEliece scheme which has a rate of 0.619, and when the McEliece scheme makes use of the concealed channel to transfer additional message symbols it becomes 0.852.

In the Niederreiter system, it is the encryption matrix E that is used as a trapdoor one-way function to disguise the parity-check matrix H. Therefore, Problem 3.1.2 can be restated as follows:

Problem 3.2.2 (The Trapdoor in the Niederreiter Scheme)
Input : *The encryption matrix E.*
Output : *An invertible matrix S, a permutation matrix P and a parity-check matrix H, for which an efficient decoding algorithm is known, such that $E = SHP$.*

Since this trapdoor corresponds to the trapdoor used in the McEliece system, similar reasoning applies to the analysis of its security. The security of the Niederreiter system is also similar, because it seems to be based on the difficulty of the corresponding BHDD problem. Based on this assertion, the same BHDD algorithms can be used to evaluate its security. These results were summarized in Table 3.2. As in the McEliece system, the code parameters used in the secret-key variant of the Niederreiter system should be the same in order to yield the same security.

3.3 The Stern Scheme

Besides encrypting messages to obtain secrecy, cryptosystems can also be used to achieve authentication. For example, an *identification protocol* is used when Alice wants to convince Bob that she is indeed Alice. In [Ste94], Stern proposed such an (interactive) identification protocol whose security relies on the difficulty of finding a vector of a given Hamming weight and given syndrome. Instead of a disguised parity-check matrix as in the Niederreiter system, Stern uses a randomly generated matrix as a one-way function. As a result, the security of his identification protocol is related to the characteristics of a randomly generated matrix used as a one-way function, and relies on the following:

Definition 3.3.1 (Protocol Set-Up) *Let H be a random $(n-k)\times n$ matrix H over $GF(q)$ with rank $(n-k)$. Each user U chooses a vector $\underline{y}_U \in GF(q)^n$ with Hamming weight w and computes $\underline{s}_U = \underline{y}_U H^T$.*

Public key *The public key for user U consists of the value w, the matrix H and the vector $\underline{s}_U$.*

Secret key *The secret key for user U is the vector $\underline{y}_U$.*

The security of this protocol set-up is based on the following problem: Given the matrix H, the value w and the vector $\underline{s}$, find a vector of weight w that satisfies $\underline{s} = \underline{y}H^T$. The random matrix H can be considered as a parity-check matrix of a random linear $[n,k,d]$-code where d is unknown. In other words, the difficulty of finding the secret key from the public key in Definition 3.3.1 is equal to the difficulty of Problem 2.4.1 as it relates to syndrome decoding.

If $w \leq \lfloor (d-1)/2 \rfloor$, then a *collision* of public-keys is impossible (i.e., different secret keys with the same syndrome). On the other hand,

Reference	n	k	w	1SSD$^{(1)}$	BSD$^{(1)}$	SBSD
[Har89]	1000	500	30	$2^{47.1}$	$2^{42.1}$	$2^{43.0}$
[Ste90]	512	256	30	$2^{45.0}$	$2^{41.1}$	$2^{40.6}$
[Ste90]	1024	512	40	$2^{56.9}$	$2^{52.4}$	$2^{48.0}$
[Ste94]	512	256	56	$2^{72.9}$	$2^{70.2}$	$2^{58.5}$
[Ste94]	1024	512	110	$2^{132.5}$	$2^{129.8}$	$2^{94.0}$

Table 3.3: The values of the code parameters proposed for each system in the reference with the work-factor in bit operations for syndrome decoding algorithms as described in Chapter 6.

when w increases and its value is greater than $\lfloor (d-1)/2 \rfloor$, then it is more likely that a coset contains more vectors of weight w. When a uniform distribution is assumed of the vectors of weight w over the cosets, then the expected number of vectors (collisions) of weight w with the same syndrome is $\binom{n}{w}/2^{n-k}$. In Stern's identification protocol [Ste94], each key in a collision can be used to impersonate a user (i.e., a key collision results in the same syndrome). Therefore, key collisions result in a decreased work-factor for the cryptanalyst. Since the matrix H is randomly generated, its true minimum distance is not known. Note that finding d for a random code is conjectured [BMT78] to be an $\mathcal{NP}$-complete problem.

In Chapter 2, it was stated that a randomly generated parity-check matrix satisfies the Gilbert-Varshamov distance (2.7) with probability asymptotically approaching one. For this reason, Stern recommended for w a value slightly below the Gilbert-Varshamov bound (Lemma 2.2.1). For example, for the binary case with $n = 2k$ he suggested using:

$$w = n\mathsf{H}_2^{-1}(1 - \frac{1}{2}) \approx 0.11n.$$

Table 3.3 summarizes the results when the syndrome decoding algorithms as described in Chapter 6 are applied to the codes in the different systems (using a similar protocol set-up) proposed by Harari and Stern. For more details, the reader is referred to [Har89, Ste90] and [Ste94]. This table illustrates that for this type of cryptosystems, it is more effective to increase w while keeping n and k fixed, than to increase the values n and k while keeping w fixed. The table also indicates that the code parameters values for the systems in [Har89] and [Ste90] are too low, and that even the random $[512, 256]$-code with weight $w = 56$ is disputable. It should be noted that if the real Gilbert-Varshamov distance (2.6) is used instead of (2.7), then the weight of 56 becomes 59. The work-factor for the SBSD algorithm increases to $\mathsf{W} \approx 2^{61}$, however, the number of possible collisions increases to $2^{3.8}$. The net result becomes $2^{57.2}$ which is even lower. The (probabilistic) algorithms described by Leon [Leo88] and Stern [Ste89] for finding vectors of low weight in a code can also be applied directly to Stern's scheme. However, as also observed by Chabaud [Cha94], these algorithms require a lot of memory. Moreover, even the variant of Stern's algorithm [Ste89] as proposed by Chabaud [Cha94] yields an higher work-factor then the 1SSD$^{(1)}$, BSD$^{(1)}$ and SBSD algorithms.

3.4 Related Signature Schemes

In [DH76], Diffie and Hellman introduced the concept of a *digital signature*. In this concept, when Alice wants to send a signed message $\underline{x}$ to Bob, she sends $\underline{x}$ with its signature $\underline{y} = \mathsf{D_A}(\underline{x})$. The function $\mathsf{D_A}$ is Alice's private signature (or decryption) algorithm. Bob verifies the signature by applying Alice's public encryption algorithm $\mathsf{E_A}$ to $\underline{y}$ (i.e., $\mathsf{E_A}(\underline{y}) = \mathsf{E_A}(\mathsf{D_A}(\underline{x})) = \underline{x}$). For this scheme to be secure, it must be assumed that, it is computationally infeasible to find an algorithm D from $\mathsf{E_A}$ that satisfies $\mathsf{D}(\mathsf{E_A}(\underline{x})) = \underline{x}$ for a non-negligible fraction of all $\underline{x}$. Recently, several other signature schemes have been discovered with more and/or different security features (e.g., blind, undeniable, convertible and fail-stop signatures). Most of these schemes are based on discrete logarithm or factoring problems.

In [Xin90], Xinmei proposed a signature scheme based on error-correcting codes. He claimed that the security of his signature scheme relied on the properties of error-correcting codes and the difficulty of factoring large matrices. Harn and Wang [HW92] observed that, because of this scheme's linearity, it is possible to combine signatures of different messages into a valid signature without factoring. In order to prevent this type of forgery, Harn and Wang proposed applying a nonlinear function to the plaintext. Other modifications were also proposed to improve the systems performance. Alabbadi and Wicker showed in [AW92b] that the Xinmei scheme is also vulnerable to two known-plaintext attacks. In [AW92a], they showed that the Harn and Wang scheme can be broken by a known-plaintext attack. In [Til92], the author showed that in both signature schemes the signature key can be obtained from the public key. As a result, both schemes are *unconditionally insecure.* In [AW93], Alabbadi and Wicker proposed a new signature scheme that used error vectors with higher weights than the coset leader. They claimed that these error vectors can not be obtained through standard decoding techniques making their system immune to the analysis discussed in [AW93]. In [Til93a], the author showed that the ability to verify n signatures with linearly-independent error vectors (where n is the code length) makes this signature scheme insecure. It was also observed that the public key in the Alabbadi-Wicker scheme must contain more information in order to verify a signature. Moreover, it was concluded that similar verification problems exist for the Xinmei scheme [Xin90] and the Harn-Wang scheme [HW92], however, these schemes can be broken without verifying a signature. In [Til93b], the author concluded that signature schemes, where the security is based only on the BHDD problem for linear codes, do not exist.

Chapter 4

Decoding of Random Linear Codes

In Section 2.4, BHDD and CHDD problems are defined for decoding a vector that is a codeword with an added error vector. The basic problem of decoding lies in identifying the error vector or the transmitted codeword from the received vector. Adding vectors to codewords in locally-randomized cryptosystems is similar to transmitting codewords over a noisy channel. The main difference is that in locally-randomized cryptosystems the number of errors is known and is correctable. Hence, the probability of erroneous decoding is zero, and it suffices to use BHDD algorithms. However, for noisy channels, where the probability of erroneous decoding exists, CHDD algorithms are more effective.

This chapter gives strategies for solving BHDD and CHDD problems for random and presumably unstructured linear codes. These strategies are presented in a standardized form in order to characterize their merits and compare their performances to the new general decoding algorithms described in the following chapters. In the past, decoding algorithms were developed for noisy channels. Their objective was to minimize the probability of erroneous decoding. Since these algorithms solve the CHDD problem, they automatically solve a BHDD problem (with the same decoding complexity). In most cases, a CHDD algorithm can be modified so that it still solves the BHDD problem but yields a lower decoding complexity. This benefit alone makes this modification of particular interest to cryptanalysis.

This chapter begins by introducing decoding terminology for noisy channels. Then section 4.2 describes three exhaustive-search decoding algorithms whose complexity is used as a basis for all decoding algorithms discussed in this dissertation. Finally, the concepts of Information Set Decoding and Reduced Search Decoding are described in Sections 4.3 and 4.4, respectively.

4.1 Preliminaries

A *q-ary channel* consists of an input alphabet $\mathcal{X} = \mathrm{GF}(q)$, an output alphabet $\mathcal{Y}$ and the *channel probability* $\Pr(y|x)$ (i.e., the probability that symbol y is received, given that, symbol x was transmitted). A channel is called *memoryless* if the conditional channel probabilities are independent. This means that, for $\underline{x} \in \mathrm{GF}(q)^n$ and $\underline{y} \in \mathcal{Y}^n$, the conditional probability satisfies

$$\mathsf{P}(\underline{y} \mid \underline{x}) = \prod_{i=1}^{n} \Pr(y_i \mid x_i).$$

If in addition $\mathcal{X} = \mathcal{Y}$, then the channel is called a *conventional q-ary channel.* Conventional channels are also considered to be *one-way channels* (i.e., feedback information is discarded). A conventional q-ary channel with symbol-error probability p, is called *symmetric* when

$$\Pr(y|x) = \begin{cases} p & \text{if } y \neq x, \\ 1-(q-1)p & \text{if } y = x, \end{cases}$$

where $p \leq 1/(q-1)$. The *binary symmetric channel* $(q = 2)$ is denoted by BSC. If $\underline{y}$ is the received vector over a QSC, then *Maximum Likelihood Decoding* (MLD) searches among all the codewords in C for a not-necessarily-unique codeword $\underline{c}$ with the maximum, conditional probability $\mathsf{P}(\underline{y} \mid \underline{c})$. Because a cosetleader is a vector of minimum Hamming weight in a coset, MLD over a QSC is similar to searching for a cosetleader with the same syndrome as the received vector $\underline{y}$.

For a q-ary $[n, k, d]$-code, the probability that an arbitrarily received vector is *erroneously* decoded using a MLD strategy is denoted by Pe_{CHDD} for CHDD and Pe_{BHDD} for BHDD. For a code with minimum distance d, any error pattern of $\lfloor(d-1)/2\rfloor$ errors can be corrected. When more than $\lfloor(d-1)/2\rfloor$ errors occur, it is uncertain whether or not correct decoding has taken place. In CHDD, a received vector with more than $\lfloor(d-1)/2\rfloor$ errors can yield the correct codeword. However, in BHDD the same number of errors results in either no codeword or an incorrect one. For a BSC with bit-error probability p, it holds that

$$Pe_{\mathrm{CHDD}} \leq Pe_{\mathrm{BHDD}} = \sum_{i=\lfloor(d-1)/2\rfloor+1}^{n} \binom{n}{i} p^i (1-p)^{n-i}.$$

For additional information, the reader is referred to [Bla83, Ber84] and [PW72].

4.2 Exhaustive Search Decoding

This section discusses three decoding strategies for both CHDD and BHDD problems. Each strategy uses an exhaustive search through the set of codewords, the set of syndrome-cosetleader pairs, or the set of error vectors. For this reason, they are called *exhaustive-search decoding* algorithms.

Algorithm 4.2.1 (Exhaustive Search Codeword Decoding) *Let C be a q-ary $[n,k,d]$-code, and let $\underline{y}$ be the received vector. In Exhaustive Search Codeword Decoding (ESCD), the algorithm searches through all codewords of the code C for the codeword $\underline{c}$ that is closest to $\underline{y}$ (CHDD), or that is at distance less than or equal to $\lfloor(d-1)/2\rfloor$ to $\underline{y}$ (BHDD) if it exists.*

As stated above, this algorithm searches through all the codewords. Because the number of codewords is $|C| = q^k = q^{n\mathrm{R}}$, the decoding complexity coefficient of Algorithm 4.2.1 equals R. This algorithm produces with a probability of 1 a solution for CHDD and the error probability agrees with MLD. This proves the following theorem:

Theorem 4.2.2 *Let R be the rate of a q-ary $[n,k,d]$-code C. For both CHDD and BHDD, the decoding complexity coefficient $dcc_{\mathrm{ESCD}}(q,R)$ of Algorithm 4.2.1 equals R. For a memoryless QSC with $p < q^{-1}$, the* *probability of erroneous decoding equals Pe_{CHDD} and Pe_{BHDD}, respectively.*

Instead of searching through the set of codewords, the following algorithm searches through the set of syndromes for a solution.

Algorithm 4.2.3 (Exhaustive Search Syndrome Decoding) *Let C be a q-ary $[n,k,d]$-code, and let $\underline{s} = \underline{y}H^T$ be the syndrome of the vector $\underline{y}$. In Exhaustive Search Syndrome Decoding (ESSD), the algorithm searches through the set of syndrome-cosetleader pairs to find the minimum weight solution $\underline{l}$ for which $\underline{l}H^T = \underline{s}$.*

The complexity of this algorithm follows from the cardinality of the set of syndrome-cosetleader pairs. For CHDD, the set has cardinality $q^{n-k} = q^{n(1-\mathrm{R})}$, therefore, the decoding complexity coefficient equals $1-\mathrm{R}$. This algorithm produces a solution with a probability of 1. For BHDD, the set only contains pairs in which the cosetleader has a weight no more than $t = \lfloor (d-1)/2 \rfloor$. The cardinality of this reduced set is

$$\sum_{i=0}^{t} \binom{n}{i} (q-1)^i = q^{n\mathrm{H}_q(t/n)+o(n)}, \quad (n \to \infty), \tag{4.1}$$

and substitution of $t = \lfloor (d-1)/2 \rfloor = \frac{1}{2} n \mathrm{H}_q^{-1}(1-\mathrm{R}) + o(n)$ yields the desired decoding complexity coefficients. For both CHDD and BHDD, the error probability agrees with MLD. The above results are summarized in the following theorem:

Theorem 4.2.4 *Let R be the rate of a q-ary $[n,k,d]$-code C. The decoding complexity coefficient $dcc_{\mathrm{ESSD}}(q,R)$ for CHDD of Algorithm 4.2.3 equals $1-R$. For virtually all linear codes C in $\mathcal{C}(n,q,R)$, the BHDD decoding complexity coefficient of Algorithm 4.2.3 is*

$$dcc_{\mathrm{ESSD}}(q,R) = H_q\left(\frac{H_q^{-1}(1-R)}{2}\right).$$

For a memoryless QSC with $p < q^{-1}$, the probability of erroneous decoding equals Pe_{CHDD} and Pe_{BHDD}, respectively.

Since the decoding objective is to find the error vector, the following algorithm searches through the set of possible error vectors for a solution.

Algorithm 4.2.5 (Exhaustive Search Error Decoding) *Let C be a q-ary $[n, k, d]$-code, and let $\underline{y}$ be the received vector. Let $\mathcal{Z}$ be the set of error vectors $\underline{z}$ of Hamming weight, less than or equal to, t. In Exhaustive Search Error Decoding (ESED), the algorithm searches through $\mathcal{Z}$ for the minimum weight solution, such that $(\underline{y} - \underline{z})H^T = \underline{0}$.*

The complexity of this algorithm follows from the number of possible error vectors. In CHDD, the maximum number of errors that need to be examined is equal to $\rho = n\mathsf{H}_q^{-1}(1 - \mathrm{R}) + o(n)$. In BHDD, this value is given by $\lfloor (d-1)/2 \rfloor = \frac{1}{2}n\mathsf{H}_q^{-1}(1 - \mathrm{R}) + o(n)$. Substitution in (4.1) yields the corresponding values of the decoding complexity coefficients. For both CHDD and BHDD, the error probability agrees with MLD. These results are summarized in the following theorem:

Theorem 4.2.6 *For virtually all linear codes C in $\mathcal{C}(n, q, R)$, the decoding complexity coefficient $dcc_{\mathrm{ESED}}(q, R)$ for CHDD of Algorithm 4.2.5 equals $1 - R$. For BHDD, the decoding complexity coefficient of Algorithm 4.2.5 equals*

$$dcc_{\mathrm{ESED}}(q, R) = H_q\left(\frac{H_q^{-1}(1 - R)}{2}\right).$$

For a memoryless QSC with $p < q^{-1}$, the probability of erroneous decoding equals Pe_{CHDD} and Pe_{BHDD}, respectively.

4.3 Information Set Decoding

This section describes a decoding approach, referred to as *Information Set Decoding* (ISD), that reconstructs a codeword from selected locations in the received vector. When these locations contain no errors, it is possible to retrieve the original codeword (error vector).

Recall that a q-ary linear $[n, k, d]$-code C has exactly q^k codewords, all are linear combinations of the rows of a generator matrix G of C.

Since the dimension of $\mathcal{C}$ is k, it is always possible to select k linearly independent columns from G. This leads to the following definition of an *Information Set* (IS):

Definition 4.3.1 (Information Set) *Let $\mathcal{C}$ be an $[n,k,d]$-code over GF(q) with generator matrix G. An Information Set $\mathcal{I}$ is a k-subset of $\{1,2,\ldots,n\}$, such that, the restriction of $\mathcal{C}$ to these coordinates (puncture the other coordinates) still has dimension k (i.e., the restriction of G to these coordinates still has full rank k).*

Therefore, an Information Set $\mathcal{I}$ has the property that for any vector $\underline{v} \in \mathrm{GF}(q)^k$ a unique codeword $\underline{c} \in \mathcal{C}$ exists whose restriction to $\mathcal{I}$ equals $\underline{v}$. Also, the k columns in the generator matrix G of $\mathcal{C}$ that correspond to $\mathcal{I}$ are linearly independent. Recall that $\underline{y} = \underline{c} + \underline{z} = \underline{x}G + \underline{z}$ (for some $\underline{x} \in \mathrm{GF}(q)^k$). So, if the locations of vector $\underline{y}$ that correspond to $\mathcal{I}$ are error free, then $\underline{x}$ can be found (e.g., by Gaussian elimination), and the codeword $\underline{c}$ follows from $\underline{x}G$. In essence, *Information Set Decoding* tries to find an error-free Information Set in the received vector.

Let k be a positive integer, and 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 $\mathfrak{J} = \{\mathcal{J}\}$ be a subset of the collection of all k-subsets on $\{1,2,\ldots,n\}$. For all $\underline{z} \in \mathrm{GF}(q)^n$, let the subvector $\underline{z}_{\mathcal{J}} \in \mathrm{GF}(q)^k$ of the vector $\underline{z}$ be defined as

$$\underline{z}_{\mathcal{J}} := \sigma_{\mathcal{J}}(\underline{z}) = (z_{j_1}, z_{j_2}, \ldots, z_{j_k}).$$

Let $G_{\mathcal{J}}$ denote the submatrix of the matrix G consisting of the columns of G with indices from $\mathcal{J}$. The set $\mathcal{L}_{\mathcal{J}}(\underline{y})$ is the set of codewords that coincides with $\underline{y}$ on the positions of $\mathcal{J}$. For each $\mathcal{J} \in \mathfrak{J}$ and vector $\underline{y} \in \mathrm{GF}(q)^n$, the set $\mathcal{L}_{\mathcal{J}}(\underline{y})$ is contained in $\mathcal{C}$. When $\mathcal{J}$ is an Information Set, the cardinality $|\mathcal{L}_{\mathcal{J}}(\underline{y})|$ is one, so it uniquely defines a codeword. When $\mathcal{I}$ is an Information Set for the code $\mathcal{C}$, the codeword $\underline{c}$ that corresponds to $\underline{y}$ is

$$L_{\mathcal{I}}(\underline{y}) = \{\underline{x} \in \mathcal{C} \mid \sigma_{\mathcal{I}}(\underline{y}) = \sigma_{\mathcal{I}}(\underline{x})\}.$$

Note that on the average, $kt/n = \mathrm{R}t$ of the $\sigma_{\mathcal{I}}(\underline{y})$ coordinates of the vector $\underline{y}$ are in error.

Deleting $d-1$ columns of G does not reduce the rank of the corresponding submatrix of G (i.e., every set of $n-d+1$ coordinates of a codeword contains at least one Information Set). Moreover, from the Singleton bound [MS77] it follows that the coset leaders have weight less than $n-k+1$. As a result, there is at least one set of k coordinates that is error free. Hence, for each cosetleader there exists an Information Set completely contained in the set of zero coordinates of the cosetleader.

Algorithm 4.3.2 (Information Set Decoding) *Let $\underline{y} \in GF(q)^n$ be the received vector and C a q-ary $[n,k,d]$-code. Let $\mathfrak{J}$ be a collection of Information Sets, such that, for any $\underline{y}$ there exists at least one error-free Information Set. For CHDD, an Information Set Decoding (ISD) algorithm searches for a not-necessarily-unique codeword $\underline{c}$, such that*

$$d_H(\underline{y},\underline{c}) = \min\{d_H(\underline{y}, L_{\mathcal{I}}(\underline{y})) \mid \mathcal{I} \in \mathfrak{J}\},$$

and for BHDD, it searches for a unique codeword $\underline{c} \in C$, such that

$$d_H(\underline{y}, L_{\mathcal{I}}(\underline{y})) \leq \lfloor (d-1)/2 \rfloor \text{ and } \mathcal{I} \in \mathfrak{J},$$

if such a codeword $\underline{c} = L_{\mathcal{I}}(\underline{y})$ exists.

The complexity of the ISD algorithm is determined by the cardinality of the set $\mathfrak{J}$ (i.e., the number of distinct Information Sets). Chan and Games [CG81] used a combinatorial approach based on covering systems to obtain the number of k-subsets needed so that at least one k-subset contains no error locations (these k-subsets are not necessarily Information Sets, however, this is not a restriction for sufficiently large codes as shown in Appendix B). Given a set of n coordinates, the (n,l,t) *covering number* $\mathsf{B}(n,l,t)$ is the minimum cardinality of any collection $\mathcal{B}$ of l-subsets on $\{1,2,\ldots,n\}$, so that, any t-subset on $\{1,2,\ldots,n\}$ is contained in at least one of the subsets of $\mathcal{B}$. A set of l-subsets that satisfies this covering condition is called an (n,l,t) *covering system.* The (n,l,t) *covering problem* is to find the (n,l,t) covering system of minimum size. In the case of ISD, the subsets of cardinality t contain the error locations, and the subsets of cardinality $l = n-k = n(1-\mathrm{R})$ contain the parity-check locations, so that the complement of these sets are error-free Information Sets of the code $\mathcal{C}$. This approach is sometimes called *error trapping* (i.e., trapping the error-symbols in the parity-check coordinates). The (n, l, t) covering problem is a long-standing, unsolved problem in combinatorics.

When a covering system for the decoding problem is known, the decoding complexity of an ISD algorithm follows from the cardinality of the set $\mathcal{B}$. In the past years, a considerable amount of work has been done on the problem of determining the covering number $\mathsf{B}(n, k, t)$ and the finding of the corresponding covering systems. If $n \geq t$, then $\mathsf{B}(n, n, t) = 1$ and $\mathsf{B}(n, t, t) = \binom{n}{t}$, and when $n \geq k \geq 1$, then $\mathsf{B}(n, k, 1) = \lfloor n/k \rfloor$. Therefore, as stated in Mills [Mil79], it is sufficient to determine $\mathsf{B}(n, n-k, t)$ for $n > n-k > t > 1$. Schönheim [Sch74] gave the following lower bound on this covering number:

$$\mathsf{B}(n, n(1-\mathrm{R}), t) \geq \left\lceil \frac{n}{n(1-\mathrm{R})} \left\lceil \frac{n-1}{n(1-\mathrm{R})-1} \left\lceil \cdots \left\lceil \frac{n-t+1}{n(1-\mathrm{R})-t+1} \right\rceil \cdots \right\rceil \right\rceil \right\rceil . \tag{4.2}$$

According to Mills [Mil79], for many values for which the covering number is known, this lower bound is actually achieved. Actually, when $n(1-\mathrm{R}) > t > 1$ is fixed and $n \to \infty$, Rödl [Röd85] proved a conjecture made by Erdös and Hanani [EH63]. It stated that there exists a collection of

$$\left(\binom{n}{t} \Big/ \binom{n(1-\mathrm{R})}{t} \right) \times (1 + o(1)) \tag{4.3}$$

k-subsets of an n-set, such that, each t-subset is contained in at least one k-subset of the collection. From (4.3) and Lemma 2.1.1, it follows that

$$\lim_{n\to\infty} \frac{1}{n} \log_q \mathsf{B}(n, n(1-\mathrm{R}), t) = \widehat{\mathsf{H}}_q(\frac{t}{n}) - (1-\mathrm{R})\widehat{\mathsf{H}}_q\left(\frac{t}{n(1-\mathrm{R})}\right), \tag{4.4}$$

where $0 < t/n < (1-\mathrm{R}) < 1$. Although there exist no general algorithms that generate minimal covering sets in an efficient way (except for small sizes or special cases, e.g., [Mil79]), suboptimal coverings, for example, can be generated by a random search [ES74, FR85]. These suboptimal coverings provide a probability of erroneous decoding which is asymptotically equal to that of MLD. Therefore, for an ISD algorithm, a sufficiently large set $\mathfrak{J}$ of k-subsets can be obtained in a probabilistic way. In [Dum93], Dumer stated that suboptimal coverings can be obtained in a constructive way in polynomial time, however this result is only of theoretical interest.

Because of the complexity of the combinatorial covering problem, the following probabilistic approach is useful. Suppose that t out of n symbols are in error. The probability that k out of n symbols selected at random are not in error is

$$p_k = \prod_{i=0}^{k-1}\left(1 - \frac{t}{n-i}\right) = \binom{n-t}{k} \Big/ \binom{n}{k} . \tag{4.5}$$

This probability is equal to the probability that $n-k$ out of n symbols are selected at random and contain t errors, so that

$$p_k = \prod_{i=0}^{t-1}\left(1 - \frac{k}{n-i}\right) = \binom{n-k}{t} \Big/ \binom{n}{t} .$$

The expected number of k-tuples to evaluate before an error-free k-tuple is found equals p_k^{-1}. From

$$\binom{n}{t} = q^{n\widehat{\mathsf{H}}_q(t/n)+o(n)} \quad \text{for every } 0 \leq t/n \leq 1,$$

it follows that

$$\lim_{n\to\infty} \frac{1}{n} \log_q p_k^{-1} = \widehat{\mathsf{H}}_q(\frac{t}{n}) - (1-\mathrm{R})\widehat{\mathsf{H}}_q\left(\frac{t}{n(1-\mathrm{R})}\right) . \tag{4.6}$$

This result coincides with (4.4). In a similar way, the Schönheim lower bound (4.2) equals the right-hand-side of (4.4) when $n \to \infty$.

Let q_k be the probability that a randomly chosen k-subset, of positions in an arbitrary q-ary linear $[n,k,d]$-code C, is an Information Set. Then, for virtually all linear codes C in $\mathcal{C}(n,q,\mathrm{R})$, q_k equals the probability that an arbitrarily chosen q-ary $k \times k$ matrix has rank k, and so it is $\prod_{i=1}^{k}(1-q^i)$, (e.g., $q=2$ yields ≈ 0.289). *Maximum Distance Separable* (MDS) codes are an exception in which every k-column combination of a generator matrix is linearly independent ($q_k = 1$). In this chapter the whole class of k-dimensional linear codes is considered. Note that the probabilities p_k and q_k correspond to independent events.

Suppose that C is a q-ary linear $[n, k, d]$-code. Let V_k be the work-factor to verify whether k columns selected at random of the generator matrix G of C are linearly independent. Let W_k be the *additional* work-factor needed to compute the codeword $\underline{c} \in C$. This codeword agrees with the received vector $\underline{y}$ in the corresponding set of k information symbols. Let D_k be the work-factor needed for the (distance) verification between $\underline{y}$ and $\underline{c}$. The work-factor of algorithms based on ISD is

$$\mathsf{W}_{\text{ISD}} = \gamma \times p_k^{-1} \times [q_k^{-1}\mathsf{V}_k + \mathsf{W}_k + \mathsf{D}_k], \qquad (4.7)$$

where $k = \lfloor n\text{R} \rfloor$, and γ is a machine-dependent constant. Since V_k, W_k, D_k, q_k and M_{ISD} in this probabilistic approach are polynomially bounded, a substitution of (4.6) in (4.7) yields an alternative proof of the following theorem (noted in [CG90b]):

Theorem 4.3.3 (Information Set Decoding) *For virtually all linear codes C in $\mathcal{C}(n, q, R)$, the decoding complexity coefficient of Information Set Decoding satisfies*

$$dcc_{ISD}(q, R) = \widehat{H}_q\left(\frac{H_q^{-1}(1-R)}{a}\right) - (1-R)\widehat{H}_q\left(\frac{H_q^{-1}(1-R)}{a(1-R)}\right), \qquad (4.8)$$

where $a = 1$ for CHDD, and $a = 2$ for BHDD. For a memoryless QSC with $p < q^{-1}$, the corresponding probabilities of erroneous decoding are Pe_{CHDD} and Pe_{BHDD}, respectively.

The binary decoding complexity coefficients for ISD, compared with the three different types of decoding algorithms (ESCD, ESSD and ESED), are illustrated in Figure 4.1.

Recall that the complexity decoding of the ISD algorithm is determined by the cardinality of the collection (of Information Sets) $\mathfrak{I}$. The size of $\mathfrak{I}$ is reduced when a search for error vectors, of weight up to $w = \lfloor \alpha t \rfloor$, is added for each Information Set $\mathcal{I}$ in Algorithm 4.3.2. The codeword is found when an Information Set contains less than w errors. The value for w is chosen to minimize the work-factor. The complexity

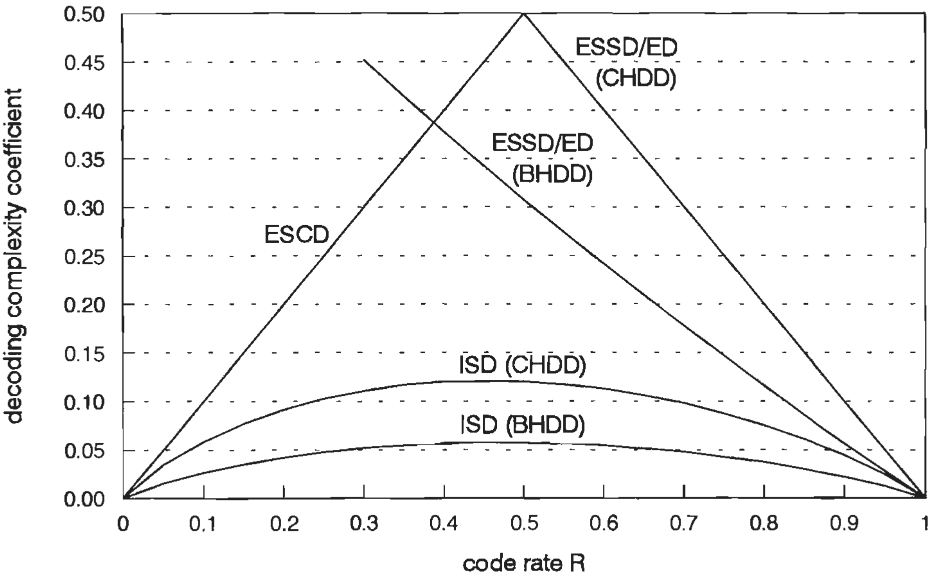


Figure 4.1: The decoding complexity coefficients for Information Set decoding compared to the three exhaustive search decoding algorithms.

coefficient of this error search is:

$$\lim_{n\to\infty} \frac{1}{n} \log_q \sum_{i=0}^{\lfloor \alpha t \rfloor} \binom{k}{i} (q-1)^i = \mathrm{R}\mathrm{H}_q\left(\frac{\alpha t}{n\mathrm{R}}\right).$$

Before finding one codeword that contains at most w errors, the expected number of k-tuples to evaluate is given by the covering number of an $(n, n(1-\mathrm{R}), t(1-\alpha))$ covering system. As a result, the following theorem is proved:

Theorem 4.3.4 *For virtually all linear codes $\mathcal{C}$ in $\mathcal{C}(n,q,R)$, the decoding complexity coefficient of an Information Set Decoding Algorithm with an $w = \lfloor \alpha t \rfloor$-error search satisfies*

$$dcc_{ISD(w)}(q,R) = \widehat{H}_q\left((1-\alpha)\frac{t}{n}\right) - (1-R)\widehat{H}_q\left(\frac{(1-\alpha)t}{(1-R)n}\right) + RH_q(\frac{\alpha t}{nR}),$$

where $t = nH_q^{-1}(1-R)$ for CHDD, and $t = \frac{1}{2}nH_q^{-1}(1-R)$ for BHDD. For a memoryless QSC with $p < q^{-1}$, the corresponding probabilities of erroneous decoding are Pe_{CHDD} and Pe_{BHDD}, respectively.

However, for sufficiently large n and t, an ISD algorithm with an error search yields the same complexity as one without (i.e., $\mathrm{dcc}_{\mathrm{ISD}}(q, \mathrm{R}) = \min_w \mathrm{dcc}_{\mathrm{ISD}(w)}(q, \mathrm{R})$).

In [CG90b], Coffey and Goodman proved the following "remarkable" decoding complexity behavior for Information Set Decoding:

> *For large q, the decoding complexity coefficient for ISD tends to $\widehat{H}_q(1-R)$. Thus $\lim_{q\to\infty} dcc_{ISD}(q, R) = 0$.*

In the proof of this statement, Coffey and Goodman considered the limits $n \to \infty$ and $q \to \infty$ independently. Recall from (2.2) that the rate R of a q-ary linear code $\mathcal{C}$ is defined as

$$\frac{\log_2 |\mathcal{C}|}{\log_2 q^n} = \frac{1}{n}\log_q |\mathcal{C}|.$$

Note that the maximum amount of information that can be transferred by code $\mathcal{C}$ is $\log_2 |\mathcal{C}|$ bits (i.e., the number of codewords $|\mathcal{C}|$ remains the same). Therefore, for fixed R, n decreases as q increases and vice versa. Hence, the limits are not independent and this statement is incorrect.

4.4 Reduced Search Decoding

To illustrate the performance of the Information-Set Decoding algorithms as they relate to other reduced search decoding algorithms, this chapter describes four reduced search decoding algorithms: Q Decoding (QD), L Decoding (LD), Zero-Neighbors Decoding (ZND) and D Syndrome Decoding (DSD).

The first algorithm described for comparison is Q Decoding as introduced by Evseev [Evs83]. The QD algorithm can be regarded as an ISD algorithm that allows errors in the Information Set (i.e., an Information Set $\mathcal{I}$ is combined with an error vector $\underline{v}$). Let $\mathfrak{I}$ denote the collection of the Information Sets of $\mathcal{C}$. The *decoding ensemble* $\mathcal{Q}$ of $\mathcal{C}$ is a subset of pairs $(\mathcal{I}, \underline{v}) \in \mathfrak{I} \times \mathrm{GF}(q)^n$, so that

$$\mathcal{Q} \subseteq \{(\mathcal{I}, \underline{v}) \mid \mathcal{I} \in \mathfrak{I},\ \underline{v} \in \mathrm{GF}(q)^n\}.$$

If the received vector is $\underline{y}$, then every pair $(\mathcal{I}, \underline{v})$ defines a unique codeword $\underline{c}$ with values that agree with $\underline{y} - \underline{v}$ in the coordinate position as determined by the Information Set $\mathcal{I}$. The corresponding error vector is $\underline{z} = \underline{y} - \underline{c}$. For BHDD, the algorithm terminates if an error vector $\underline{z}$ is obtained with $\mathrm{w_H}(\underline{z}) \leq \lfloor (d-1)/2 \rfloor$. For CHDD, the algorithm searches through $\mathcal{Q}$ for an error vector $\underline{z}$ of minimum weight. The following algorithm implements the QD strategy.

Algorithm 4.4.1 (Q Decoding) *Let $\mathcal{C}$ be a q-ary $[n, k, d]$-code, and let $\mathcal{Q}$ be a corresponding decoding ensemble. Suppose $\underline{y}$ is the received vector. For CHDD, the QD algorithm searches for a not-necessarily-unique codeword $\underline{c}$, such that*

$$d_H(\underline{y}, \underline{c}) = \min\{d_H(\underline{y}, L_{\mathcal{I}}(\underline{y} - \underline{v})) \mid (\mathcal{I}, \underline{v}) \in \mathcal{Q}\}.$$

For BHDD, it searches for a unique codeword $\underline{c} \in \mathcal{C}$, such that

$$d_H(\underline{y}, L_{\mathcal{I}}(\underline{y} - \underline{v})) \leq \lfloor (d-1)/2 \rfloor \text{ and } (\mathcal{I}, \underline{v}) \in \mathcal{Q},$$

if such a codeword $L_{\mathcal{I}}(\underline{y} - \underline{v})$ exists.

The complexity of Algorithm 4.4.1 is determined by the cardinality of $\mathcal{Q}$ which makes the construction of the set $\mathcal{Q}$ important. For example, if $\underline{v} = \underline{0}$, then Algorithm 4.3.2 (ISD) is obtained. When the objective is to find an Information Set that contains at most the average number of erroneous locations, then at most $\lfloor \mathrm{R}t \rfloor$ errors in each Information Set are searched for, so that

$$\sum_{i=0}^{\lfloor \mathrm{R}t \rfloor} \binom{k}{i} (q-1)^i = q^{n\mathrm{R}\mathrm{H}_q(\frac{t}{n}) + o(n)}, \quad (n \to \infty). \tag{4.9}$$

For this situation, the number of different Information Sets is polynomially bounded. For example, for cyclic codes it suffices to choose at most $k + t - \mathrm{R}t = k + (1 - \mathrm{R})t = O(n)$ cyclic shifts from the k information set coordinates. Substitution of $t = \rho = n\mathsf{H}_q^{-1}(1 - \mathrm{R}) + o(n)$ and $t = \lfloor (d-1)/2 \rfloor = \frac{1}{2}n\mathsf{H}_q^{-1}(1 - \mathrm{R}) + o(n)$ into (4.9) yields the corresponding values of the decoding complexity coefficients for CHDD and BHDD, respectively. This result is summarized in the following theorem:

Theorem 4.4.2 *Let n, q and R be given. Then a decoding ensemble $\mathcal{Q}$ with an $\lfloor Rt \rfloor$-error search (4.9) exists, such that, for virtually all linear codes $\mathcal{C}$ in $\mathcal{C}(n, q, R)$, the decoding complexity coefficient for CHDD in Algorithm 4.4.1, satisfies*

$$dcc_{QD}(q, R) = R(1 - R),$$

and for BHDD the decoding complexity coefficient satisfies

$$dcc_{QD}(q, R) = RH_q\left(\frac{H_q^{-1}(1 - R)}{2}\right).$$

For a memoryless QSC with $p < q^{-1}$, the probability of erroneous decoding equals Pe_{CHDD} and Pe_{BHDD}, respectively.

For more details about the QD strategy, the reader is referred to [Evs83] and [Dum93].

In [CGF91], Coffey, Goodman and Farrell described an approach similar to QD, however, an Information Set $\mathcal{I}$ is used instead of a k-subset $\mathcal{J}$. Since the rank of the generator matrix $G_{\mathcal{J}}$ is $k - c$, for some nonnegative number c, a set of $k - c$ linearly independent columns is taken from $\mathcal{J}$, and additional c columns are taken from outside $\mathcal{J}$. The resulting set of k-columns is an information set $\mathcal{I}$. Then, an exhaustive error-search is applied on these additional c coordinates, so that when the coordinates corresponding to $\mathcal{J}$ are error-free, the codeword is found. This approach yields the same complexity as stated in Theorem 4.4.2.

Another reduced- was described by Kruk [Kru89]. This algorithm is similar to ISD, except that, the k-subsets are not necessarily Information Sets. The set of codewords that correspond to a k-subset $\mathcal{J}$ is denoted by $\mathcal{L}_{\mathcal{J}}(\underline{y})$ (i.e., the set of codewords that coincides with $\underline{y}$ on the $\mathcal{J}$ positions).

Algorithm 4.4.3 (L Decoding) *Let C be a q-ary $[n,k,d]$-code, $t = \rho$ for CHDD and $t = \lfloor (d-1)/2 \rfloor$ for BHDD. Let $\mathfrak{J} = \{\mathcal{J}\}$ be a collection of k-subsets on $\{1,2,\ldots,n\}$, and let $\underline{y} \in GF(q)^n$ be the received vector. The LD algorithm consists of the following steps:*

Step 1 *For each $\mathcal{J} \in \mathfrak{J}$ compute the set of codewords*

$$\mathcal{L}_{\mathcal{J}}(\underline{y}) = \{\underline{c} \in C \mid \sigma_{\mathcal{J}}(\underline{y}) = \sigma_{\mathcal{J}}(\underline{c})\}, \text{ and let } \mathcal{L} = \bigcup_{\mathcal{J} \in \mathfrak{J}} \mathcal{L}_{\mathcal{J}}(\underline{y}).$$

CHDD Step 2 *Search for a not-necessarily-unique codeword $\underline{c}$, such that*

$$d_H(\underline{y}, \underline{c}) = \min\{d_H(\underline{y}, \underline{x}) \mid \underline{x} \in \mathcal{L}\}.$$

BHDD Step 2 *Search for a unique codeword $\underline{c} \in C$, such that*

$$d_H(\underline{y}, \underline{c}) \leq \lfloor (d-1)/2 \rfloor \text{ and } \underline{c} \in \mathcal{L},$$

if such a codeword $\underline{c}$ exists.

The complexity depends on the cardinality of the set $\mathcal{L}$. This in turn, depends on the construction of the set $\mathfrak{J}$ (the collection of k-subsets). For CHDD, the set $\mathfrak{J}$ must contain at least one k-subset, for which, each error vector with weight t contains no errors in its coordinates. Therefore, the set $\mathfrak{J}$ should be an $(n, n-k, t)$ covering system. In this case, the LD algorithm solves BHDD and CHDD for a QSC with the probability of erroneous decoding equal to Pe_{BHDD} and Pe_{CHDD}, respectively. This result is summarized in the following theorem:

Theorem 4.4.4 *Let n, q and R be given. Then a set $\mathfrak{J}$ exists, such that, for virtually all linear codes C in $\mathcal{C}(n,q,R)$, the decoding complexity coefficient of Algorithm 4.4.3 satisfies*

$$dcc_{\text{LD}}(q,R) = \widehat{H}_q\left(\frac{H_q^{-1}(1-R)}{a}\right) - (1-R)\widehat{H}_q\left(\frac{H_q^{-1}(1-R)}{a(1-R)}\right), \quad (4.10)$$

where $a = 1$ for CHDD, and $a = 2$ for BHDD. For a memoryless QSC with $p < q^{-1}$, the corresponding probabilities of erroneous decoding are Pe_{CHDD} and Pe_{BHDD}, respectively.

The probability of erroneous decoding in the LD algorithm depends on the structure of set $\mathfrak{J}$. Kruk [Kru89] showed that, for almost all codes in $\mathcal{C}(n,q,\mathrm{R})$, when $\mathfrak{J}$ is randomly selected and its cardinality satisfies

$$|\mathfrak{J}| \geq \gamma \binom{n}{t} \Big/ \binom{n-k}{t} \quad \text{and } \gamma = n^2, \tag{4.11}$$

the probability of erroneous decoding is upper-bounded by $Pe_{\mathrm{CHDD}}(1+o(1))$ (a suboptimal covering with a probability of erroneous decoding Pe_{CHDD}, when $\gamma = n\log n$ [Dum93]). Therefore, an LD algorithm with a random set $\mathfrak{J}$, where the cardinality satisfies (4.11), has the same complexity as stated in Theorem 4.4.4. As with QD, some errors are allowed to be in the k-subsets (i.e., combine k-subsets with error vectors). For more details about the LD strategy, the reader is referred to [Kru89] and [Dum93].

Note that the LD and QD strategies can be generalized by relaxing the decoding ensemble $\mathcal{Q}$ to a *decoding ensemble* $\mathcal{R}$ where each k-tuple is combined with an $w \leq \lfloor \alpha t \rfloor$-error search. Then, Algorithm 4.3.2 with a $w = \lfloor \alpha t \rfloor$-error search is obtained with a complexity that was given in Theorem 4.3.4.

From [MS77, Chapter 9] it follows that the average code has a weight distribution approximately equal to a binomial distribution. The low-weight codewords represent the tail of the distribution (i.e., there are few low-weight codewords). Coffey, Goodman and Farrell [CGF91] stated that the difference between a low-weight word and its cosetleader is a codeword of low-weight. Their strategy is to first, find a low-weight word in the coset. Then, add any combination of the low-weight codewords. Finally, take the lowest weight result as the cosetleader. This approach is similar to a reduced exhaustive search throughout the set of codewords (i.e., a set containing only low-weight codewords). The third algorithm described for comparison is based on this principle.

Levitin and Hartman [LH85] proposed the *Zero-Neighbors Decoding* (ZND) algorithm. It is different from the QD, LD and ISD algorithms in that it uses a special set of codewords instead of a decoding ensemble as in QD, a set of k-subsets as in LD, or an Information Set as in ISD.

Define the *domain* $\mathcal{D}(\underline{c})$ as the set of all $\underline{x} \in \mathrm{GF}(q)^n$, such that $\mathrm{d_H}(\underline{x},\underline{c}) \leq \mathrm{d_H}(\underline{x},\underline{c}')$ for all $\underline{c}' \in \mathcal{C} \setminus \{\underline{c}\}$. Let the set $\mathcal{B}(\underline{x})$ contain all words with distance 1 from the word $\underline{x}$. The *domain frame* $\mathcal{G}(\underline{c})$ of a codeword $\underline{c}$ is the set

$$\mathcal{G}(\underline{c}) = \bigcup_{\underline{x}\in\mathcal{D}(\underline{c})} \mathcal{B}(\underline{x}) - \mathcal{D}(\underline{c}).$$

It contains at least one word $\underline{x}$ that lies in the domain of another codeword $\underline{c}'$ ($\neq \underline{c}$). Since the code is linear, without loss of generality, the codeword $\underline{c}$ is $\underline{0}$ (i.e., the set $\mathcal{N}$ of neighbor codewords whose domains are adjacent to the domain of the all-zero codeword). This set is called the *set of zero-neighbors.* The set $\mathcal{N}$ is defined as a subset of codewords $\mathcal{C}$, such that

$$|\mathcal{N}| = \min\{|\mathcal{N}'| \mid \mathcal{N}' \subseteq \mathcal{C} \text{ and } \mathcal{G}(\underline{0}) \subset \bigcup_{\underline{c}\subset\mathcal{N}'} \mathcal{D}(\underline{c})\}. \qquad (4.12)$$

In essence, this domain forms a minimum covering for the domain frame of the all-zero codeword. Note that in the worst case $\mathcal{N}$ contains all codewords. The cardinality of (4.12) is uniquely defined, even though, the structure of the set is not. Note that when $\underline{x}$ is not in $\mathcal{D}(\underline{0})$, there exists a codeword $\underline{c} \in \mathcal{N}$, such that $\mathrm{w_H}(\underline{x} - \underline{c}) < \mathrm{w_H}(\underline{x})$. This observation forms the basis for the following implementation of the ZND algorithm:

Algorithm 4.4.5 (Zero-Neighbors Decoding) *Let $\mathcal{N}$ be the set of zero-neighbors of the q-ary $[n, k, d]$-code $\mathcal{C}$, let $\underline{y}$ be the received vector, and let $\underline{z} = \underline{y}$.*

Step 1 *If there exists an $\underline{n} \in \mathcal{N}$ such that $w_H(\underline{z} - \underline{n}) < w_H(\underline{z})$, then proceed to Step 2, otherwise, proceed to Step 3.*

Step 2 *Let $\underline{z} = \underline{z} - \underline{n}$, and return to Step 1.*

Step 3 *The codeword is $\underline{c} = \underline{y} - \underline{z}$.*

In [LH85], Levitin and Hartman showed that the set $\mathcal{N}$ contains all codewords of minimum weight and does not contain codewords of weight greater than $2\rho + 1$. For random codes, the construction of the set of zero-neighbors is an $\mathcal{NP}$-complete problem. Provided the set of zero-neighbors is given, each round of the ZND algorithm decreases the weight of $\underline{z}$ by at least 1 [LH85] (i.e., the number of rounds in Algorithm 4.4.5 is $\leq \mathrm{w_H}(\underline{y}) \leq n$). Therefore, the ZND algorithm terminates after no more than n rounds and achieves with probability 1 a solution for CHDD (and therefore BHDD). The cardinality of $\mathcal{N}$ is given by the number of codewords of weight $\leq 2\rho + 1$. For virtually all linear codes in $\mathcal{C}(n, q, \mathrm{R})$, the number of codewords of a given weight w equals $q^{n(\mathrm{H}_q(w/n)-(1-\mathrm{R}))+o(\sqrt{n})}$ [CGF91], so that substituting $2\rho + 1$ for w and using (2.10) for ρ results in the following theorem:

Theorem 4.4.6 *Let n, q and R be given. A set $\mathcal{N}$ exists, such that, for virtually all linear codes $\mathcal{C}$ in $\mathcal{C}(n, q, R)$, the decoding complexity coefficient of Algorithm 4.4.5 for CHDD and BHDD is:*

$$dcc_{\mathrm{ZND}}(q, R) = \min\{R\ ,\ H_q(2H_q^{-1}(1-R)) - (1-R)\}.$$

For a memoryless QSC with $p < q^{-1}$, the corresponding probabilities of erroneous decoding are Pe_{CHDD} and Pe_{BHDD}, respectively.

For a discussion of this and other variants of the ZND algorithms, the reader is referred to [LH85] and [CGF91].

The next reduced-search decoding algorithm, *D Syndrome Decoding* (DSD), was described by Dumer [Dum89] and [Dum93]. This algorithm is based on constructing two sets of possible syndromes. If there is at least one match between the two sets, then the decoding algorithm terminates with a solution, otherwise, a decoder failure is detected. In DSD, the syndrome $\underline{s}$ of the received vector $\underline{y}$ is computed. If this syndrome is the all-zero vector, then the received vector is a codeword. If $\underline{s} \neq \underline{0}$, the vector $\underline{y}$ is divided into $\underline{y}_{\mathcal{A}}$ and $\underline{y}_{\mathcal{B}}$. Then, an exhaustive error-search is applied to each half, such that, the sum of the corresponding error syndromes $\underline{s}_{\mathcal{A}}$ and $\underline{s}_{\mathcal{B}}$ yields the syndrome $\underline{s}$ (i.e., $\underline{s}_{\mathcal{A}} + \underline{s}_{\mathcal{B}} = \underline{s}$).

Without loss of generality, the following algorithm assumes that the received vector contains t errors:

Algorithm 4.4.7 (DSD(t)) *Let H be the parity-check matrix of a q-ary $[n, k, d]$-code, and let $\underline{y}$ be the received vector. Define two sets $\mathcal{A}$ and $\mathcal{B}$, such that $\mathcal{A} \cup \mathcal{B} = \{1, 2, \ldots, n\}$, $\mathcal{A} \cap \mathcal{B} = \emptyset$, and $|\mathcal{A}| \approx |\mathcal{B}|$. A DSD algorithm consists of the following five steps:*

Step 1 *Compute the syndrome* $\underline{s} = \underline{y}H^T$, *and let i=0.*

Step 2 *Compute the sets:*

$$\begin{aligned} \mathcal{S}_{\mathcal{A}} &= \{(\underline{x},\underline{z}) \mid \underline{z} \in GF(q)^{|\mathcal{A}|},\ w_H(\underline{z}) = i, \text{ and } \underline{x} = \underline{z}H_{\mathcal{A}}^T\}, \text{ and} \\ \mathcal{S}_{\mathcal{B}} &= \{(\underline{x},\underline{z}) \mid \underline{z} \in GF(q)^{|\mathcal{B}|},\ w_H(\underline{z}) = t - i, \text{ and } \underline{x} = \underline{z}H_{\mathcal{B}}^T\}. \end{aligned}$$

Step 3 *If there exists a pair* $(\underline{s}_{\mathcal{A}}, \underline{z}_{\mathcal{A}}) \in \mathcal{S}_{\mathcal{A}}$ *and a pair* $(\underline{s}_{\mathcal{B}}, \underline{z}_{\mathcal{B}}) \in \mathcal{S}_{\mathcal{B}}$, *such that* $\underline{s}_{\mathcal{A}} + \underline{s}_{\mathcal{B}} = \underline{s}$, *then let* $\underline{z} = \sigma_{\mathcal{A}}^{-1}(\underline{z}_{\mathcal{A}}) + \sigma_{\mathcal{B}}^{-1}(\underline{z}_{\mathcal{B}})$, *and proceed to Step 5.*

Step 4 *If* $i < t$, *then let i=i+1 and return to Step 2, otherwise, exit with the message "decoder failure".*

Step 5 *The codeword is* $\underline{c} = \underline{y} - \underline{z}$.

When the number of errors is unknown and $\underline{s} \neq \underline{0}$; apply the DSD($t$) algorithm first with a search for one error, then two, then three, ..., up to the covering radius ρ for CHDD, or $\lfloor (d-1)/2 \rfloor$ for BHDD. Although the t cases can be combined into one algorithm, its complexity remains the same.

In [Dum89], Dumer sorted the sets $\mathcal{S}_{\mathcal{A}}$ and $\mathcal{S}_{\mathcal{B}}$ (as natural numbers), and then merged them into the resulting sorted set $\mathcal{S}$ (note that the complete list of possible codewords is found). He proved that the complexity of the DSD algorithm follows from the cardinality of $\mathcal{S}$ (i.e., $n^{-1} \log |\mathcal{S}|$). When $|\mathcal{A}| \approx |\mathcal{B}|$, then in each round of Algorithm 4.4.7 the set $\mathcal{S}$ has cardinality no greater than

$$2\sum_{i=0}^{t/2} \binom{n/2}{i} (q-1)^i = q^{\frac{n}{2}\mathsf{H}_q(t/n)+o(n)}, \quad n \to \infty. \qquad (4.13)$$

Together the substitution of $t = \rho = n\mathsf{H}_q^{-1}(1-\mathrm{R}) + o(n)$ for CHDD and the substitution of $t = \lfloor (d-1)/2 \rfloor = \frac{n}{2}\mathsf{H}_q^{-1}(1-\mathrm{R}) + o(n)$ for BHDD yields the following theorem:

Theorem 4.4.8 *For virtually all linear codes* C *in* $\mathcal{C}(n,q,R)$, *the decoding complexity coefficient for Algorithm 4.4.7 for CHDD satisfies*

$$dcc_{\mathrm{DSD}}(q,R) = (1-R)/2,$$

and for BHDD the decoding complexity coefficient satisfies

$$dcc_{DSD}(q, R) = \frac{1}{2} H_q \left(\frac{H_q^{-1}(1-R)}{2} \right).$$

For a memoryless QSC with $p < q^{-1}$, the probability of erroneous decoding equals Pe_{CHDD} and Pe_{BHDD}, respectively.

For values of the code rate R greater than 0.96, the decoding complexity for DSD is lower than for ISD. Note that $\lim_{R \to \infty} \mathrm{dcc}_{\mathrm{ISD}}(q, \mathrm{R}) \to 1 - \mathrm{R}$. It seems not possible to expand the DSD algorithm in terms of the generator matrix to obtain an algorithm that yields a decoding complexity of R/2 for CHDD. For more details about the DSD strategy, the reader is referred to [Dum89] and [Dum93].

In [Dum91], Dumer describes a decoding ensemble (GD) which uses the DSD(t) algorithm. Basicly [Dum94], it decodes a punctured code using the DSD(t) algorithm, but actually it tries to correct beyond the distance of this punctured code. The decoding complexity of this algorithm is slightly lower than for ISD as illustrated in Table 4.1. Special cases of this algorithm are the QD, DSD and LD algorithm. Other implementations of (and generalized) strategies can be found in [Leo88, Ste89, Dum91] and [Dum93].

R	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
ISD	0.058	0.091	0.111	0.200	0.200	0.112	0.098	0.075	0.044
GD	0.057	0.089	0.107	0.115	0.115	0.107	0.093	0.070	0.040

Table 4.1: The CHDD decoding complexity coefficient values [Dum91] for the GD and ISD algorithms.

A comparison of all binary decoding complexity coefficients of the BHDD algorithms discussed in this chapter are illustrated in Figure 4.2.

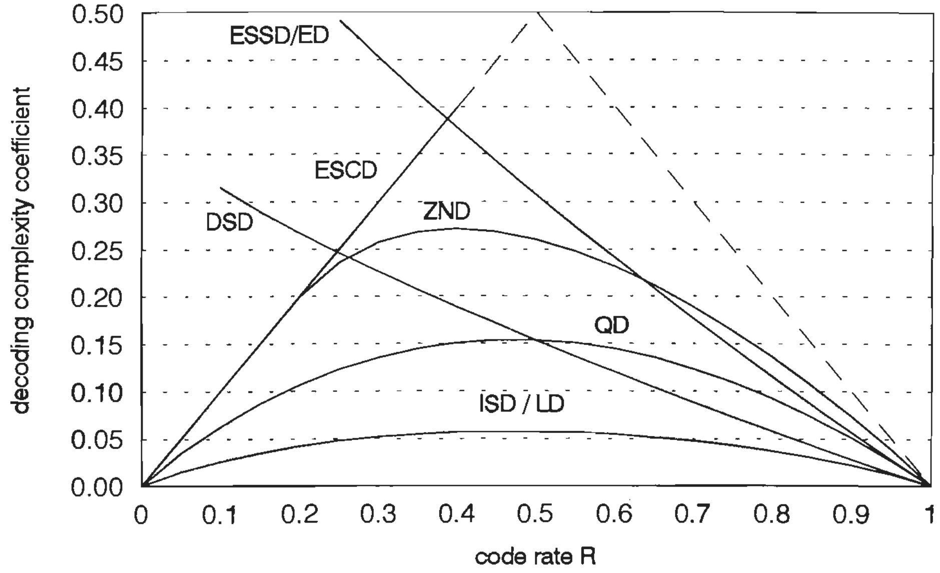


Figure 4.2: Decoding complexity coefficients for different types of bounded hard-decision algorithms.

Chapter 5

Permutation Decoding

The *Bit-Swapping Decoding* algorithm [Til90], based on permutation decoding [MS77], first appeared in a security analysis of the McEliece public-key cryptosystem. Although the name implies that merely bits are swapped, this technique can also be generalized to the q-ary case. This is referred to as *Suitable Permutation Decoding* (SPD), which is based on a swapping procedure that selects permutations from a restricted set of *suitable permutations.* Intuitively, the complexity of SPD seems to be equal to that of Information Set Decoding. However, since the selected permutations in two consecutive rounds of the SPD algorithm are dependent, their equality is not as obvious. Section 5.2, however, gives the proof that their complexity is in fact equal. Then in Section 5.3, the condition of suitable permutation decoding is modified to *Restricted Permutation Decoding* (RPD). In this RPD algorithm, the selected permutation in each round does not necessarily correspond to an information set. Therefore, verification only takes place when an information set is obtained. This section proves that the asymptotical complexity of RPD equals that of Information Set Decoding. Section 5.4 extends this result to *i-Suitable Permutation Decoding* (iSPD). The main characteristic of this class of permutation decoding algorithms is that only information sets are verified for error-freeness. Because more efficient implementations with the same asymptotical complexity are given in Chapter 6, a detailed analysis of the work-factor and memory-

This chapter is based on joint work with Raymond Doyen [DT94b].

factor is omitted.

5.1 Preliminaries

This section establishes the basis for Permutation Decoding used in the context of this dissertation. It also proves a theorem that states whether an information set is error-free. This result is used as a verification step in permutation decoding algorithms.

Consider a q-ary $[n,k,d]$-code $\mathcal{C}$ with generator matrix G, and let $\underline{y}$ be the received vector. A BHDD algorithm finds a solution if, and only if, the received vector can be written as

$$\underline{y} = \underline{c} + \underline{z} = \underline{x}G + \underline{z}, \tag{5.1}$$

where $\underline{z}$ is the error vector with $\mathrm{w_H}(\underline{z}) \leq \lfloor (d-1)/2 \rfloor$ and $\underline{c} \in \mathcal{C}$. Let P be an $n \times n$ permutation matrix, such that the first k columns of GP are linearly independent over $\mathrm{GF}(q)$. Let S be a $k \times k$ non-singular matrix, such that

$$SGP = (I_k \mid A), \tag{5.2}$$

where A is a $k \times (n-k)$ matrix. Since every linear code is equivalent to a code with a generator matrix in systematic form, this decomposition is always possible. If P is applied to $\underline{y}$ in (5.1), then

$$\underline{y}P = \underline{x}GP + \underline{z}P = \underline{x}'SGP + \underline{z}P = \underline{x}'(I_k \mid A) + \underline{z}', \tag{5.3}$$

is obtained, and $\mathrm{w_H}(\underline{z}') = \mathrm{w_H}(\underline{z}) \leq \lfloor (d-1)/2 \rfloor$.

Definition 5.1.1 *Let $\underline{y} = (y_1, y_2, \ldots, y_k, y_{k+1}, \ldots, y_n)$ be a vector of length n. The function $\sigma(\underline{y})$ selects the first k coordinates of the vector $\underline{y}$, and the function $\tau(\underline{y})$ selects the remaining $n-k$ symbols. Hence, $\sigma(\underline{y}) = (y_1, y_2, \ldots, y_k)$ and $\tau(\underline{y}) = (y_{k+1}, y_{k+2}, \ldots, y_n)$.*

The k symbols in the vector $\sigma(\underline{y}P)$ form an information set and are equal to $\underline{x}'S$ in (5.3). If $\sigma(\underline{y}P)$ is error-free, then the correct information vector $\underline{x}'S$ is obtained and $\mathrm{w_H}(\underline{z}P) = \lfloor (d-1)/2 \rfloor$. Because the code $\mathcal{C}$ has minimum distance d, any error in $\sigma(\underline{y}P)$ introduces an additional error $\underline{e}$ of Hamming weight at least d to $\underline{x}'S(I_k \mid A)$. From the triangle inequality, it follows that even when all the $\lfloor (d-1)/2 \rfloor$ errors of $\underline{z}P$ coincide with some of the errors in $\underline{e}$ (i.e, $\mathsf{w}_{\mathsf{H}}(\underline{e} + \underline{z}P) \geq d - \lfloor (d-1)/2 \rfloor > (d-1)/2)$. This is the reason why the weight property, as proved in the following theorem, can be used in the permutation decoding algorithm to verify whether the vector $\sigma(\underline{y}P)$ is error-free.

Theorem 5.1.2 *Let G be a generator matrix of a q-ary $[n,k,d]$-code $\mathcal{C}$. Let P be an $n \times n$ permutation matrix, such that $SGP = (I_k \mid A)$, where S is a $k \times k$ non-singular matrix, and A is a $k \times (n-k)$ matrix. If $\underline{y} = \underline{c} + \underline{z}$, with $\underline{c} \in \mathcal{C}$ and $w_H(\underline{z}) \leq \lfloor (d-1)/2 \rfloor$, then*

$$w_H(\sigma(\underline{z}P)) = 0 \Longleftrightarrow d_H(\sigma(\underline{y}P)SGP, \underline{y}P) \leq \lfloor (d-1)/2 \rfloor.$$

Proof: When $\mathsf{w}_{\mathsf{H}}(\underline{z}P) \leq \lfloor (d-1)/2 \rfloor$, it follows that

$$\mathsf{d}_{\mathsf{H}}(\underline{c}P, \underline{y}P) = \mathsf{d}_{\mathsf{H}}(\sigma(\underline{y}P - \underline{z}P)(I_k \mid A), \underline{y}P) \leq \lfloor (d-1)/2 \rfloor. \qquad (5.4)$$

If $\mathsf{w}_{\mathsf{H}}(\sigma(\underline{z})P) = 0$, then (5.4) yields $\mathsf{d}_{\mathsf{H}}(\sigma(\underline{y}P)(I_k \mid A), \underline{y}P) \leq \lfloor (d-1)/2 \rfloor$. On the other hand, if $\mathsf{d}_{\mathsf{H}}(\sigma(\underline{y}P)(I_k \mid A), \underline{y}P) \leq \lfloor (d-1)/2 \rfloor$, then $\sigma(\underline{y}P)(I_k \mid A)$ must be the corresponding codeword for $\underline{y}P$. Therefore, $\sigma(\underline{y}P)$ must be error-free, and $\mathsf{w}_{\mathsf{H}}(\sigma(\underline{z}P))$ must equal 0. □

5.2 Suitable Permutation Decoding

This section describes the SPD algorithm and establishes its complexity. In the initial phase of the SPD, k-linearly-independent columns of the generator matrix together with the corresponding symbols from $\underline{y}$ are randomly selected. Let the k-selected symbols define the subset $\mathcal{A}$ of $\{1, 2, \ldots, n\}$, and the remaining $n-k$ symbols form set $\mathcal{B}$. The selection of new symbols corresponds to a permutation that swaps new symbols from set $\mathcal{B}$ for symbols from set $\mathcal{A}$. Otherwise stated, this selection corresponds to a permutation that swaps one of the k-columns that corresponds to set $\mathcal{A}$ with one of the $(n-k)$-columns (in the matrix $k \times n$ matrix $(I_k \mid A)$) that corresponds to set $\mathcal{B}$. This leads to the following definition:

Definition 5.2.1 (Swap) *Let $(I_k \mid A)$ be a $k \times n$ matrix. A swap permutes only one column from the I_k-part of the matrix for one column in the A-part.*

The permutation is only successful if the columns of the generator matrix, which correspond to the new set $\mathcal{A}$, are linearly independent. A permutation matrix that fulfills this condition is called a *suitable permutation.*

Definition 5.2.2 (Suitable Permutation) *Let $(I_k \mid A)$ be a $k \times n$ matrix. A permutation P is suitable if the first k columns of matrix $(I_k \mid A)P$ form a non-singular submatrix (i.e., they are linearly independent).*

Based on the previous definitions and Theorem 5.1.2, the SPD algorithm is described as follows:

Algorithm 5.2.3 (Suitable Permutation Decoding) *Let G be the generator matrix of a q-ary $[n, k, d]$-code, and let $\underline{y}$ be a received vector. The SPD algorithm consists of the following five steps:*

Step 1 *Initialization*

- Randomly select a permutation matrix $P^{(0)}$, such that the first k columns of $GP^{(0)}$ are linearly independent.
- Perform elementary row operations on $GP^{(0)}$ to obtain a matrix in systematic form $G^{(0)} := (I_k \mid A^{(0)})$.
- Let $\underline{y}^{(0)} := \underline{y}P^{(0)}$, $u := 0$ and $P := P^{(0)}$. Proceed to Step 4.

Step 2 *Permutation Selection*

- Select a suitable permutation $P^{(u)}$ that swaps only one column, and let $P := PP^{(u)}$.

Step 3 *Swapping and Updating*

- Transform $G^{(u-1)}P^{(u)}$ into $G^{(u)} = (I_k \mid A^{(u)})$, and let $\underline{y}^{(u)} := \underline{y}^{(u-1)}P^{(u)}$.

Step 4 *Verification*

- If $\mathrm{d_H}(\sigma(\underline{y}^{(u)})A^{(u)}, \tau(\underline{y}^{(u)})) \leq \lfloor (d-1)/2 \rfloor$ is false, then $u := u + 1$, and return to Step 2.

Step 5 *Codeword Calculation*

- At this stage, there should be no errors in the first k symbols of $\underline{y}^{(u)}$, consequently $\sigma(\underline{z}^{(u)}) = \underline{0}$. The codeword $\underline{c}$ equals $\sigma(\underline{y}^{(u)})G^{(u)}P^T$.

Define a *round* as Step 2 through Step 4. Then the SPD algorithm can be viewed as a sequence of rounds. Each yields a generator matrix in systematic form. The initial round obtains the matrix $G^{(0)} = (I_k \mid A^{(0)})$. In round u, Algorithm 6.2.1 selects a suitable permutation $P^{(u)}$, such that

$$\begin{aligned} G^{(u)} &= S^{(u)}G^{(u-1)}P^{(u)}, \\ &= S^{(u)}S^{(u-1)}\ldots S^{(0)}GP^{(0)}\ldots P^{(u-1)}P^{(u)}, \\ &= (I_k \mid A^{(u)}). \end{aligned}$$

Since each generator matrix in systematic form (generating a code equivalent to that generated by G) can occur in a round, the SPD algorithm terminates provided it checks each information set no more than once. Moreover, it will terminate in round e with a codeword $\underline{c} = \sigma(\underline{y}^{(e)})G^{(e)}P^T$ provided that

$$\mathsf{d_H}(\sigma(\underline{y}^{(e)})G^{(e)}, \underline{y}^{(e)}) \leq \lfloor (d-1)/2 \rfloor.$$

In practice, the permutations used by this algorithm are neither tracked nor stored.

In CHDD, the upper-bound on the number of errors is usually larger than for BHDD. Therefore, in CHDD the error vector might not be unique. This means that termination with the closest codeword $\underline{c}$ is not guaranteed for the algorithm. However, if the process is continued long enough (i.e., Θ rounds), then it is possible that the desired codeword is found. Heuristic values for Θ are $n\mathsf{B}(n, n-k, t)$, or np_k^{-1}. With high probability, this extension finds the desired solution. When the algorithm terminates, the codeword $\underline{c}$ is a possible solution. In the case where $\mathsf{w_H}(\underline{c}) \leq \lfloor (d-1)/2 \rfloor$, then the obtained codeword $\underline{c}$ is the desired solution.

In the SPD algorithm, symbols are repeatedly swapped in order to obtain k error-free symbols. Let $\Pr(l+s \mid l)$ with $s \in \{-1, 0, 1\}$ denote the probability that $l+s$ of the first k symbols (in $\sigma(\underline{y}^{(u)})$) are in error after one swap, given that l of them were in error before the swap (in $\sigma(\underline{y}^{(u-1)})$). The SPD algorithm starts in an error-state with the number of errors in $\sigma(\underline{y}^{(0)})$ between 0 and $a := \min\{t, k\}$. Without loss of generality, let $t = \lfloor (d-1)/2 \rfloor$. For $1 \leq l \leq a$, the conditional probabilities satisfy:

$$\begin{aligned} \Pr(l-1 \mid l) &= \frac{l(n-k-t+l)}{k(n-k)}, & (5.5) \\ \Pr(l+1 \mid l) &= \frac{(k-l)(t-l)}{k(n-k)}, & (5.6) \\ \Pr(l \mid l) &= 1 - \Pr(l-1 \mid l) - \Pr(l+1 \mid l). & (5.7) \end{aligned}$$

Lemma 5.2.4 *Within Algorithm 5.2.3, let $\widehat{N}_l$ $(1 \leq l \leq a)$ denote the expected number of rounds in an error-state with l errors before going to an error-state with $l-1$ errors. The expected number of rounds in error-state $i-1$ can be computed recursively by:*

$$\widehat{N}_l = \frac{\Pr(l \mid l) + \Pr(l+1 \mid l)(\widehat{N}_{l+1} + 2)}{\Pr(l-1 \mid l)}. \qquad (5.8)$$

Proof: Let l be the error-state of Algorithm 5.2.3. With probability $\Pr(l \mid l)$, the algorithm is expected to stay in the same error-state after one round. With probability $\Pr(l+1 \mid l)$, it reaches error-state $l+1$ after one round and stays there for $\widehat{\mathsf{N}}_{l+1}$ rounds. Only one round is needed to return to error-state l. The expected number of rounds needed to leave error-state l and advance to error-state $l-1$ is $\Pr(l-1 \mid l)^{-1}$. Therefore,

$$\Big(1 \times \Pr(l \mid l) + (\widehat{\mathsf{N}}_{l+1} + 2) \times \Pr(l+1 \mid l)\Big) \times \Pr(l-1 \mid l)^{-1}$$

is the expected number of rounds in error-state l. □

The definition of the expected number of rounds, in the above lemma, is stated in [Til90]. However, it is more convenient to add the round needed to advance to the error-state $l-1$ to the definition of $\widehat{\mathsf{N}}_l$. Therefore, the number of rounds needed to pass to an error-state with $l-1$ errors is defined as $\mathsf{N}_l := \widehat{\mathsf{N}}_l + 1$. Substitution of N_l in Lemma 5.2.4 yields the following corollary:

Corollary 5.2.5 *In Algorithm 5.2.3, the expected number of rounds $N_l := \widehat{N}_l + 1$ $(1 \leq l \leq a)$ needed to pass from an error-state of l errors to an error-state of $l-1$ errors can be computed recursively by:*

$$N_l = \frac{1 + \Pr(l+1 \mid l) N_{l+1}}{\Pr(l-1 \mid l)}. \qquad (5.9)$$

For Algorithm 5.2.3, the expected number of rounds for termination is: the weighted sum of the expected number of rounds over all possible initial error-states. Each one of these error-states $j > 0$ has an expected number of $\mathsf{N}_j + \mathsf{N}_{j-1} + \ldots + \mathsf{N}_1$ rounds with a corresponding weight factor of $\binom{k}{j}\binom{n-k}{t-j}/\binom{n}{t}$. This proves the following lemma:

Lemma 5.2.6 *Let N_l $(1 \leq l \leq a)$ be defined as in Corollary 5.9. Then the expected number of rounds before Algorithm 5.2.3 terminates (i.e., reaches error-state zero) is*

$$\mathsf{N}_{SPD} = \sum_{j=1}^{a} \frac{\binom{k}{j}\binom{n-k}{t-j}}{\binom{n}{t}} \sum_{l=1}^{j} \mathsf{N}_l. \tag{5.10}$$

The SPD algorithm is not sensitive to the initial number of errors in the selected k symbols (initial error-state).

The SPD algorithm starts in a state with a certain number of errors in $\sigma(\underline{y}^{(0)})$. To reduce the number of errors, the SPD algorithm repeatedly swaps symbols until an error-free state is achieved. An additional error search in Step 4 reduces the number of possible error-states, and consequently, the expected number of rounds before the decoding algorithm terminates. Let $\mathcal{U}$ be a subset of $GF(q)^k$, such that the Hamming weight of all the vectors in $\mathcal{U}$ are equal to j. Replace the check in Step 4 with:

- If there exists an $\underline{u} \in \mathcal{U}$, such that

  $$\mathsf{d}_{\mathsf{H}}((\sigma(\underline{y}) - \underline{u})A, \tau(\underline{y})) \leq \lfloor (d-1)/2 \rfloor - \mathsf{w}_{\mathsf{H}}(\underline{u}),$$

  then remove $\underline{u}$ from $\sigma(\underline{y})$ and proceed to Step 5, otherwise, continue to Step 2.

The vector $\sigma(\underline{y})$ is tested by an exhaustive search for an error vector with j errors. The traditional k-out-of-n algorithm is obtained for $j = 0$, and an exhaustive-error-search algorithm (Algorithm 4.2.5) is obtained for $j = t$. The parameter j is chosen, such that it minimizes the work-factor for a given code. For details see [Til90].

Let W_j denote the average work-factor of Step j in Algorithm 5.2.3. Each round has a work-factor of $\mathsf{W}_2 + \mathsf{W}_3 + \mathsf{W}_4$. The algorithm is expected to terminate after N_{SPD} rounds. Since Steps 1 and 5 are executed only once, the average work-factor for the SPD algorithm becomes

$$\mathsf{W}_{\text{SPD}} = (\mathsf{W}_2 + \mathsf{W}_3 + \mathsf{W}_4)\mathsf{N}_{\text{SPD}} + \mathsf{W}_1 + \mathsf{W}_5. \tag{5.11}$$

A suitable permutation can, for example, be found in the following way. Randomly select one column i of the I_k-part. Next, randomly select a column of the A-part that has a non-zero element on its i-th coordinate. In this way, with probability of approximately one-half, a suitable permutation $P^{(u)}$ is found in Step 2. Since a suitable permutation is recognized in an efficient way, $\mathsf{W}_2 = O(n)$. Steps 3 and 4 are executed only after a suitable permutation is obtained. Step 3 computes the matrix $G^{(u)}$ and updates the vector $\underline{y}^{(u)}$, therefore the work-factor W_3 is $O(k \times (n-k)) + O(n) = O(n^2)$. The work-factor W_4 is $O(k \times (n-k)) + O(n) = O(n^2)$, because of a single vector-matrix multiplication and one vector verification. Further, the work-factor W_1 is $O(n^3)$ and W_5 equals $O(n^3)$. Moreover, Lemma A.1.2 shows that N_{SPD} is not polynomial in n. Since Steps 1 and 5 are executed only once, they are neglected. This also justifies the omission of polynomial complexities W_1 and W_5. Substitution of the work-factors in Equation (5.11) provides an asymptotical estimate for the work-factor of the SPD algorithm:

$$\mathsf{W}_{\text{SPD}} = O(n^2)\mathsf{N}_{\text{SPD}}, \quad (n \to \infty). \tag{5.12}$$

For the memory-factor, note that only the matrix $G^{(u)}$, the vector $\underline{y}^{(u)}$ and the permutation matrix P need to be stored. Therefore,

$$\mathsf{M}_{\text{SPD}} = O(n^2). \tag{5.13}$$

The following theorem gives the complexity coefficient of the SPD algorithm. The proof makes use of a lemma that can be found in Appendix A.

Theorem 5.2.7 (Suitable Permutation Decoding) *For virtually all linear codes $\mathcal{C}$ in $\mathcal{C}(n, q, R)$, the complexity coefficient of the Suitable Permutation Decoding algorithm for BHDD satisfies*

$$\mathit{dcc}_{\text{SPD}}(q, R) = \widehat{H}_q\left(\frac{H_q^{-1}(1-R)}{2}\right) - (1-R)\widehat{H}_q\left(\frac{H_q^{-1}(1-R)}{2(1-R)}\right).$$

For a memoryless QSC with $p < q^{-1}$, the probability of erroneous decoding equals Pe_{BHDD}.

Proof: According to Lemma A.1.1, the value of N_1 dominates all the values N_l $(2 \leq l \leq a)$. Therefore, from (5.10) it follows that

$$\mathsf{N}_{\mathsf{SPD}} = \left(\sum_{j=1}^{a} \frac{\binom{k}{j}\binom{n-k}{t-j}}{\binom{n}{t}} \right) \times a\mathsf{N}_1 \times O(1).$$

From Lemmas A.1.2 and A.1.3 it follows that

$$\mathsf{N}_{\mathsf{SPD}} = a\mathsf{N}_1 \times O(1) = O(n^2\sqrt{n})q^{\left(\widehat{\mathsf{H}}_q(\frac{t}{n})-(1-\mathrm{R})\widehat{\mathsf{H}}_q\left(\frac{t}{n(1-\mathrm{R})}\right)\right)}. \tag{5.14}$$

Substituting (5.14) into (5.12) yields the following expression for the complexity of the SPD algorithm:

$$\mathsf{DC}_{\mathsf{SPD}}(n, q, \mathrm{R}) = O(n^4\sqrt{n})q^{n\left(\widehat{\mathsf{H}}_q(\frac{t}{n})-(1-\mathrm{R})\widehat{\mathsf{H}}_q\left(\frac{t}{n(1-\mathrm{R})}\right)\right)}. \tag{5.15}$$

Note that the memory-factor (5.13) is polynomially bounded. When the algorithm terminates, the probability of erroneous decoding equals Pe_{BHDD}. The theorem follows from (5.15) with $d = n\mathsf{H}_q^{-1}(1-\mathrm{R}) + o(n)$ (Lemma 2.2.2). □

For practical situations, the above complexities can not be used directly as they are only asymptotically correct. For a binary, irreducible Goppa code of length 1024, Figure 5.2 shows the expected number of rounds needed in the SPD algorithm for decoding as a function of t. The maximum decoding complexity is at $k = 634$ and $t = 39$.

Example 5.2.8 ([Til90]) *For a binary, 39-error-correcting* $[1024, 634]$-*code C, the number of rounds is given in Table 5.1 and illustrated in Figure 5.1. The corresponding work-factor is approximately* $2^{76.8}$ *bit operations with* $2^{59.4}$ *expected number of rounds.*

When an error search is added to the SPD algorithm, the number of rounds decreases along with the work-factor. For a single error search, the overall work-factor is approximately $2^{71.1}$ *bit operations and the expected number of rounds is reduced to* $2^{53.4}$. *If a (partial) search for patterns with two errors is also performed, then the overall work-factor is approximately* $2^{69.7}$ *bit operations (for a uniform distribution of the error patterns). The expected number of rounds in this case is* $2^{50.3}$.

$\log_2 N_{SPD}$	$\log_2 N_1$	$\log_2 N_2$	$\log_2 N_3$	$\log_2 N_4$	$\log_2 N_5$
59.4	59.4	53.3	48.3	43.9	39.9

Table 5.1: The number of rounds for a $[1024, 634]$*-code with 39 errors.*

For $n = 1024$, $k = 634$ and $t = 39$, the lower-bound (4.2) equals $2^{56.4}$. For $n \to \infty$, a SPD algorithm with an error search yields the same result as one without.

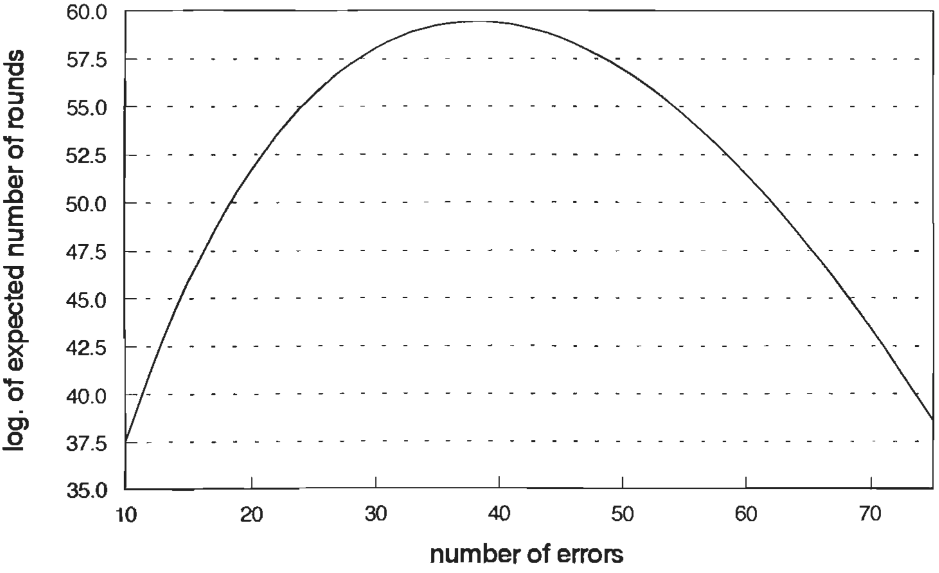


Figure 5.1: The expected number of rounds N_i for error-state i.

5.3 Restricted Permutation Decoding

When the condition of suitable permutations is replaced by one-swap permutations, then the SPD is referred to as the *Restricted Permutation Decoding.* In this section, the RPD algorithm is described and its

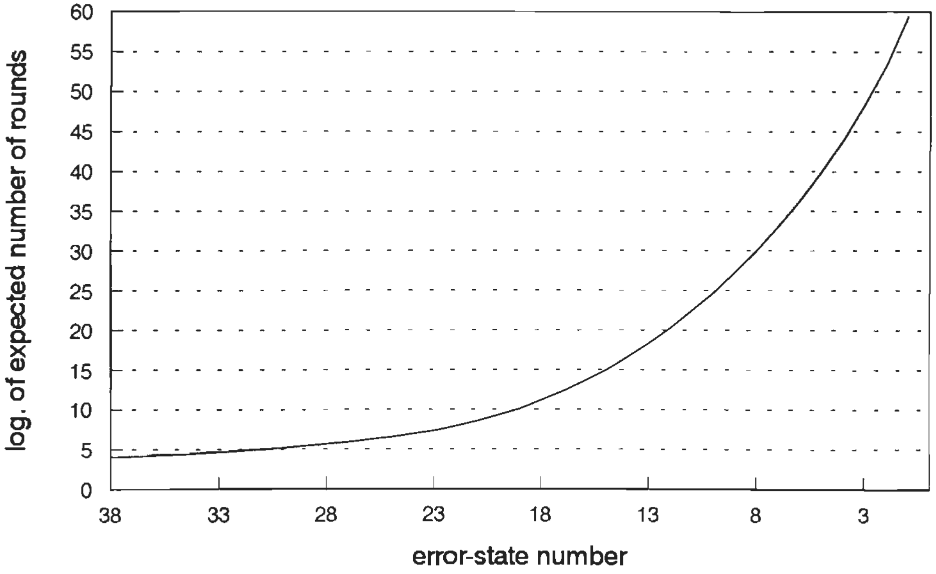


Figure 5.2: The expected number of rounds N as a function of the number of errors t where $n = 1024$, $m = 10$ and $k = n - mt$.

complexity is established. The RPD algorithm uses a special form of the generator matrix G which is defined as:

Definition 5.3.1 (r-KR-Form) *Let $1 \leq r \leq k \leq n$. A $k \times n$ matrix G is said to be in* r-KR-form *if*

$$G = (K \mid R) = \left(\begin{array}{c|c|c} I_r & A & \\ \hline O_{k-r,r} & O_{k-r,k-r} & R \end{array} \right).$$

For $r = k$, the r-KR-form coincides with the systematic form.

After one round in the RPD algorithm, one column from the K-part is exchanged for one column from the R-part. When not restricted to suitable permutations, the rank of the matrix K may vary between 1 and k. Without loss of generality, the RPD algorithm always starts with a full rank K-matrix.

Definition 5.3.2 (Rank Probability) *Let $K^{(u)}$ be a $k \times k$ submatrix of $G^{(u)}$ after u rounds in the RPD algorithm. For $1 \leq r \leq k$, and $u \geq 0$, define*

$$p_r^{(u)} = \Pr\{rank(K^{(u)}) = r\}$$

as the probability that after u rounds the rank of the matrix $K^{(u)}$ is r. The rank probability vector is defined as

$$\underline{p}^{(u)} = (p_1^{(u)}, p_2^{(u)}, \ldots, p_k^{(u)}),$$

with $\underline{p}^{(0)} = (0, 0, \ldots, 0, 1)$.

The sum of the rank probabilities is equal to one, so that for any $u \geq 0$, it holds that

$$\sum_{r=1}^{k} p_r^{(u)} = 1.$$

Definition 5.3.3 (Rank Transition Probability) *Let $K^{(u)}$ be a $k \times k$ submatrix of $G^{(u)}$ after u rounds in the RPD algorithm. For $u \geq 0$, $1 \leq r \leq k$ and $s \in \{-1, 0, 1\}$, define*

$$q_r^{r+s} = \Pr\{rank(K^{(u+1)}) = r + s \mid rank(K^{(u)}) = r\}$$

as the probability that after one-bit swap the rank of the newly obtained matrix $K^{(u+1)}$ equals $r + s$.

In the initialization step of Algorithm 5.3.4 the matrix $K^{(0)}$ has full rank $k \geq 1$. The sum of the rank transition probabilities for a given rank r is equal to one. Therefore, if $1 \leq r \leq k$, then

$$q_r^{r-1} + q_r^r + q_r^{r+1} = 1, \quad \text{and } q_1^0 = q_k^{k+1} = 0.$$

In Section A.2, the rank transition probabilities are calculated for an *arbitrary* but *fixed* k. Define the rank transition matrix as

$$Q := \begin{pmatrix} q_1^1 & q_1^2 & \cdots & 0 & 0 & 0 \\ q_2^1 & q_2^2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & q_{k-2}^{k-2} & q_{k-2}^{k-1} & 0 \\ 0 & 0 & \cdots & q_{k-1}^{k-2} & q_{k-1}^{k-1} & q_{k-1}^{k} \\ 0 & 0 & \cdots & 0 & q_k^{k-1} & q_k^k \end{pmatrix} \tag{5.16}$$

Then for rounds $u > 0$, the rank probability vector $\underline{p}^{(u)}$ is computed as follows:

$$\underline{p}^{(u+1)} = \underline{p}^{(u)}Q = \underline{p}^{(0)}Q^u,$$

with $\underline{p}^{(0)} = (0, 0, \ldots, 0, 1)$.

Let $\underline{\pi} = (\pi_1, \pi_2, \ldots, \pi_k)$ denote the rank probability vector in the equilibrium state (i.e., $\underline{\pi} = \underline{\pi}Q$). Although the probability vector $\underline{p}^{(u)}$ only exists for rounds u before the algorithm terminates, the sequence $\{\underline{p}^{(u)}\}_{u=0,1,2,\ldots}$ is well-defined for every $u \geq 0$. Therefore, the probability that $rank(K^{(u)}) = r$, if the number of rounds u grows to infinity, equals:

$$\lim_{u \to \infty} p_r^{(u)} = \pi_r, \quad 1 \leq r \leq k.$$

The following two asymptotical values for π_r follow from (A.21) and (A.25):

$$\begin{aligned} \pi_k &= 0.289, \qquad (k = \lfloor n\mathrm{R} \rfloor, \text{ with } n \to \infty), \\ \pi_{k-1} &= 0.578, \qquad (k = \lfloor n\mathrm{R} \rfloor, \text{ with } n \to \infty). \end{aligned}$$

Algorithm 5.3.4 (Restricted Permutation Decoding) *Let G be the generator matrix of a q-ary $[n, k, d]$-code, and let $\underline{y}$ be a received vector. The RPD algorithm consists of the following five steps:*

Step 1 *Initialization*

- Randomly select a permutation matrix $P^{(0)}$, such that the first k columns of $GP^{(0)}$ are linearly independent.
- Perform elementary row operations on $GP^{(0)}$ to obtain a matrix in systematic form $G^{(0)} := (I_k \mid R^{(0)})$.
- Let $\underline{y}^{(0)} := \underline{y}P^{(0)}$, $u := 0$, $P := P^{(0)}$ and $r := k$. Proceed to Step 4.

Step 2 *Permutation Selection*

- Select a permutation $P^{(u)}$ that swaps one column.

Step 3 *Swapping and Updating*

- Transform

$$G^{(u-1)}P^{(u)} = \left(\begin{array}{c|c|c} I_r & A^{(u-1)} & \\ \hline O_{k-r,r} & O_{k-r,k-r} & \end{array} \, R^{(u-1)} \right) P^{(u)}$$

  into

$$\left(\begin{array}{c|c|c} I_s & A^{(u)} & \\ \hline O_{k-s,s} & O_{k-s,k-s} & \end{array} \, R^{(u)} \right) P_*^T,$$

  where s is an integer with $s \in \{r-1, r, r+1\}$, and P_* is an $n \times n$ permutation matrix (only introduced for an easy description of the algorithm) in the form

$$P_* = \left(\begin{array}{c|c} P' & O_{k,n-k} \\ \hline O_{n-k,k} & I_{n-k} \end{array} \right), \text{ and let } P^{(u)} := P^{(u)}P_*.$$

  Let $r := s$, $P := PP^{(u)}$ and let $\underline{y}^{(u)} := \underline{y}^{(u-1)}P^{(u)}$. Assign $G^{(u)} := (K^{(u)} \mid R^{(u)})$.

- If $r \neq k$, then $u := u + 1$ and return to Step 2.

Step 4 *Verification*

- If $\mathrm{d_H}(\sigma(\underline{y}^{(u)})R^{(u)}, \tau(\underline{y}^{(u)})) \leq \lfloor (d-1)/2 \rfloor$ is false, then $u := u + 1$, and return to Step 2.

Step 5 *Codeword Calculation*

- At this stage, there should be no errors in the first k symbols of $\underline{y}^{(u)}$, consequently $\sigma(\underline{z}^{(u)}) = \underline{0}$. The codeword $\underline{c}$ equals $\sigma(\underline{y}^{(u)})G^{(u)}P^T$.

Define a round as the sequence of Steps 2 through 4. The termination of the RPD algorithm is guaranteed under the same conditions as for the SPD algorithm. It is possible to introduce a *threshold* in the RPD algorithm: Let δ $(1 \leq \delta \leq k)$ be an integer that prevents the rank of K to be lower than δ in any round. If in a round the rank reaches the value $\delta - 1$, then the permutation P in Step 2 is not accepted. For $\delta = k$, the SPD algorithm is obtained. This chapter considers only the RPD without a threshold.

The basic difference between the RPD algorithm and the SPD algorithm is that in Step 2 non-suitable permutations are also accepted (but not verified). Therefore, the rank of the matrix K might be less than k. As a consequence, the definition of state 0 in the RPD algorithm is different from the SPD algorithm and a state -1 is added:

- For $1 \leq l \leq a$, state l corresponds to l errors (as in the SPD algorithm).
- State 0 corresponds to 0 errors, but $rank(K) < k$.
- In state -1, the algorithm terminates (i.e., no errors and $rank(K) = k$).

Therefore, the conditional probabilities $\Pr(-1 \mid 0, u)$, $\Pr(0 \mid 0, u)$ and $\Pr(-1 \mid 1, u)$ also exist, and $\Pr(0 \mid 1)$ depends on the round u. The first conditional probability is given by

$$\begin{aligned}\Pr(-1 \mid 0,u) &= \Pr\{\text{no errors in round } u+1 \mid \text{no errors in round } u\} \\ &\quad \times \Pr\{rank(K^{(u+1)}) = k \mid rank(K^{(u)}) \neq k\}, \\ &= \frac{n-k-t}{n-k} \times \frac{p_{k-1}^{(u)} q_{k-1}^{k}}{1-p_k^{(u)}}.\end{aligned}$$

Similarly, the other conditional probabilities are

$$\begin{aligned}\Pr(0 \mid 0,u) &= \frac{n-k-t}{n-k} \times (1 - \frac{p_{k-1}^{(u)} q_{k-1}^{k}}{1-p_k^{(u)}}), \\ \Pr(-1 \mid 1,u) &= \frac{n-k-t+1}{k(n-k)} \times p_k^{(u+1)},\end{aligned}$$

and

$$\Pr(0 \mid 1,u) = \frac{n-k-t+1}{k(n-k)} \times (1 - p_k^{(u+1)}).$$

Lemma 5.3.5 *Let N_{SPD} be defined as in Lemma 5.2.6. Then the expected number of rounds before Algorithm 5.3.4 terminates is*

$$N_{RPD} \simeq N_{SPD} + (1 - \pi_k) N_0, \tag{5.17}$$

with

$$\mathsf{N}_0 \simeq \frac{1+\Pr(1 \mid 0,\infty)\mathsf{N}_1}{\Pr(-1 \mid 0,\infty)}. \tag{5.18}$$

Proof: For $u > 5$, the probabilities $p_k^{(u)}$ and $p_{k-1}^{(u)}$ are tightly approximated by π_k and π_{k-1}, respectively. Therefore, let $p_k^{(u)} \simeq \pi_k$ and $p_{k-1}^{(u)} \simeq \pi_{k-1}$. With probability π_k, Algorithm 5.3.4 terminates from state N_1, and with probability $1-\pi_k$ it terminates from state N_0. The expected number of rounds for termination is: the weighted sum, of the expected number of rounds, over all possible initial error-states. As a result,

$$\begin{aligned}\mathsf{N}_{\mathrm{RPD}} &\simeq \pi_k \sum_{j=1}^{a} \frac{\binom{k}{j}\binom{n-k}{t-j}}{\binom{n}{t}} \sum_{l=1}^{j} \mathsf{N}_l + (1-\pi_k)\sum_{j=0}^{a} \frac{\binom{k}{j}\binom{n-k}{t-j}}{\binom{n}{t}} \sum_{l=0}^{j} \mathsf{N}_l, \\ &= \sum_{j=1}^{a} \frac{\binom{k}{j}\binom{n-k}{t-j}}{\binom{n}{t}} \sum_{l=1}^{j} \mathsf{N}_l + (1-\pi_k)\sum_{j=0}^{a} \frac{\binom{k}{j}\binom{n-k}{t-j}}{\binom{n}{t}} \mathsf{N}_0.\end{aligned}$$

The first term equals $\mathsf{N}_{\mathrm{SPD}}$ in Lemma (5.2.6), and the second term equals $(1-\pi_k)\mathsf{N}_0$. This proves the lemma. □

Let W_j denote the average work-factor of Step j in Algorithm 5.3.4. In each round, Step 4 is only executed with a probability of approximately π_k (i.e., when $r = k$). Therefore, each round has a work-factor of approximately $(\mathsf{W}_2+\mathsf{W}_3+\pi_k\mathsf{W}_4)$ operations. The algorithm is expected to terminate after $\mathsf{N}_{\mathrm{RPD}}$ rounds. Hence, the expected work-factor becomes

$$\mathsf{W}_{\mathrm{RPD}} \simeq (\mathsf{W}_2+\mathsf{W}_3+\pi_k\mathsf{W}_4)\mathsf{N}_{\mathrm{RPD}} + \mathsf{W}_1 + \mathsf{W}_5. \tag{5.19}$$

Similar to the SPD algorithm, $\mathsf{W}_2 = O(n)$, $\mathsf{W}_3 = O(n^2)$ and $\mathsf{W}_4 = O(n^2)$. Also, $\mathsf{W}_1 = O(n^3)$ and $\mathsf{W}_5 = O(n^3)$ are neglected. Substitution of these work-factors in Equation (5.19) provides an asymptotical estimate for the work-factor of Algorithm 5.3.4 (RPD):

$$\mathsf{W}_{\mathrm{RPD}} = O(n^2)\mathsf{N}_{\mathrm{RPD}}, \quad (n \to \infty). \tag{5.20}$$

The memory-factor is the same as in the SPD algorithm:

$$\mathsf{M}_{\mathrm{RPD}} = O(n^2). \tag{5.21}$$

The following theorem gives the complexity coefficient of the RPD algorithm.

Theorem 5.3.6 (RPD) *For virtually all linear codes C in $C(n,q,R)$, the complexity coefficient of the Restricted Permutation Decoding algorithm for BHDD satisfies*

$$dcc_{\mathsf{RPD}}(q,R) = \widehat{H}_q\left(\frac{H_q^{-1}(1-R)}{2}\right) - (1-R)\widehat{H}_q\left(\frac{H_q^{-1}(1-R)}{2(1-R)}\right).$$

For a memoryless QSC with $p < q^{-1}$, the probability of erroneous decoding equals Pe_{BHDD}.

Proof: Substitution of (5.17) in (5.20) yields

$$\mathsf{W}_{\mathsf{RPD}} \simeq O(n^2)\left(\mathsf{N}_{\mathsf{SPD}} + (1-\pi_k)\mathsf{N}_0\right), \quad (n \to \infty). \tag{5.22}$$

From Equation (5.14), it follows that

$$\mathsf{N}_{\mathsf{SPD}} = O(n^2\sqrt{n})q^{\left(\widehat{\mathsf{H}}_q(\frac{t}{n})-(1-\mathrm{R})\widehat{\mathsf{H}}_q\left(\frac{t}{n(1-\mathrm{R})}\right)\right)}. \tag{5.23}$$

To obtain an expression for N_0 in N_1, note that

$$\mathsf{N}_0 \simeq \lim_{u\to\infty} \frac{1+\Pr(1\mid 0,u)\mathsf{N}_1}{\Pr(-1\mid 0,u)} = \frac{1+(\frac{t}{n-k})\times \mathsf{N}_1}{(1-\frac{t}{n-k})\times\frac{\pi_{k-1}q_{k-1}^k}{1-\pi_k}}. \tag{5.24}$$

The bounds derived in Section A.3 for π_k and π_{k-1} yield for fixed k:

$$\frac{1}{5}\left(1-\frac{1}{k+1}\right) \le \frac{\pi_{k-1}q_{k-1}^k}{1-\pi_k} \le \frac{1}{3}\left(1+\frac{1}{k}\right), \text{ and } \lim_{k\to\infty}\frac{\pi_{k-1}q_{k-1}^k}{1-\pi_k} = 0.203.$$

From these results, it follows that N_0 and N_1 are polynomially related. Substituting (5.23), (5.24) and (A.1) in (5.22) leads to the following expression for the complexity of the RPD algorithm:

$$\mathsf{DC}_{\mathsf{RPD}}(n,q,\mathrm{R}) \simeq O(n^4\sqrt{n})q^{n\left(\widehat{\mathsf{H}}_q(\frac{t}{n})-(1-\mathrm{R})\widehat{\mathsf{H}}_q\left(\frac{t}{n(1-\mathrm{R})}\right)\right)}. \tag{5.25}$$

Note that the memory-factor (5.21) is polynomially bounded. When the algorithm terminates, the probability of erroneous decoding equals Pe_{BHDD}. The theorem follows from (5.25) with $d = n\mathsf{H}_q^{-1}(1-\mathrm{R}) + o(n)$ (Lemma 2.2.2). □

Example 5.3.7 *For a binary, 39-error-correcting* $[1024, 634]$*-code* C*, the work-factor* W_{RPD} *is* $2^{77.0}$*, and the expected number of rounds* N_{RPD} *is* $2^{59.9}$*. Table 5.1 on page 70 gives the number of rounds for state* N_l*. This example also illustrates that although the decoding complexity of this algorithm equals that of SPD, the performance of the SPD algorithm is better.*

5.4 i-Suitable Permutation Decoding

The iSPD algorithm is a generalization of the SPD algorithm, in the sense that up to i columns can be simultaneously swapped in Step 2. The iSPD algorithm can also be viewed as a sequence of i rounds in the RPD algorithm. In the last round, this sequence yields a generator matrix in k-KR-form and leads to the following definition of a *suitable* i*-sequence*:

Definition 5.4.1 (Suitable i**-Sequence)** *A sequence of* i *rounds in the RPD algorithm that yields a generator matrix in* k*-KR-form is called a suitable* i*-sequence.*

A suitable i-sequence corresponds to a suitable permutation that swaps at most i columns in one round in the iSPD algorithm. This algorithm is described as follows:

Algorithm 5.4.2 (i-Suitable Permutation Decoding) *Let* G *be the generator matrix of a* q*-ary* $[n, k, d]$*-code, and let* $\underline{y}$ *be a received vector. The* i*SPD algorithm consists of the following five steps:*

Step 1 *Initialization*

- Randomly select a permutation matrix $P^{(0)}$, such that the first k columns of $GP^{(0)}$ are linearly independent.
- Perform elementary row operations on $GP^{(0)}$ to obtain a matrix in systematic form $G^{(0)} := (I_k \mid A^{(0)})$.
- Let $\underline{y}^{(0)} := \underline{y}P^{(0)}$, $u := 0$ and $P := P^{(0)}$. Proceed to Step 4.

Step 2 *Permutation Selection*

- Select a suitable permutation $P^{(u)}$ that swaps at most i columns, and let $P := PP^{(u)}$.

Step 3 *Swapping and Updating*

- Transform $G^{(u-1)}P^{(u)}$ into $G^{(u)} = (I_k \mid A^{(u)})$, and let $\underline{y}^{(u)} := \underline{y}^{(u-1)}P^{(u)}$.

Step 4 *Verification*

- If $\mathsf{d}_\mathrm{H}(\sigma(\underline{y}^{(u)})A^{(u)}, \tau(\underline{y}^{(u)})) \leq \lfloor (d-1)/2 \rfloor$ is false, then $u := u+1$, and return to Step 2.

Step 5 *Codeword Calculation*

- At this stage, there should be no errors in the first k symbols of $\underline{y}^{(u)}$, consequently $\sigma(\underline{z}^{(u)}) = \underline{0}$. The codeword $\underline{c}$ equals $\sigma(\underline{y}^{(u)})G^{(u)}P^T$.

The termination of the iSPD algorithm is guaranteed under the same conditions as for the SPD algorithm.

Define a round as Step 2 through Step 4. In Step 2, the selected permutation $P^{(u)}$ is suitable, so that, the first k columns of $G^{(u)}$ are linearly independent. Consider τ rounds in the iSPD algorithm. When interpreted as τ suitable i-sequences, the iSPD algorithm can also be viewed as an extension of the RPD algorithm with $rank(K^{(u)}) = k$ for $u = i$. Thus the complexity of the iSPD algorithm can be derived from the complexity of the RPD algorithm.

With probability $p_k^{(i)}$, the matrix $G^{(u)}$ in round $u = i$ of the RPD algorithm is in systematic form, so that, verification can take place. Therefore, in this model on the average $i \times \mathsf{N}_{i\mathrm{SPD}} = \mathsf{N}_{\mathrm{RPD}}/p_k^{(i)}$, where $\mathsf{N}_{\mathrm{RPD}}$ is computed with i instead of ∞. This leads to the following lemma:

Lemma 5.4.3 *Let N_{SPD} be defined as in Lemma 5.2.6. Then the expected number of rounds before Algorithm 5.4.2 terminates is*

$$\mathsf{N}_{iSPD} = \frac{\mathsf{N}_{RPD}}{ip_k^{(i)}} = \frac{\mathsf{N}_{SPD}}{ip_k^{(i)}} + \frac{(1-p_k^{(i)})}{ip_k^{(i)}}\mathsf{N}_0, \quad 1 < i \leq k, \tag{5.26}$$

with

$$\mathsf{N}_0 = \frac{1 + \Pr(1 \mid 0, i)\mathsf{N}_1}{\Pr(-1 \mid 0, i)}. \tag{5.27}$$

For $i > 5$, the probability $p_k^{(i)}$ is tightly approximated by π_k. For $i = 1$, the iSPD algorithm is the same as the SPD algorithm. However, the number of rounds in (5.26) differs from (5.10) due to the introduction of the zero state. For the SPD algorithm, the zero state corresponds to a non-suitable permutation and is not accepted (i.e., the contribution of the zero state must be removed from (5.26)).

Let W_j denote the average work-factor of Step j in Algorithm 5.4.2. Each round has a work-factor of $\mathsf{W}_2+\mathsf{W}_3+\mathsf{W}_4$. After $\mathsf{N}_{i\mathrm{SPD}}$ rounds the algorithm is expected to terminate. Since Steps 1 and 5 are executed only once, the average work-factor for the SPD algorithm becomes

$$\mathsf{W}_{i\mathrm{SPD}} = (\mathsf{W}_2 + \mathsf{W}_3 + \mathsf{W}_4)\mathsf{N}_{i\mathrm{SPD}} + \mathsf{W}_1 + \mathsf{W}_5. \tag{5.28}$$

With probability approximately $p_k^{(i)}$ $(2/7 \leq p_k^{(i)} \leq 1/2)$, a suitable i-sequence is found in Step 2. The work-factors for each step are: $\mathsf{W}_2 = O(n^3)$, $\mathsf{W}_3 = O(n^3)$ and $\mathsf{W}_4 = O(n^2)$. Since Steps 1 and 5 are executed only once (like in the SPD algorithm), $\mathsf{W}_1 = O(n^3)$ and $\mathsf{W}_5 = O(n^3)$ are neglected. Substitution of these work-factors in Equation (5.28) provides an asymptotical estimate for the iSPD algorithm's work-factor:

$$\begin{aligned}\mathsf{W}_{i\mathrm{SPD}} &= O(n^3)\mathsf{N}_{i\mathrm{SPD}}, \quad (n \to \infty) && (5.29)\\ &= O(n^3)\frac{\mathsf{N}_{\mathrm{RPD}}}{ip_k^{(i)}} = O(n^3)\mathsf{N}_{\mathrm{RPD}}, \quad (n \to \infty). && (5.30)\end{aligned}$$

The memory-factor is the same as for the SPD algorithm:

$$\mathsf{M}_{i\mathrm{SPD}} = O(n^2) \tag{5.31}$$

and is polynomially bounded. The following theorem, which follows directly from Theorem 5.3.6, gives the complexity coefficient of the iSPD algorithm:

Theorem 5.4.4 (iSPD) *For virtually all linear codes $\mathcal{C}$ in $\mathcal{C}(n,q,R)$, the complexity coefficient of the i-Suitable Permutation Decoding algorithm $(1 \leq i \leq k)$ for BHDD satisfies*

$$dcc_{i\mathrm{SPD}}(q,R) = \widehat{H}_q\left(\frac{H_q^{-1}(1-R)}{2}\right) - (1-R)\widehat{H}_q\left(\frac{H_q^{-1}(1-R)}{2(1-R)}\right).$$

For a memoryless QSC with $p < q^{-1}$, the probability of erroneous decoding equals Pe_{BHDD}.

Example 5.4.5 *To compare the number of rounds in the case of SPD, RPD and iSPD, the number of swaps is calculated. For the SPD and RPD algorithms the number of swaps is equal to the number of rounds. The number of swaps in an iSPD algorithm is not necessarily i in each round. The expected number of swaps in a round u, however, is*

$$\hat{\imath} = \left\lceil \sum_{j=1}^{i} \frac{j\binom{k}{j}\binom{n-k}{j}}{\sum_{l=0}^{i}\binom{k}{l}\binom{n-k}{l}} \right\rceil .$$

For a binary, 39-error-correcting [1024, 634]-code $\mathcal{C}$, it is found that $\hat{\imath} = 242$, the expected number of swaps N_{kSPD} is $2^{67.3}$, and the work-factor W_{kSPD} equals $2^{84.2}$. This example illustrates that the performance of the SPD algorithm (or Bit-Swapping algorithm) is the best among all iSPD algorithms.

Chapter 6

Syndrome Decoding

Chapter 5 introduced the Permutation Decoding algorithm and this chapter relates it to a new class of syndrome decoding algorithms. First, section 6.1 establishes the basis for these algorithms and relates them to permutation decoding. Section 6.2 than expands the Permutation Decoding algorithm in terms of the parity-check matrix. The benefits of the new syndrome decoding algorithm is a lower work-factor, and it has the ability to find vectors of a given Hamming weight and syndrome (provided the weight is less than the minimum distance of the code). Therefore, the new syndrome-decoding algorithms yield a direct way to examine the security of identification and authentication schemes based on syndrome-decoding problems (e.g., [Har89, Ste90] and [Ste94]). It can also be used to examine public-key cryptosystems, such as [McE78] and [Nie86]. Section 6.3 describes how the new syndrome-decoding algorithms yield a net reduction in the work-factor by processing several syndromes as a batch of syndromes. A slight structure modification of the batch description yields a syndrome-decoding algorithm with several simple, dedicated circuits (in parallel). This further reduces the work-factor as described in Section 6.4.

This chapter is based on joint work with Peter Roelse [RT94].

6.1 Preliminaries

This section establishes the relationship between the permutation decoding, as described in Chapter 5, and syndrome decoding. Also, it proves a theorem that will be used as a verification step for the new syndrome-decoding algorithms described in this chapter.

The Permutation Decoding algorithms are based on the concept of searching for error-free information sets. Let $\mathcal{C}$ be a q-ary $[n,k,d]$-code with generator matrix G. Recall that an information set $\mathcal{I}$ is a k-subset of $\{1,2,\ldots,n\}$, such that, the restriction of $\mathcal{C}$ to these coordinates still has dimension k (i.e., the restriction of G to these coordinates still has full rank k). Note that $\underline{y} = \underline{c} + \underline{z} = \underline{x}G + \underline{z}$ for some $\underline{x} \in \mathrm{GF}(q)^k$, therefore, if the locations of the vector $\underline{y}$ corresponding to $\mathcal{I}$ are error free, then $\underline{x}$ is revealed (e.g., by Gaussian elimination) and the codeword $\underline{c}$ follows from $\underline{x}G$. The permutation decoding algorithms are a combination of Algorithm 4.2.1 (codewords) and Algorithm 4.2.5 (error vectors) exhaustive-search decoding algorithms.

Using constantly renewed information sets, the permutation decoding algorithm generates codewords and compares them to the received vector. The generation of these codewords depends on the error-freeness of the selected information-set symbols in the received word (i.e., the $\mathcal{I}$-restricted set of error vectors). For large rate R, the set of $\mathcal{I}$-restricted codewords dominates; while for small rate R, the $\mathcal{I}$-restricted set of error vectors dominates. The permutation decoding algorithm terminates when a permutation matrix P is found, such that, the first k columns of the matrix GP are linearly independent and the corresponding symbols in $\underline{y}P$ are error-free (i.e., an error-free information set). This termination criterion can be rewritten in terms of the parity-check matrix as follows:

Theorem 6.1.1 *Let H be an $(n-k)\times n$ parity-check matrix of a q-ary $[n,k,d]$-code $\mathcal{C}$. Let P be an $n \times n$ permutation matrix, such that, the last $(n-k)$ columns of the matrix HP are linearly independent over $GF(q)$. Let S be an $(n-k)\times(n-k)$ matrix, such that, $SHP = (A \mid I_{n-k})$ where A is an $(n-k)\times k$ matrix. If $\underline{y} = \underline{c} + \underline{z}$ with $\underline{c} \in \mathcal{C}$, $w_H(\underline{z}) \leq \lfloor (d-1)/2 \rfloor$ and $\underline{s} = \underline{y}H^T$, then it holds that*

$$w_H(\underline{s}S^T) \leq \lfloor (d-1)/2 \rfloor \quad \Longleftrightarrow \quad \underline{z} = (\underline{0} \mid \underline{s}S^T)P^T.$$

Proof: Since $\underline{s} = \underline{y}H^T = \underline{z}H^T = \underline{z}PP^TH^T$, the syndrome of $\underline{y}P$ with respect to the matrix SHP equals $\underline{s}S^T \Leftrightarrow \underline{z}P(SHP)^T = (\underline{0} \mid \underline{s}S^T)(A \mid I_{n-k})^T$. Therefore, it follows that $\underline{y}P$ and $(\underline{0} \mid \underline{s}S^T)$ are in the same coset. Because $\mathrm{w_H}(\underline{0} \mid \underline{s}S^T) = \mathrm{w_H}(\underline{s}S^T) \leq \lfloor (d-1)/2 \rfloor$ it holds that $\underline{y}P = (\underline{0} \mid \underline{s}S^T)$. Note that a coset contains at most one vector of weight $\leq \lfloor (d-1)/2 \rfloor$.

On the other hand, it holds that $\mathrm{w_H}(\underline{s}S^T) = \mathrm{w_H}(\underline{0} \mid \underline{s}S^T) = \mathrm{w_H}(\underline{z})$ which is $\leq \lfloor (d-1)/2 \rfloor$ by assumption. □

Based on this theorem, the new decoding algorithm searches for a permutation P, such that, the last $n-k$ columns of the matrix HP are linearly independent and the corresponding locations in $\underline{y}P$ contain all the errors. In the initial phase of the syndrome-decoding algorithms, $n-k$ linearly independent columns of the parity-check matrix, together with the corresponding symbols from $\underline{y}$, are randomly selected. Let the $n-k$ symbols define the subset $\mathcal{A}$ of $\{1, 2, \ldots, n\}$, and the remaining k symbols form set $\mathcal{B}$.

Definition 6.1.2 (Swap) *Let $(A \mid I_{n-k})$ be a $(n-k) \times n$ matrix. A swap permutes only one column from the I_{n-k}-part of the matrix for one column in the A-part.*

Thus, selecting new symbols corresponds to a permutation that swaps new symbols from set $\mathcal{A}$ for symbols from set $\mathcal{B}$. The permutation is only successful if, the columns of the parity-check matrix, which correspond to the new set $\mathcal{B}$, are linearly independent. A permutation that fulfills this condition is called a *suitable permutation.*

Definition 6.1.3 (Suitable Permutation) *Let $(A \mid I_{n-k})$ be an $(n-k) \times n$ matrix. A permutation P is suitable if the last $n-k$ columns of matrix $(A \mid I_{n-k})P$ form a non-singular submatrix (i.e., they are linearly independent).*

The difference between the generator matrix description and the parity-check matrix description is: The first checks information sets for error-freeness, while the second checks the received vector for error vectors. The next theorem provides a relationship between the two algorithms:

Theorem 6.1.4 *Let $G = (I_k \mid A)$ be a generator matrix of $\mathcal{C}$, and let $H = (-A^T \mid I_{n-k})$ be a corresponding parity-check matrix. If $\underline{y}$ is the received vector, then*

$$d_H(\underline{y}, \sigma(\underline{y})G) = w_H(\underline{y}H^T).$$

Proof: The distance between the received vector $\underline{y}$ and the codeword, as defined by the first k information symbols of $\underline{y}$, equals

$$d_H(\underline{y}, \sigma(\underline{y})G) = d_H(\tau(\underline{y}), \sigma(\underline{y})A) = w_H(\tau(\underline{y}) - \sigma(\underline{y})A) = w_H(\underline{y}H^T),$$

which is the Hamming weight of the syndrome of $\underline{y}$ with respect to H of $\mathcal{C}$. □

6.2 One-Swap Syndrome Decoding

In this section, the 1SSD algorithm is described and its work-factor is obtained. Moreover, it is shown that its complexity is equal to that of information-set decoding. The 1SSD algorithm can be described as follows:

Algorithm 6.2.1 (One-Swap Syndrome Decoding) *Let H be a parity-check matrix of a q-ary $[n, k, d]$-code, and let $\underline{y}$ be a received vector. The 1SSD algorithm consists of the following five steps:*

Step 1 *Initialization*

- Randomly select a permutation matrix $P^{(0)}$, such that, the last $n-k$ columns of the matrix $HP^{(0)}$ are linearly independent.
- Perform elementary row operations on $HP^{(0)}$ to obtain a matrix in systematic form $H^{(0)} := (A^{(0)} \mid I_{n-k})$.
- Compute the syndrome $\underline{s}^{(0)} := \underline{y}H^{(0)^T}$, $P := P^{(0)}$, and let $u := 0$. Proceed to Step 4.

Step 2 *Permutation Selection*

- Select a suitable permutation $P^{(u)}$ that swaps only one symbol, and let $P := PP^{(u)}$.

Step 3 *Swapping and Updating*

- Transform $(H^{(u-1)}P^{(u)} \mid \underline{s}^{(u-1)^T})$ into $(A^{(u)} \mid I_{n-k} \mid \underline{s}^{(u)^T})$. Assign $H^{(u)} := (A^{(u)} \mid I_{n-k})$.

Step 4 *Verification*

- If $\mathsf{w_H}(\underline{s}^{(u)}) \leq \lfloor (d-1)/2 \rfloor$ is false, then let $u := u+1$ and return to Step 2, otherwise, compute the error vector as $\underline{z} = (\underline{0} \mid \underline{s}^{(u)})P^T$.

Step 5 *Codeword Calculation*

- Compute the codeword as $\underline{c} = \underline{y} - \underline{z}$.

Define a *round* as Step 2 through Step 4. Then the 1SSD algorithm can be viewed as a sequence of rounds, each yielding a parity-check matrix in $(A \mid I)$ form. In the initial round, the matrix $H^{(0)} = S^{(0)}HP^{(0)}$ is obtained. In round u, Algorithm 6.2.1 selects a suitable permutation $P^{(u)}$, such that

$$\begin{aligned} H^{(u)} &= S^{(u)}H^{(u-1)}P^{(u)}, \\ &= S^{(u)}S^{(u-1)}\ldots S^{(0)}HP^{(0)}\ldots P^{(u-1)}P^{(u)}, \\ &= (A^{(u)} \mid I_{n-k}). \end{aligned}$$

Any parity-check matrix in $(A \mid I)$-form that corresponds to a generator matrix in systematic form (generating a code equivalent to the code generated by G) can occur in a round. A sufficient condition for termination of the 1SSD algorithm is to verify each permutation P at least once. Moreover, it will terminate in round e with a unique codeword $\underline{c} = \underline{y} - (\underline{0} \mid \underline{s}^{(e)})P^T$, provided that

$$\mathsf{w_H}(\underline{s}S^T) = \mathsf{w_H}(\underline{s}^{(e)}) \leq \lfloor (d-1)/2 \rfloor,$$

where $S = S^{(e)}S^{(e-1)}\ldots S^{(0)}$. In practice, the permutations used by the algorithm are neither tracked nor stored.

Note that it is not necessary to compute $\underline{s}^{(u)} = \underline{s}^{(u-1)}S^{(u)^T}$ using a matrix multiplication. In Step 3, the new syndrome $\underline{s}^{(u)}$ is obtained by performing the same elementary row operations on $\underline{s}^{(u-1)}$ as were performed on $H^{(u-1)}P^{(u)}$.

In the initial phase of the 1SSD algorithm, exactly $n-k$ linearly independent columns of the parity-check matrix, together with the corresponding symbols from $\underline{y}$, are randomly selected. Let the $n-k$ symbols define the subset $\mathcal{A}$ of $\{1,2,\ldots,n\}$, and let the remaining k symbols form set $\mathcal{B}$. The suitable permutation in Algorithm 6.2.1 swaps one symbol from set $\mathcal{B}$ for one symbol from set $\mathcal{A}$. The algorithm terminates when all t error-locations are in set $\mathcal{A}$.

Without loss of generality, let $t = \lfloor (d-1)/2 \rfloor$. Let l denote the number of error-locations in set $\mathcal{A}$ (i.e., the error-state number). Also, let $\Pr(l+s \mid l)$ with $s \in \{-1,0,1\}$, denote the probability that $l+s$ locations in set $\mathcal{B}$ are in error after a swap, given that l of them were in error prior to the swap. Algorithm 6.2.1 starts in an error-state with the number of error-locations in set $\mathcal{B}$ between 0 and $a := \min\{t,k\}$. For $1 \le l \le a$, the conditional probabilities satisfy:

$$\begin{aligned} \Pr(l-1 \mid l) &= \frac{l(n-k-t+l)}{k(n-k)}, && (6.1) \\ \Pr(l+1 \mid l) &= \frac{(k-l)(t-l)}{k(n-k)}, && (6.2) \\ \Pr(l \mid l) &= 1 - \Pr(l-1 \mid l) - \Pr(l+1 \mid l). && (6.3) \end{aligned}$$

Lemma 6.2.2 *Within Algorithm 6.2.1, let* N_l $(1 \le l \le a)$ *denote the expected number of rounds (swaps) needed to pass from a state of* l *errors to a state of* $l-1$ *errors. Then* $\mathsf{N}_a = \Pr(a-1 \mid a)^{-1}$, *and each* N_l *can be computed recursively by:*

$$\mathsf{N}_l = \frac{1 + \Pr(l+1 \mid l)\mathsf{N}_{l+1}}{\Pr(l-1 \mid l)}.$$

Proof: Let l be the error-state of Algorithm 6.2.1. With probability $\Pr(l-1 \mid l)$, the algorithm leaves state l and advances to state $l-1$ after one round. With probability $\Pr(l \mid l)$, the algorithm stays in state l after one round and reaches state $l-1$ in an expected number of $1+\mathsf{N}_l$ rounds. With probability $\Pr(l+1 \mid l)$, the algorithm advances to state $l+1$. In this case, the expected number of rounds needed to proceed to state $l-1$ is $1+\mathsf{N}_{l+1}+\mathsf{N}_l$. Therefore,

$$\mathsf{N}_l = 1 \cdot \Pr(l-1 \mid l) + (1+\mathsf{N}_l) \cdot \Pr(l \mid l) + (1+\mathsf{N}_l+\mathsf{N}_{l+1}) \cdot \Pr(l+1 \mid l)$$

is the expected number of rounds in state l. Then, after reordering the terms and applying (6.3) twice, the lemma follows. □

When the round needed to advance to state $l-1$ is subtracted from N_l, then the round definition and results as stated in [Til90] are obtained. For Algorithm 6.2.1, the expected number of rounds for termination is: the weighted sum, of the expected number of rounds, over all possible initial error-states. Each one of these error-states $j > 0$ has an expected number of $\mathsf{N}_j + \mathsf{N}_{j-1} + \ldots + \mathsf{N}_1$ rounds, with a corresponding weight factor of $\binom{k}{j}\binom{n-k}{t-j}/\binom{n}{t}$. Thus, the expected number of rounds before Algorithm 6.2.1 terminates is

$$\mathsf{N}_{1\mathrm{SSD}} = \sum_{j=1}^{a} \frac{\binom{k}{j}\binom{n-k}{t-j}}{\binom{n}{t}} \sum_{l=1}^{j} \mathsf{N}_l. \tag{6.4}$$

The expected number of rounds $\mathsf{N}_{1\mathrm{SSD}}$ is not sensitive to the number of errors in the initial error-state. For sufficiently large n and $a > 1$, the number of rounds in (6.4) is dominated by N_1, so that

$$\sum_{l=1}^{j} \mathsf{N}_l \approx \mathsf{N}_1, \quad \text{for } 2 \leq j \leq a,$$

and therefore

$$\mathsf{N}_{1\mathrm{SSD}} \approx \mathsf{N}_1 \sum_{j=1}^{a} \frac{\binom{k}{j}\binom{n-k}{t-j}}{\binom{n}{t}} = \mathsf{N}_1.$$

The algorithm terminates when the set $\mathcal{B}$ contains no error-locations of the error vector $\underline{z}$. The number of rounds needed to accomplish this is dominated by N_1 (i.e., removing the last error-location from the set $\mathcal{B}$). As a result, an error search in set $\mathcal{B}$ reduces the number of possible error-states and, consequently, the expected number of rounds before Algorithm 6.2.1 terminates.

When an error search for w errors is added in each round, then the expected number of rounds before Algorithm 6.4.1 terminates is given by (6.4) and the summations start at w instead of at one. Moreover, it is only necessary in the initialization (i.e., round $u = 0$) to verify whether there are w or less errors. For rounds $u > 0$, no more than one error-location can be swapped from set $\mathcal{A}$ to set $\mathcal{B}$ and vice versa. Therefore, when a one-symbol swap suitable permutation is used, it is not necessary to verify for less than w errors. This proves the following lemma:

Lemma 6.2.3 *Let* N_l $(1 \leq l \leq a)$ *be defined as in Lemma 6.2.2. Then the expected number of rounds before Algorithm 6.2.1 with a* w*-error search terminates is*

$$\mathsf{N}^{(w)}_{1SSD} = \sum_{j=w+1}^{a} \frac{\binom{k}{j}\binom{n-k}{t-j}}{\binom{n}{t}} \sum_{l=w+1}^{j} \mathsf{N}_l. \qquad (6.5)$$

Suppose only $\lfloor (d-1)/2 \rfloor - 1$ error-locations of the $\lfloor (d-1)/2 \rfloor$ errors correspond to the columns of the I-part. Then the last error-location corresponds to the column for which the Hamming distance between a column of the A-part and the syndrome is $\leq \lfloor (d-1)/2 \rfloor - 1$.

Let $\underline{a}_j^{(u)^T}$ denote the j-th column in the matrix $A^{(u)}$, and let $\underline{u}_j$ denote the j-th k-bit unit vector. Then replace Step 4 in Algorithm 6.2.1 with:

- If for a j $(1 \leq j \leq k)$ and an $\alpha \in \mathrm{GF}(q) \setminus \{0\}$, it holds that

  $$\mathrm{d_H}(\alpha \underline{s}^{(u)}, \underline{a}_j^{(u)}) \leq \lfloor (d-1)/2 \rfloor - 1,$$

  then the error vector equals $\underline{z} = (\alpha^{-1}\underline{u}_j \mid \underline{s}^{(u)})P^T$ and proceed to Step 5, otherwise, let $u := u + 1$ and return to Step 2.

Because most applications relate to the binary case, the work-factors in this chapter are given in bit operations only. The work-factors for the q-ary case can also be computed in the same way. The work-factors give an *indication* of the amount of bit operations that are needed and are used to compare the algorithms. In agreement with [AM89], [LB88] and [Til90], the Gaussian elimination is applied in a straightforward manner.

Let W_j denote the average work-factor of Step j in Algorithm 6.2.1. Each round has an average of $\mathsf{W}_2 + \mathsf{W}_3 + \mathsf{W}_4$ bit operations. After N_{1SSD} rounds, the algorithm is expected to terminate. Since Steps 1 and 5 are executed only once, the average work-factor for the 1SSD algorithm becomes

$$\mathsf{W}_{1\mathrm{SSD}} = (\mathsf{W}_2 + \mathsf{W}_3 + \mathsf{W}_4)\mathsf{N}_{1\mathrm{SSD}} + \mathsf{W}_1 + \mathsf{W}_5. \tag{6.6}$$

A pointer-table is used to track the column permutations of the matrix $H^{(u)}$. With approximately one-half probability, a suitable permutation is found in Step 2. A suitable permutation is recognized in an efficient way (one bit-check, the reader is referred to page 68), hence the work-factor W_2 is neglectable. The algorithm returns to Step 2 from Step 4 when the weight of the syndrome is at least $t+1$. The vector $\underline{s}^{(u)}$ is assumed to be random. Therefore, on the average, Step 4 requires $2(t+1)$ bit operations. In Step 3, the elementary row operations require $(n-k-1)$ bit verifications to check whether a coordinate in the swapped column (from the submatrix $A^{(u-1)}$) is nonzero. With approximately one-half probability, a coordinate of this swapped column is one, so that it also requires a k-bit addition. Therefore, the work-factor W_3 involves approximately $(n-k-1)+k(n-k-1)/2$ bit operations. Since Steps 1 and 5 are executed only once, W_1 and W_5 are neglectable. Substitution of the work-factors in (6.6) provides the following estimate of the overall work-factor:

$$\mathsf{W}_{1\mathrm{SSD}} \approx \left(2(t+1) + (\frac{k}{2}+1)(n-k-1)\right)\mathsf{N}_{1\mathrm{SSD}}.$$

If Step 4 is replaced by a one-error search, then the algorithm returns to Step 2 when the distance verification for all of the k columns in the submatrix $A^{(u)}$ is at least t. With approximately one-half probability, each coordinate of submatrix $A^{(u)}$ is one, so that $2t$ checks per column on the average are needed. Therefore, the work-factor W_4 becomes $2kt$. The expected work-factor for Algorithm 6.2.1 with a one-error search becomes

$$\mathsf{W}^{(1)}_{1\mathrm{SSD}} \approx \left(2kt + (\frac{k}{2}+1)(n-k-1)\right)\mathsf{N}^{(1)}_{1\mathrm{SSD}}.$$

Example 6.2.4 *Table 6.1 illustrates the security of some values of the parameters when used in the McEliece or Niederreiter public-key cryptosystem. The table indicates that the optimal value is not critical.*

n	k	t	$\log_2 \mathsf{N}_{1\mathrm{SSD}}$	$\log_2 \mathsf{W}_{1\mathrm{SSD}}$	$\log_2 \mathsf{N}^{(1)}_{1\mathrm{SSD}}$	$\log_2 \mathsf{W}^{(1)}_{1\mathrm{SSD}}$
1024	654	37	59.4	76.4	53.3	70.7
1024	644	38	59.4	76.4	53.4	70.7
1024	634	39	59.4	76.4	53.4	70.8
1024	624	40	59.4	76.3	53.3	70.8
1024	614	41	59.3	76.2	53.3	70.7
1024	524	50	57.0	74.0	51.2	68.6

Table 6.1: The expected number of rounds and bit operations for Algorithm 6.2.1 (1SSD) with and without a one-error search.

In order to evaluate authentication and identification schemes, the weight condition in Step 4 of Algorithm 6.2.1 is replaced by $\mathrm{w_H}(\underline{s}^{(u)}) = w$. The input is the syndrome $\underline{s}$, and the output is the error vector. If d is the minimum distance of the code and $w \leq d$, then the error-locations correspond to independent columns of $H^{(u)}$, and the decoding algorithm can be applied.

Example 6.2.5 *Table 6.2 illustrates the security of the cryptosystems [Har89, Ste90] and [Ste94] for some values of the parameters n, k and w. The table also indicates that for identification schemes, it is more effective to increase w while keeping n and k fixed, instead of increasing the values n and k while keeping w fixed. If $w \leq \lfloor (d-1)/2 \rfloor$, then collision of public-keys (i.e., different secret keys with the same syndrome) is impossible. On the other hand, when w increases and $w > \lfloor (d-1)/2 \rfloor$, then is more likely that a coset contains more vectors of weight w. Each key in the collision can be used to impersonate a user, therefore key collisions result in a decreased work-factor. However, finding d for a random code is conjectured to be $\mathcal{NP}$-complete [BMT78]. Since d is not known, the values for the schemes in Table 6.2 are upper-bounds.*

A closer look at the 1SSD algorithm shows: Only the parity-check matrix H, the vector $\underline{y}$, and the permutation matrix P need to be stored. Therefore, given the number q for a q-ary alphabet, the memory-factor is

$$\mathsf{M} = \gamma n^2 \mathrm{R} q, \tag{6.7}$$

n	k	w	$\log_2 \mathsf{N}_{1\mathrm{SSD}}$	$\log_2 \mathsf{W}_{1\mathrm{SSD}}$	$\log_2 \mathsf{N}^{(1)}_{1\mathrm{SSD}}$	$\log_2 \mathsf{W}^{(1)}_{1\mathrm{SSD}}$
512	256	30	34.4	49.4	29.5	45.0
512	256	56	63.1	78.1	57.0	72.9
1000	500	30	34.8	51.7	29.9	47.1
1024	512	40	44.9	61.9	39.5	56.9
1024	512	110	121.7	138.7	114.6	132.5

Table 6.2: The expected number of rounds and bit operations for Algorithm 6.2.1 (1SSD) with and without a one-error search.

where γ is a constant. The size n of the q-ary representation of a codeword depends on the choice of q (i.e., n decreases as q increases and vice versa). In order to minimize memory-factor, an optimal number system is needed, such that M is minimal. In other words, find the optimal number q and corresponding n, such that, M is minimal for fixed R. Define $x = q^{n^2\mathrm{R}}$, then

$$n^2\mathrm{R} = \frac{\log x}{\log q}, \tag{6.8}$$

where the logarithm base is arbitrary. Substituting (6.8) in M yields

$$\mathsf{M} = \gamma \frac{q \log x}{\log q}.$$

The minimum value for M follows from $\mathrm{d}M/\mathrm{d}q = 0$, that is

$$\frac{(\log q - \log e)\log x}{(\log q)^2} = 0.$$

The minimum is achieved for $q_0 = e \approx 2.718$. For the binary or ternary type 1SSD algorithm, the memory-factor is minimal. Using similar reasoning, the *least* optimal number system for memory-factor is obtained for $q \to \infty$.

The following theorem gives the complexity coefficient of the 1SSD algorithm.

Theorem 6.2.6 (1SSD) *For virtually all linear codes $\mathcal{C}$ in $\mathcal{C}(n, q, R)$, the complexity coefficient of the one-swap, syndrome-decoding algorithm for BHDD satisfies*

$$dcc_{1SSD}(q, R) = \widehat{H}_q\left(\frac{H_q^{-1}(1-R)}{2}\right) - (1-R)\widehat{H}_q\left(\frac{H_q^{-1}(1-R)}{2(1-R)}\right).$$

For a memoryless QSC with $p < q^{-1}$, the probability of erroneous decoding equals Pe_{BHDD}.

Proof: From (6.7), it follows that the memory-factor $\mathsf{M}_{1\mathrm{SSD}}$ equals $O(n^2)$. The work-factor in (6.6) is upper-bounded by $\mathsf{W}_{1\mathrm{SSD}} = O(n^2)\mathsf{N}_{1\mathrm{SSD}}$. Since $\mathsf{N}_{1\mathrm{SSD}} = \mathsf{N}_{\mathrm{SPD}}$, then

$$\mathsf{DC}_{1\mathrm{SSD}}(n, q, \mathrm{R}) = O(n^4)\mathsf{N}_{1\mathrm{SSD}} = O(n^4)\mathsf{N}_{\mathrm{SPD}}. \qquad (6.9)$$

Substituting (5.14) and $d = n\mathrm{H}_q^{-1}(1-\mathrm{R}) + o(n)$ (Lemma 2.2.2) into (6.9) and (2.11) yields the expression for the complexity coefficient. When the algorithm terminates, the probability of erroneous decoding equals Pe_{BHDD}. □

6.3 Batch Syndrome Decoding

An advantage of the parity-check matrix description of the PD algorithm is that Step 4 only involves a weight computation. The generator matrix description [Til90], on the other hand, also requires a vector-matrix multiplication. Algorithm 6.2.1 can be expanded to process a batch of syndromes (e.g., in an identification scheme with multiple users, several public-keys can be examined at the same time). The net result is a reduced work-factor for finding a solution of at least one syndrome (public-key) in each batch. This one solution is enough to discredit the chosen values of the system parameters. Note that for the McEliece scheme, this corresponds to processing a batch of ciphertexts.

Let C be a q-ary $[n, k, d]$-code with parity-check matrix H. For $1 \leq i \leq l$, let $\underline{y}_i = \underline{c}_i + \underline{z}_i$ be the i-th received vector. Without loss of generality, suppose that the l syndromes (i.e., $\underline{s}_i = \underline{y}_i H^T$ with $1 \leq i \leq l$)

are all distinct. Let the $l \times (n-k)$ matrix B and the $l \times n$ matrix Y be defined as follows:

$$B := \begin{pmatrix} \underline{s}_1 \\ \underline{s}_2 \\ \vdots \\ \underline{s}_l \end{pmatrix} \quad \text{and} \quad Y := \begin{pmatrix} \underline{y}_1 \\ \underline{y}_2 \\ \vdots \\ \underline{y}_l \end{pmatrix}, \quad \text{so that} \quad B = YH^T.$$

Algorithm 6.3.1 (Batch Syndrome Decoding) *Let H be a parity-check matrix of a q-ary $[n,k,d]$-code, and let Y be a received vector matrix. The BSD algorithm consists of the following five steps:*

Step 1 *Initialization*

- Randomly select a permutation matrix $P^{(0)}$, such that, the last $n-k$ columns of the matrix $HP^{(0)}$ are linearly independent.

- Perform elementary row operations on $HP^{(0)}$ in order to obtain a matrix in systematic form $H^{(0)} := (A^{(0)} \mid I_{n-k})$.

- Compute the syndrome matrix $B^{(0)} := YH^{(0)^T}$ and $P := P^{(0)}$. Let $u := 0$, and proceed to Step 4.

Step 2 *Permutation Selection*

- Select a suitable permutation $P^{(u)}$, and let $P := PP^{(u)}$.

Step 3 *Swapping and Updating*

- Transform $(H^{(u-1)}P^{(u)} \mid B^{(u-1)^T})$ into $(A^{(u)} \mid I_{n-k} \mid B^{(u)^T})$. Assign $H^{(u)} := (A^{(u)} \mid I_{n-k})$.

Step 4 *Verification*

- Let $\underline{s}_j^{(u)}$ denote the j-th row of $B^{(u)}$. If $\mathsf{w}_{\mathrm{H}}(\underline{s}_j^{(u)}) \leq \lfloor (d-1)/2 \rfloor$ is false for all j $(1 \leq j \leq l)$, then let $u := u+1$ and return to Step 2, otherwise, compute the error vector as $\underline{z}_j = (\underline{0} \mid \underline{s}_j^{(u)})P^T$.

Step 5 *Codeword Calculation*

- Compute the codeword as $\underline{c}_j = \underline{y}_j - \underline{z}_j$.

When the permutation selection is restricted to one-swap suitable permutations and applied to only one syndrome ($l = 1$), the 1SSD algorithm is obtained.

For sufficiently large l, the work-factor of a round is dominated by Steps 3 and 4 (even when Step 2 is not restricted to suitable one-swap permutations). Therefore, in the following analysis, an $(n-k)$-permutation is examined (i.e., in each round, $n-k$ columns of the matrix H are randomly selected). The algorithm terminates if at least one of the l vectors $\underline{y}_j$ $(1 \leq j \leq l)$ has all of its errors in the corresponding $n-k$ positions.

Lemma 6.3.2 *Suppose all l error vectors are distinct and have Hamming weight t. If $\binom{n}{t} - l + 1 \geq \binom{n-k}{t}$, then the expected number of rounds before Algorithm 6.3.1 terminates is*

$$N_{BSD}(l) = \left[1 - \prod_{i=0}^{l-1}\left(1 - \frac{\binom{n-k}{t}}{\binom{n}{t} - i}\right)\right]^{-1}, \tag{6.10}$$

otherwise, $N_{BSD}(l) = 1$.

Proof: There are $\binom{n}{t}$ possible error patterns of weight t of which $\binom{n-k}{t}$ error patterns lead to termination. Therefore, the algorithm terminates in the first round when $\binom{n}{t} - l + 1 < \binom{n-k}{t}$ is true.

Suppose $\binom{n}{t} - l + 1 \geq \binom{n-k}{t}$ and none of the $\underline{y}_1 P, \underline{y}_2 P, \ldots, \underline{y}_i P$ vectors have all their errors in the last $n-k$ positions. Then the i-th factor in the product of (6.10) relates to the probability that the vector $\underline{y}_{i+1} P$ does not have all its errors in the last $n-k$ positions. The product denotes the probability that none of the l rows in the matrix YP have all errors in the last $n-k$ positions (i.e., the round was not successful). Note that each round has the same probability of being successful. □

In cryptographic applications, the value of l is negligible compared to the large value of $\binom{n}{t}$. Moreover, the value of $\binom{n-k}{t}$ is considerably smaller than the value of $\binom{n}{t}$. Therefore, the following approximation for (6.10) is used:

$$\mathsf{N}_{\mathrm{BSD}}(l) \approx \frac{\binom{n}{t}}{\binom{n-k}{t} l}. \tag{6.11}$$

Let W_j denote the average work-factor of Step j in Algorithm 6.3.1. In order to show the influence of l on the work-factor, let W_{3a} denote the average work-factor needed to obtain the matrix $H^{(u)} = (A^{(u)} \mid I_{n-k})$ by performing elementary row operations on $H^{(u-1)}P^{(u)}$. Let W_{3b} denote the average work-factor needed to perform elementary row operations on only *one row* of the (syndrome) batch matrix $B^{(u-1)}$ in order to obtain the matrix $B^{(u)}$. Hence, $\mathsf{W}_3(l) = \mathsf{W}_{3a} + \mathsf{W}_{3b}l$. Let W_{4a} denote the average work-factor needed for one verification that a row in $B^{(u)}$ has weight $\leq \lfloor (d-1)/2 \rfloor$, so that $\mathsf{W}_4(l) = \mathsf{W}_{4a}l$. Then each round has an average of $\mathsf{W}_2 + \mathsf{W}_3(l) + \mathsf{W}_4(l)$ bit operations. After $\mathsf{N}_{\mathrm{BSD}}(l)$ rounds, the algorithm is expected to terminate with an average work-factor of

$$\begin{aligned} \mathsf{W}_{\mathrm{BSD}}(l) &= (\mathsf{W}_2 + \mathsf{W}_3(l) + \mathsf{W}_4(l))\, \mathsf{N}_{\mathrm{BSD}}(l) + \mathsf{W}_1 + \mathsf{W}_5 \qquad (6.12)\\ &= (\frac{\mathsf{W}_2 + \mathsf{W}_{3a}}{l} + \mathsf{W}_{3b} + \mathsf{W}_{4a})\mathsf{N}_{\mathrm{BSD}}(1) + \mathsf{W}_1 + \mathsf{W}_5. \end{aligned}$$

Since Steps 1 and 5 are executed only once, their work-factors are negligible. For sufficiently large l, the value of $(\mathsf{W}_2+\mathsf{W}_{3a})/l$ is negligible compared to $\mathsf{W}_{3b} + \mathsf{W}_{4a}$. Then using (6.11) in (6.12), yields

$$\mathsf{W}_{\mathrm{BSD}}(l) \approx (\mathsf{W}_{3b} + \mathsf{W}_{4a})\frac{\binom{n}{t}}{\binom{n-k}{t}} =: \mathsf{W}_{\mathrm{BSD}}. \tag{6.13}$$

To obtain an estimate of W_{3b}, proceed as follows. On the average, the $(n-k)$-permutation $P^{(u)}$ swaps $(n-k)k/n$ columns of the matrix $H^{(u-1)}$ between the I-part and the $A^{(u-1)}$-part. Assume that $A^{(u-1)}$ is a random matrix. Then, with approximately one-half probability, the value of a coordinate in the swapped columns (from $A^{(u-1)}$) is one. Only then does it require l bit operations. For each of the $(n-k)k/n$ swapped columns, approximately $l(n-k-1)/2$ bit operations are required to perform the elementary row operations. Hence, $\mathsf{W}_{3b} = (n-k-1)(n-k)k/2n$. The algorithm returns to Step 2 when the Hamming weight of each of the l rows in the matrix $B^{(u)}$ is at least $t+1$. Therefore, Step 4 requires an average of $2(t+1)l$ bit operations (i.e., $\mathsf{W}_{4a} = 2(t+1)$).

Substitution of W_{3b} and W_{4a} in (6.12) provides an estimation of the overall work-factor of Algorithm 6.3.1:

$$\mathsf{W}_{\mathrm{BSD}} \approx \left(2(t+1) + \frac{k}{2n}(n-k)(n-k-1)\right) \frac{\binom{n}{t}}{\binom{n-k}{t}}. \tag{6.14}$$

To make "for sufficiently large l" more precise, it is necessary to obtain estimates of the work-factors W_2 and W_{3a}. The work-factor W_2 corresponds to finding a suitable $(n-k)$-permutation. With probability $\gamma = \prod_{i=1}^{n-k}(1-2^{-i}) \approx 0.289$, the selected $n-k$ columns of the matrix $H^{(u-1)}$ are linearly independent. Gaussian elimination, for example, can be used to verify this. Assume that $A^{(u-1)}$ is a random matrix. On the average, $(n-k)k/n$ columns of the $n-k$ selected columns are not unit vectors. For each of the $(n-k)k/n$ columns, the elementary row operations require $(n-k-1)$ bit verifications to check whether a coordinate in the swapped column (from the submatrix $A^{(u-1)}$) is nonzero. With approximately one-half probability, a coordinate of this swapped column is one, so that it also requires $k(n-k)/n$ bit operations. Therefore, for each swapped column this involves $(n-k-1)+(n-k-1)k(n-k)/2n$ bit operations, so that

$$\begin{aligned} \mathsf{W}_2 &\approx \frac{k(n-k)}{\gamma n}\left((n-k-1)(1+\frac{k(n-k)}{2n})\right) \\ &\approx \frac{k^2(n-k)^3}{2\gamma n^2}. \end{aligned} \tag{6.15}$$

To give an estimate of W_{3a}, determine the average number of elementary row operations on $H^{(u-1)}P^{(u)}$ needed to obtain the matrix $H^{(u)} = (A^{(u)} \mid I_{n-k})$. In Step 2, exactly $n-k$ linearly independent columns are found. Recall that, on the average, $(n-k)k/n$ columns of the selected $n-k$ columns of the matrix $H^{(u)}$ are not yet unit vectors. With approximately one-half probability, the coordinate value of such a swapped column is one, so that besides the bit verification, $k(n-k)/n+k$ bit operations are also required. An estimate of this part of the work-factor is:

$$\mathsf{W}_{3a} \approx \frac{k(n-k)}{n}\left((n-k-1)+(\frac{k(n-k)}{n}+k)\frac{(n-k-1)}{2}\right)$$

$$\approx \frac{k^2(n-k)^3}{2n^2} + \frac{k^2(n-k)^2}{2n}. \tag{6.16}$$

When combining (6.15) and (6.16), it results that

$$\mathsf{W}_2 + \mathsf{W}_{3a} \approx \frac{k^2(n-k)^3}{2n^2}(\gamma^{-1}+1) + \frac{k^2(n-k)^2}{2n}. \tag{6.17}$$

From the above estimates, it follows that $\mathsf{W}_2 + \mathsf{W}_{3a} \approx l(\mathsf{W}_{3b} + \mathsf{W}_{4a})$ for

$$l \approx n\Big(\mathrm{R}(1-\mathrm{R})(\gamma^{-1}+1) + \mathrm{R}\Big) \approx n.$$

So it stands to reason that the phrase *for sufficiently large l* means that l is a small multiple of n.

As with Algorithm 6.2.1, an additional one-error search in each round will reduce the expected number of rounds needed to find the solution. Since a suitable $(n-k)$ permutation is used in Step 2, it is also necessary for each round to verify if the selected coordinates are error-free. If a one-error search is added in Step 4, then when the weight of each of the k columns is at least t the algorithm returns to Step 2. Let $\underline{a}_j^{(u)T}$ denote the j-th column in the matrix $A^{(u)}$, and let $\underline{u}_j$ denote the j-th k-bit unit vector. Then in Algorithm 6.2.1, add the weight verification in Step 4 after the following:

- If for a j $(1 \le j \le k)$ and $\alpha \in \mathrm{GF}(q) \setminus \{0\}$ it holds that

  $$\mathsf{d}_{\mathrm{H}}(\underline{s}_i^{(u)}, \alpha\underline{a}_j^{(u)}) \le \lfloor (d-1)/2 \rfloor - 1,$$

  for some i $(1 \le i \le l)$, then the error vector is $\underline{z} = (\alpha\underline{u}_j \mid \underline{s}_i^{(u)})P^T$. Proceed to Step 5, otherwise, let $u := u+1$ and return to Step 2.

The work-factor for this step involves $2kt$ bit operations for one row of the matrix $B^{(u)}$. The algorithm terminates if at least one error vector has t or $t-1$ of its errors in the $n-k$ selected coordinates. Therefore, the expected number of rounds before the algorithm terminates becomes

$$\mathsf{N}_{\mathrm{BSD}}^{(1)}(l) \approx \frac{\binom{n}{t}}{\binom{n-k}{t} + \binom{k}{1}\binom{n-k}{t-1}} \times \frac{1}{l}.$$

An estimate of the work-factor $\mathsf{W}^{(1)}_{\mathrm{BSD}}$ is

$$\mathsf{W}^{(1)}_{\mathrm{BSD}} \approx \left(2(t+1) + 2kt + \frac{k}{2n}(n-k)(n-k-1)\right) \mathsf{N}^{(1)}_{\mathrm{BSD}}(1).$$

n	k	t	$\log_2 \mathsf{N}_{\mathrm{BSD}}$	$\log_2 \mathsf{W}_{\mathrm{BSD}}$	$\log_2 \mathsf{N}^{(1)}_{\mathrm{BSD}}$	$\log_2 \mathsf{W}^{(1)}_{\mathrm{BSD}}$
1024	654	37	56.1	71.5	49.9	66.4
1024	644	38	56.1	71.6	49.9	66.5
1024	634	39	56.1	71.6	49.9	66.5
1024	624	40	56.0	71.6	49.9	66.5
1024	614	41	56.0	71.6	49.8	66.5
1024	524	50	55.8	71.5	49.7	66.4

Table 6.3: The average number of bit operations for Algorithm 6.3.1 (BSD) with and without a one-error search.

n	k	t	$\log_2 \mathsf{N}_{\mathrm{BSD}}$	$\log_2 \mathsf{W}_{\mathrm{BSD}}$	$\log_2 \mathsf{N}^{(1)}_{\mathrm{BSD}}$	$\log_2 \mathsf{W}^{(1)}_{\mathrm{BSD}}$
512	256	30	31.3	45.3	26.2	41.1
512	256	56	60.9	74.9	54.7	70.2
1000	500	30	30.6	46.6	25.6	42.1
1024	512	40	41.1	57.1	35.7	52.4
1024	512	110	119.5	135.5	112.3	129.8

Table 6.4: The average number of bit operations for Algorithm 6.3.1 (BSD) with and without a one-error search.

Example 6.3.3 *The estimates of the work-factor for different values of the code parameters, with and without an additional one-error search, are given in Tables 6.3 and 6.4. Table 6.3 illustrates the security of some values of the parameters when used in the McEliece of Niederreiter public-key cryptosystem. Again, the table indicates that the optimal value is not critical. The BSD algorithm becomes highly suitable for parallel implementation by simply distributing the columns of the $l \times (n-k)$ batch matrix B over a number of processing units. However, to drastically lower the work-factor an extremely large value of l must be used.*

The following theorem gives the complexity coefficient of the BSD algorithm.

Theorem 6.3.4 (BSD) *For virtually all linear codes C in $C(n,q,R)$, the complexity coefficient of the batch syndrome decoding algorithm for BHDD satisfies*

$$dcc_{\mathrm{BSD}}(q,R) = \widehat{H}_q\left(\frac{H_q^{-1}(1-R)}{2}\right) - (1-R)\widehat{H}_q\left(\frac{H_q^{-1}(1-R)}{2(1-R)}\right).$$

For a memoryless QSC with $p < q^{-1}$, the probability of erroneous decoding equals Pe_{BHDD}.

Proof: For the memory-factor, note that only the parity-check matrix H, the vector $\underline{y}$ and the permutation matrix P need to be stored, therefore $\mathsf{M}_{\mathrm{BSD}} = O(n^2)$. From (6.14) it follows that the work-factor is $\mathsf{W}_{\mathrm{BSD}} = O(n^2)\mathsf{N}_{\mathrm{BSD}}(1)$, so that

$$\mathsf{DC}_{\mathrm{BSD}}(n,q,\mathrm{R}) = O(n^4)\mathsf{N}_{\mathrm{BSD}}(1). \tag{6.18}$$

Substituting (4.6) and $d = n\mathrm{H}_q^{-1}(1-\mathrm{R}) + o(n)$ (Lemma 2.2.2) into (6.18) and (2.11) yields the expression for the complexity coefficient. When the algorithm terminates, the probability of erroneous decoding equals Pe_{BHDD}. □

6.4 Structured-Batch Syndrome Decoding

The main advantage of a structured batch in Algorithm 6.3.1 is that Steps 3 and 4 can be distributed over several simple dedicated circuits. Each circuit performs only elementary row operations and a single weight computation. The structure of this algorithm is given in Figure 6.1.

Instead of using a batch with l syndromes from different ciphertexts, the batch B is constructed for the same ciphertext $\underline{y}$. Also, the permutation selection is restricted to a suitable one-swap permutation. The structured matrix B is chosen, such that, $\binom{t}{w}$ rows have $t-w$ errors $(t \geq w)$. Therefore, this method is comparable to Algorithm 6.2.1 with a w-error search.

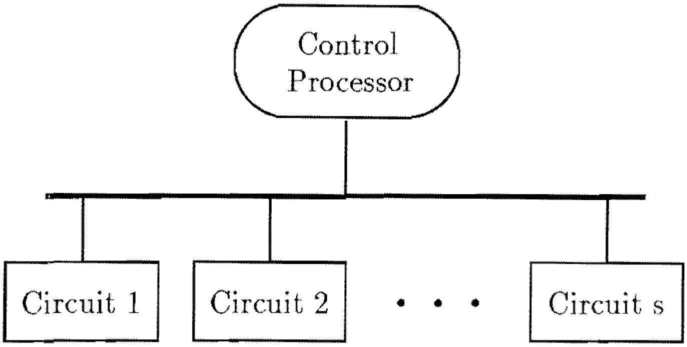


Figure 6.1: Structure of Algorithm 6.4.1 (SBSD).

Let s be the number of circuits and, without loss of generality, suppose that s divides $\binom{n}{w}$. Let $m = \binom{n}{w}/s$, and let B_r $(1 \leq r \leq s)$ be the r-th submatrix of B consisting of the rows $(r-1)m+1$ to rm.

Circuit The r-th circuit consists of the following four steps:

Step C1 *Initialization*

- Receive the submatrix B_r.

Step C2 *Verification*

- Let $\underline{s}_j$ denote the j-th row of B_r. If $\mathrm{w_H}(\underline{s}_j) \leq \lfloor (d-1)/2 \rfloor$ holds true for the j-th row of the matrix B_r, then proceed to Step 4.

Step C3 *Updating*

- Wait until i and vector $\underline{a}$ are received from the control processor in Algorithm 6.4.1.
- Perform elementary row operations on $(\underline{a}^T \mid B_r^T)$ to obtain $(\underline{u}_i^T \mid A_r^T)$. Define $B_r := A_r$, and return to Step 2.

Step C4 *Solution*

- Send the solution $\underline{b}_j$ (i.e., the j-th row of B_r) to the control processor.

Let $\underline{y} = \underline{c} + \underline{z}$ with $\mathrm{w_H}(\underline{z}) = t \leq \lfloor (d-1)/2 \rfloor$ be the received vector. Then define the ordered set $\mathcal{Y}_w$ with $w \leq t$ as

$$\mathcal{Y}_w := \{\underline{y} + \underline{w} \mid \mathrm{w_H}(\underline{w}) = w\}.$$

The cardinality of $\mathcal{Y}_w$ is $\binom{n}{w}(q-1)^w$ and this set contains $\binom{t}{w}$ error vectors of weight $t - w$. Let $\underline{y}_i$ be the i-th vector in $\mathcal{Y}_w$, and let Y be the matrix representation (i.e., the i-th row of Y corresponds to the i-th element in $\mathcal{Y}_w$).

Algorithm 6.4.1 (Structured-Batch Syndrome Decoding) *Let H be a parity-check matrix of a q-ary $[n, k, d]$-code, let w be a positive integer, and let $\underline{y}$ be a received vector. The algorithm consists of the following five steps:*

Step 1 *Initialization*

- Randomly select a permutation matrix $P^{(0)}$, such that, the last $n-k$ columns of the matrix $HP^{(0)}$ are linearly independent.
- Perform elementary row operations on $HP^{(0)}$ in order to obtain a matrix in systematic form $H^{(0)} := (A^{(0)} \mid I_{n-k})$.
- Compute the syndromes of $\underline{y} + \underline{w}$ with $\mathrm{w_H}(\underline{w}) = w$ defined by $\underline{b}_j = \underline{y}H^T + \underline{w}_j H^T$ for $j = 1, 2, \ldots, \binom{n}{w}$. Define the syndrome matrix $B^{(0)} := (\underline{b}_1^T, \underline{b}_2^T, \ldots, \underline{b}_{\binom{n}{w}}^T)^T$, $P := P^{(0)}$, and let $u := 0$.
- Distribute $B^{(0)} = (B_1^{(0)} \mid B_2^{(0)} \mid \ldots \mid B_s^{(0)})$ over the s circuits.

Step 2 *Receiving*

- If a solution is received from one of the s circuits, then proceed to Step 5, otherwise, let $u := u + 1$.

Step 3 *Permutation Selection*

- Select a suitable permutation $P^{(u)}$ (that swaps one symbol).

Step 4 *Sending, Swapping and Updating*

- Suppose that the i-th column of the I_{n-k} part is swapped for the column $\underline{a}^{(u-1)^T}$ from the $A^{(u-1)}$ matrix. Then send i and $\underline{a}^{(u-1)}$ to the s circuits.

- Transform $H^{(u-1)}P^{(u)}$ into $H^{(u)} = (A^{(u)} \mid I_{n-k})$.
- Let $P := PP^{(u)}$, and return to Step 2.

Step 5 *Codeword Calculation*

- Suppose that from the r-th circuit a solution is received. Then compute the error vector as $\underline{z} = (\underline{0} \mid \underline{s}^{(u)}_{(r-1)m+j})P^T - \underline{w}_{(r-1)m+j}$, and the codeword follows from $\underline{c} = \underline{y} - \underline{z}$.

Algorithm 6.4.1 terminates if, and only if, set $\mathcal{B}$ contains at most w error-locations of the error vector $\underline{z}$. The set $\mathcal{Y}_w$ contains at least one vector with $t - w$ errors, such that no error-locations are in the set $\mathcal{B}$ if, and only if, $\mathcal{B}$ contains at most w error-locations of $\underline{y}$. Therefore, this method is equivalent to adding a search for w or less errors in each round, and proves the following lemma:

Lemma 6.4.2 *Let N_l $(1 \leq l \leq a)$ be defined as in Lemma 6.2.2. Then the expected number of rounds before Algorithm 6.4.1 terminates is*

$$\mathsf{N}^{(w)}_{SBSD} = \mathsf{N}^{(w)}_{OSSD}. \tag{6.19}$$

The work-factor of Algorithm 6.4.1 depends on the work-factor of the control processor $\mathsf{W}^{(w)}_{\mathrm{P}}$, and the work-factor of a circuit $\mathsf{W}^{(w)}_{\mathrm{C}}$. The overall work-factor of *one round* is determined by the longest process, therefore on the average, each round needs approximately $\max\{\mathsf{W}^{(w)}_{\mathrm{P}}, \mathsf{W}^{(w)}_{\mathrm{C}}\}$ bit operations. The time estimate needed for communication between the control processor and the circuits is neglected. After $\mathsf{N}^{(w)}_{\mathrm{SBSD}}$ rounds, the algorithm is expected to terminate with an average work-factor of

$$\mathsf{W}^{(w)}_{\mathrm{SBSD}} \approx \max\{\mathsf{W}^{(w)}_{\mathrm{P}}, \mathsf{W}^{(w)}_{\mathrm{C}}\}\mathsf{N}^{(w)}_{\mathrm{SBSD}} + \mathsf{W}_1 + \mathsf{W}_5. \tag{6.20}$$

The work-factor for Step 1 is dominated by the matrix multiplication $YH^{(0)^T}$ and equals $\mathsf{W}_1 \approx (n-k)w\binom{n}{w}$. This work-factor increases rapidly in w. Therefore, W_1 can only be neglected when its value is smaller than $\mathsf{W}^{(w)}_{\mathrm{SBSD}}$ (i.e., provided that the value of w is not too large). Step 2 is a formal step, therefore its work-factor is neglected. Since Step 5 is only executed once, its work-factor is also neglected. With approximately one-half probability, a suitable permutation is found in

Step 3. Since a suitable permutation is recognized in an efficient way (one bit-check), the work-factor W_3 is neglected. In Step 4, the elementary row operations require $(n-k-1)$ bit verifications to check whether or not a coordinate in the swapped column (from the sub-matrix $A^{(u-1)}$) is nonzero. With approximately one-half probability, a coordinate of this swapped column is one, so that it also requires a k-bit addition. Therefore, the work-factor W_4 involves approximately $(n-k-1)+k(n-k-1)/2$ bit operations. This provides the following estimate:

$$\mathsf{W}_{\mathsf{P}}^{(w)} \approx (n-k-1)(k+1)/2. \tag{6.21}$$

Since Steps C1 and C4 are executed only once, their work-factors are neglected. As a result, the average work-factor for a circuit is dominated by Steps C2 and C3. Assume that B_r is a random matrix. The circuit proceeds to Step C3 when the weight of all the m syndromes is at least $t+1-w$, therefore, Step C2 requires $2m(t+1-w)$ bit operations for weight verification. In Step C3, the elementary row operations require $(n-k-1)$ bit verifications to check whether a coordinate in the vector $\underline{a}$ is nonzero. With approximately one-half probability, a coordinate is one, so that it also requires a m-bit addition. The work-factor W_{C3} involves approximately $(n-k-1)+m(n-k-1)/2$ bit operations, and

$$\mathsf{W}_{\mathsf{C}}^{(w)} \approx (m+2)(n-k-1)/2 + 2m(t+1-w). \tag{6.22}$$

Substitution of the work-factors (6.21) and (6.22) in (6.20) provides the following estimate of the overall work-factor:

$$\begin{aligned}\mathsf{W}_{\mathsf{SBSD}}^{(w)} \;\approx\; & \max\{\frac{k+1}{2}(n-k-1)\;,\;\frac{m+2}{2}(n-k-1)+ \\ & +2m(t+1-w)\}\mathsf{N}_{\mathsf{SBSD}}^{(w)} + (n-k)w\binom{n}{w}.\end{aligned}$$

Example 6.4.3 *In Table 6.5, the results for the McEliece system with parameter values $n = 1024$, $k = 634$ and $t = 39$ are given. The number of columns m of the matrix B_r that is assigned to each circuit is 470.*

If the size m of the circuits is chosen, such that the values of $\mathsf{W}_{\mathsf{P}}^{(w)}$ and $\mathsf{W}_{\mathsf{C}}^{(w)}$ are almost equal, then

$$m \approx \frac{k}{(1+\frac{4t}{n-k})},$$

w	$\log_2 s$	$\log_2 \mathsf{N}^{(w)}_{\mathsf{SBSD}}$	$\log_2 \mathsf{W}^{(w)}_{\mathsf{SBSD}}$
1	1.12	53.4	70.3
2	10.1	48.3	65.3
3	18.5	43.9	60.9
4	26.5	40.0	57.0
5	34.2	36.5	53.4

Table 6.5: The average number of bit operations for Algorithm 6.4.1 (SBSD) with m=470 for a random [1024,634]-code with 39 errors.

so that the size m of a circuit grows almost linearly in k.

Example 6.4.4 *The estimates of the work-factor for different values of the code parameters, with and without an additional one-error search, are given in Tables 6.6 and 6.7. Table 6.6 illustrates the security of some values of the parameters when used in the McEliece of Niederreiter public-key cryptosystem. Again, the table indicates that the optimal value is not critical.*

Since Algorithm 6.4.1 corresponds to Algorithm 6.2.1 with a w-error search, the complexity coefficient of the SBSD algorithm follows directly from Theorem 6.2.6 and is given in the following theorem.

Theorem 6.4.5 (SBSD) *For virtually all linear codes $\mathcal{C}$ in $\mathcal{C}(n,q,R)$, the complexity coefficient of the structured batch decoding algorithm for BHDD satisfies*

$$dcc_{\mathsf{SBSD}}(q,R) = \widehat{H}_q\left(\frac{H_q^{-1}(1-R)}{2}\right) - (1-R)\widehat{H}_q\left(\frac{H_q^{-1}(1-R)}{2(1-R)}\right).$$

For a memoryless QSC with $p < q^{-1}$, the probability of erroneous decoding equals Pe_{BHDD}.

n	k	t	$\log_2 \mathsf{N}^{(4)}_{\mathsf{SBSD}}$	$\log_2 \mathsf{W}^{(4)}_{\mathsf{SBSD}}$
1024	654	37	39.8	56.7
1024	644	38	40.0	56.9
1024	634	39	40.0	57.0
1024	624	40	40.1	57.0
1024	614	41	40.1	57.0
1024	604	42	40.0	57.0
1024	594	43	39.9	56.9
1024	524	50	38.6	55.6

Table 6.6: The average number of rounds and bit operations for Algorithm 6.4.1 (SBSD) with $\log_2 s \approx 34$ and $m \approx 470$.

n	k	t	w	m	$\log_2 s$	$\log_2 \mathsf{N}^{(w)}_{\mathsf{SBSD}}$	$\log_2 \mathsf{W}^{(w)}_{\mathsf{SBSD}}$
512	256	30	2	174	17.0	25.6	40.6
512	256	56	4	137	31.0	43.5	58.5
1000	500	30	2	403	18.7	26.0	43.0
1024	512	40	3	390	26.8	31.5	48.0
1024	512	110	9	275	70.0	77.0	94.0

Table 6.7: The average number of rounds and bit operations for Algorithm 6.4.1 (SBSD).

Chapter 7

Soft-Decision Decoding

Hard-decision-decoding algorithms recover the original codeword from the received vector by using the redundancy of the code only. For many applications, these decoding algorithms are too restrictive, because additional information is available which can provide a better decoding performance. In cryptology for example, a locally-randomized cryptosystem is used that can correct t errors, however, only $t < \lfloor (d-1)/2 \rfloor$ errors are added intensionally, so that $\lfloor (d-1)/2 \rfloor - t$ additional errors can be introduced by a noisy channel that are correctable. Or, when the error vector is chosen according to a strict rule that provides an additional information channel (concealed or subliminal channel) with its own channel characteristics. In both cases, the cryptanalyst might obtain an important advantage through information leakage that will be additional input for the decoding algorithm. As a result, the work-factor to obtain the error vector (or, equivalently, the plaintext) decreases significantly.

In [Cha72], Chase described an example of such a decoding technique called *soft-decision decoding* in which the additional information is based on channel measurements. Also, Wolf [Wol78] defined a (different) type of soft-decision decoding. This chapter's objective, however, is to examine the complexity of channel-measurement decoding for random codes and to prove that its asymptotical complexity is equal to that of the underlying BHDD algorithm. Moreover, the difference

This chapter is based on joint work with Raymond Doyen [Doy92, DT94a].

in complexity is demonstrated by a direct application of ISD to soft-decision decoding as compared with an ISD algorithm applied directly in Chase's algorithm. In [CGF91], Coffey, Goodman and Farrell showed that bounded, soft-decision decoding has the same complexity coefficient as CHDD. This chapter shows that for Gaussian and Rayleigh fading channels, and for virtually all linear codes, bounded soft-decision decoding has the same complexity coefficient as BHDD. The workfactors are not computed since they are only meaningful when viewed with an estimate of the corresponding probability of erroneous decoding.

7.1 Preliminaries

Chase proposes a soft-decision-decoding algorithm based on the existence of an algorithm for the BHDD. The BHDD algorithm corrects up to $\lfloor (d-1)/2 \rfloor$ errors, where d is the minimum Hamming distance of the code.

Assume that the output of a channel consists of an output vector $\underline{y} \in \mathrm{GF}(2)^n$ together with a vector of n positive real numbers $\underline{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$. These numbers are called *channel measurements.* Each value α_i is an estimate of the reliability of the corresponding bit y_i $(1 \leq i \leq n)$. If $\alpha_i > \alpha_j$, then y_i is more likely to be correct than y_j. Therefore, the channel-measurement vector provides a reliability measure of the received vector.

Using channel measurements, the definition of binary Hamming weight and distance can be extended as follows:

Definition 7.1.1 (Binary Analog Weight and Distance) *Let* $\underline{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ *be a vector of* n *positive real numbers. The binary analog weight of an* n*-bit vector* $\underline{x}$ *is*

$$w_{\underline{\alpha}}(\underline{x}) = \sum_{i=1}^{n} \alpha_i x_i.$$

The binary analog distance between two n*-bit vectors* $\underline{x}$ *and* $\underline{y}$ *is*

$$d_{\underline{\alpha}}(\underline{x}, \underline{y}) = w_{\underline{\alpha}}(\underline{x} + \underline{y}) = \sum_{i=1}^{n} \alpha_i (x_i \oplus y_i),$$

where $\oplus$ represents the addition modulo two.

Similar to hard-decision decoding, there is a distinction between complete soft-decision decoding and bounded soft-decision decoding. Therefore, it is important to first define complete soft-decision decoding as it relates to channel measurements.

Definition 7.1.2 (Complete Soft-Decision Decoding) *Let C be a binary linear code. Let the vector $\underline{\alpha}$ provide a reliability measurement of the received vector $\underline{y}$. A complete soft-decision decoding algorithm decodes the received vector $\underline{y}$ with the channel-measurement vector $\underline{\alpha}$ into a not-necessarily-unique codeword $\underline{c} \in C$ for which*

$$d_{\underline{\alpha}}(\underline{y}, \underline{c}) = \min\{d_{\underline{\alpha}}(\underline{y}, \underline{x}) \mid \underline{x} \in C\}.$$

The probability of erroneous decoding of an arbitrary received vector $\underline{y}$ is denoted by Pe_{CSDD}.

The covering radius and the minimum distance for virtually all linear codes are asymptotically equal (see Lemma 2.2.2). Therefore, using the redundancy of the code along with channel measurements extends the error-correcting capability from $\lfloor (d-1)/2 \rfloor$ errors (as in BHDD) to $d-1$ errors [Cha72]. This result is the basis for the following definition:

Definition 7.1.3 (Bounded Soft-Decision Decoding) *Let C be a binary linear code with minimum distance d. Let the vector $\underline{\alpha}$ provide a reliability measurement of the received vector $\underline{y}$. A bounded soft-decision algorithm decodes the received vector $\underline{y}$ with* channel-measurement vector $\underline{\alpha}$ *into a not-necessarily-unique codeword $\underline{c} \in C$ for which*

$$d_{\underline{\alpha}}(\underline{y}, \underline{c}) = \min\{d_{\underline{\alpha}}(\underline{y}, \underline{x}) \mid \underline{x} \in C \text{ and } d_H(\underline{y}, \underline{x}) \leq d-1\},$$

if such a codeword exists. The probability of erroneous decoding is denoted by Pe_{BSDD}.

In general, more than one codeword exists at Hamming distance less than or equal to $d-1$ from the received vector. However, under these circumstances the codeword with the highest reliability based on channel measurements is chosen. In other words, select the codeword with the minimum analog distance and at Hamming distance less than or equal to $d-1$ from the received vector. If more than one codeword meets this condition, then select the codeword with the lowest Hamming distance.

The output $\underline{c}$ of a CSDD algorithm is only a solution to the BSDD problem if it is at Hamming distance less than or equal to $d-1$ from the received vector $\underline{y}$. Apart from this, knowledge of the solution of the CSDD problem provides no additional information to solve the BSDD problem. As a consequence, sometimes the relation $Pe_{\text{CSDD}} \leq Pe_{\text{BSDD}}$ does not hold.

As CHDD is an $\mathcal{NP}$-complete problem [BMT78] and is a subclass of CSDD, the CSDD is also an $\mathcal{NP}$-complete problem (e.g., an efficient algorithm for CSDD solves CHDD). However, like BHDD, this does not imply that the BSDD problem is $\mathcal{NP}$-complete.

7.2 The Chase Algorithm

In [Cha72], Chase describes an algorithm for binary BSDD. The algorithm uses a binary BHDD algorithm to obtain a relatively small set of possible error vectors. The following algorithm uses a set of test patterns that will be discussed in Section 7.3.

Algorithm 7.2.1 (Chase) *Let $\mathcal{C}$ be a binary linear code with minimum distance d. Let $\mathcal{T}$ be a set of test patterns with Hamming weight no greater than $\lfloor d/2 \rfloor$. Suppose $\underline{r}$ is the received vector with the channel-measurement vector $\underline{\alpha}$. Let $w_{min} := \infty$, $\underline{z}_{min} := \underline{0}$ and $u := 0$. Chase's algorithm consists of the following five steps:*

Step 1 *Select a new (not previously used) vector $\underline{t}$ from $\mathcal{T}$. Let $\underline{y} := \underline{r} + \underline{t}$ and $u := u + 1$.*

Step 2 *Decode the vector $\underline{y}$ with a BHDD algorithm for $\mathcal{C}$. Let $\underline{z}$ with $w_H(\underline{z}) \leq \lfloor (d-1)/2 \rfloor$ be the error vector obtained. (If the BHDD algorithm does not find a codeword within radius $\lfloor (d-1)/2 \rfloor$ of $\underline{y}$, then let $\underline{z} := \underline{0}$.)*

Step 3 *Let* $w := \mathrm{w}_{\underline{\alpha}}(\underline{z}+\underline{t})$. *If* $w < w_{min}$, *then* $\underline{z}_{min} := \underline{z}+\underline{t}$ *and* $w_{min} := w$.

Step 4 *If* $u < |\mathcal{T}|$, *then return to Step 1.*

Step 5 *Exit with* $\underline{c}$ $(= \underline{r} + \underline{z}_{min})$.

In Step 2, the BHDD algorithm only outputs a unique codeword $\underline{c}$ when $\mathrm{d_H}(\underline{y},\underline{c}) \leq \lfloor (d-1)/2 \rfloor$. When no codeword is found, the output is assumed to be $\underline{y}$ (i.e., $\underline{z} = \underline{0}$) although this vector is certainly in error (when $\underline{y}$ is not a codeword).

For every test pattern $\underline{t}$ in Algorithm 7.2.1, it holds that $\mathrm{w_H}(\underline{t}) \leq \lfloor d/2 \rfloor$. Therefore, $\mathrm{w_H}(\underline{z}+\underline{t}) \leq \mathrm{w_H}(\underline{z}) + \mathrm{w_H}(\underline{t}) \leq \lfloor (d-1)/2 \rfloor + \lfloor d/2 \rfloor = d-1$.

Lemma 7.2.2 *Let* $\mathcal{T}$ *be a set of test patterns. If a binary BHDD algorithm has complexity* $DC_{\mathrm{BHDD}}(n,2,R)$, *then the complexity of Algorithm 7.2.1 is given by:*

$$DC_{\mathrm{Chase},\mathcal{T}}(n,2,R) = |\mathcal{T}| \times DC_{\mathrm{BHDD}}(n,2,R) \times 2^{o(n)} \quad (n \to \infty), \quad (7.1)$$

and the complexity coefficient is

$$dcc_{\mathrm{Chase},\mathcal{T}}(2,R) = dcc_{\mathrm{BHDD}}(2,R) + \lim_{n\to\infty} \frac{1}{n} \log_2 |\mathcal{T}|. \qquad (7.2)$$

Proof: Define a *round* as the sequence of Steps 1 through 4. Since the complexity of an n-bit vector addition is $O(n)$, each round has complexity $\mathrm{DC_{BHDD}}(n,2,\mathrm{R}) + O(n)$. After $|\mathcal{T}|$ rounds the algorithm terminates. Therefore, the complexity of the algorithm becomes

$$\begin{aligned} &\mathrm{DC}_{\mathrm{Chase},\mathcal{T}}(n,2,\mathrm{R}) \\ &= |\mathcal{T}| \times [\mathrm{DC_{BHDD}}(n,2,\mathrm{R}) + O(n)] + O(n) \\ &= |\mathcal{T}| \times [\mathrm{DC_{BHDD}}(n,2,\mathrm{R}) + 2^{o(n)}], \quad (n \to \infty). \end{aligned}$$

From (2.3.2), the following expression for the complexity coefficient is obtained:

$$\mathrm{dcc}_{\mathrm{Chase},\mathcal{T}}(2,\mathrm{R}) = \mathrm{dcc_{BHDD}}(2,\mathrm{R}) + \lim_{n\to\infty} \frac{1}{n} \log_2 |\mathcal{T}|,$$

which proves the lemma. □

From (7.2) it follows that, if the size of the set of test patterns is polynomially bounded in n, then $\mathrm{dcc}_{\mathrm{Chase},\mathcal{T}}(2,\mathrm{R}) = \mathrm{dcc_{BHDD}}(2,\mathrm{R})$.

7.3 Test Patterns

The objective of Algorithm 7.2.1 is to select a set of test patterns such that all the error vectors within the sphere of radius $d-1$ can be reached. Chase obtained the following three sets of test patterns via heuristics and computer simulations. In practical implementations, these test patterns yield relatively low error probabilities.

Test Set 1 $\mathcal{T}_1 := \{\underline{t} \in \mathrm{GF}(2)^n \mid \mathrm{w_H}(\underline{t}) = \lfloor d/2 \rfloor\} \cup \{\underline{0}\}$.
Every pattern of Hamming weight less than or equal to $d-1$ is the sum of an element of $\mathcal{T}_1$ and a vector of Hamming weight less than or equal to $\lfloor (d-1)/2 \rfloor$. Thus, for an arbitrary error vector of weight less than or equal to $d-1$, there exists an test pattern $\underline{t}$ in $\mathcal{T}_1$, such that, after subtraction of the error vector by $\underline{t}$, the remainder is detected by a BHDD algorithm. As a result, Algorithm 7.2.1 with $\mathcal{T}_1$ corrects any error vector of Hamming weight less than or equal to $d-1$. If the received vector $\underline{r}$ is a codeword, then only $\underline{t} = \underline{0}$ yields the estimate $\underline{c} = \underline{r}$. Therefore, $\underline{t} = \underline{0}$ is in $\mathcal{T}_1$ and its cardinality is given by

$$|\mathcal{T}_1| = \binom{n}{\lfloor d/2 \rfloor} + 1. \tag{7.3}$$

This cardinality increases rapidly in d. Therefore, $\mathcal{T}_1$ is only useful for codes with a small minimum distance.

Without loss of generality, assume that the channel-measurement vector satisfies $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_n$. The next test set only considers $\alpha_1, \alpha_2, \ldots, \alpha_{\lfloor d/2 \rfloor}$ (i.e., the $\lfloor d/2 \rfloor$ lowest channel measurements).

Test Set 2 $\mathcal{T}_2 := \{\underline{t} \in \mathrm{GF}(2)^n \mid t_j = 0 \text{ for all } j > \lfloor d/2 \rfloor\}$.
Every vector $\underline{t}$ has any combination of ones located in the $\lfloor d/2 \rfloor$ positions with the lowest channel measurements. Therefore, the number of possible test patterns equals

$$|\mathcal{T}_2| = 2^{\lfloor d/2 \rfloor}, \tag{7.4}$$

and its size is considerably smaller than (7.3).

To further reduce the cardinality of the test set, a subset of $\mathcal{T}_2$ (including the zero test pattern) is defined. For every i ($1 \leq i \leq \lfloor d/2 \rfloor$), consider the set of the i lowest channel measurements (i.e., the first i positions).

Test Set 3 $\mathcal{T}_3 := \{(0\ldots0), (10\ldots0), (110\ldots0), \ldots, (\overbrace{1\ldots1}^{\lfloor d/2\rfloor\times}0\ldots0)\}$. Each test pattern has Hamming weight no greater than $\lfloor d/2 \rfloor$, therefore the cardinality of $\mathcal{T}_3$ equals

$$|\mathcal{T}_3| = \lfloor d/2 \rfloor + 1. \tag{7.5}$$

This cardinality increases linearly as a function of the minimum distance d of C and is far less than (7.3) and (7.4).

For almost all binary linear codes of length n and rate R, the *minimum distance* d satisfies $n\mathsf{H}_2^{-1}(1-\mathrm{R}) + o(n)$ for $n \to \infty$ (i.e., $q = 2$ in Equation (2.9)). Substitution in (7.3), (7.4) and (7.5) proves the following lemma.

Lemma 7.3.1 *For virtually all binary linear codes C with length n and rate R, the cardinality of the three sets of test patterns satisfies*

$$\begin{aligned}
\log_2 |\mathcal{T}_1| &= nH_2\left(\frac{H_2^{-1}(1-R)}{2}\right) + o(n), \quad (n \to \infty),\\
\log_2 |\mathcal{T}_2| &= \frac{nH_2^{-1}(1-R)}{2} + o(n), \quad (n \to \infty),\\
\log_2 |\mathcal{T}_3| &= \log_2\left(\frac{nH_2^{-1}(1-R)}{2} + o(n)\right), \quad (n \to \infty).
\end{aligned}$$

The following theorem is alluded to in [Cha72].

Theorem 7.3.2 *For a Gaussian and a Rayleigh fading channel and for virtually all binary linear codes C with length n and rate R, the probability of erroneous decoding using Algorithm 7.2.1 is asymptotically equal for each of the three test pattern sets.*

7.4 Complexity Coefficients

Theorem 7.3.2 states that for two specific channels a binary BSDD algorithm can extend the error-correcting power of code C (with minimum distance d) to a maximum of $d-1$ errors. Coffey and Goodman [CG90b] used this result in their proof of the following theorem:

Theorem 7.4.1 (Coffey and Goodman) *For virtually all binary linear codes C with length n and rate R, the complexity coefficient of the generalized Information Set Decoding algorithm for BSDD satisfies*

$$dcc_{GISD}(2,R) = (1-R)\left[1 - H_2\left(\frac{H_2^{-1}(1-R)}{1-R}\right)\right].$$

Based on the principle of Information Set Decoding (Section 4.3), a BHDD algorithm has for virtually all linear codes of length n and rate R, a complexity coefficient that satisfies

$$\mathsf{dcc}_{\mathrm{ISD}}(2,\mathrm{R}) = \mathsf{H}_2\left(\frac{\mathsf{H}_2^{-1}(1-\mathrm{R})}{2}\right) - (1-\mathrm{R})\mathsf{H}_2\left(\frac{\mathsf{H}_2^{-1}(1-\mathrm{R})}{2(1-\mathrm{R})}\right), \quad (7.6)$$

and the probability of erroneous decoding is Pe_{BHDD}. A direct application of this class of BHDD algorithms in Algorithm 7.2.1 yields the following theorem and the results are illustrated in Figure 7.1.

Theorem 7.4.2 *Let n, q and R be given, and let the ISD algorithm be the BHDD algorithm in Algorithm 7.2.1. Then, for virtually all binary linear codes C with length n and rate R, the complexity coefficient for each test set satisfies*

$$\begin{aligned} dcc_{Chase,T_1}(2,R) &= dcc_{ISD}(2,R) + H_2\left(\frac{H_2^{-1}(1-R)}{2}\right), \\ dcc_{Chase,T_2}(2,R) &= dcc_{ISD}(2,R) + \frac{H_2^{-1}(1-R)}{2}, \\ dcc_{Chase,T_3}(2,R) &= dcc_{ISD}(2,R). \end{aligned}$$

For Gaussian and Rayleigh fading channels, the corresponding probability of erroneous decoding in each of the three cases is Pe_{BSDD}.

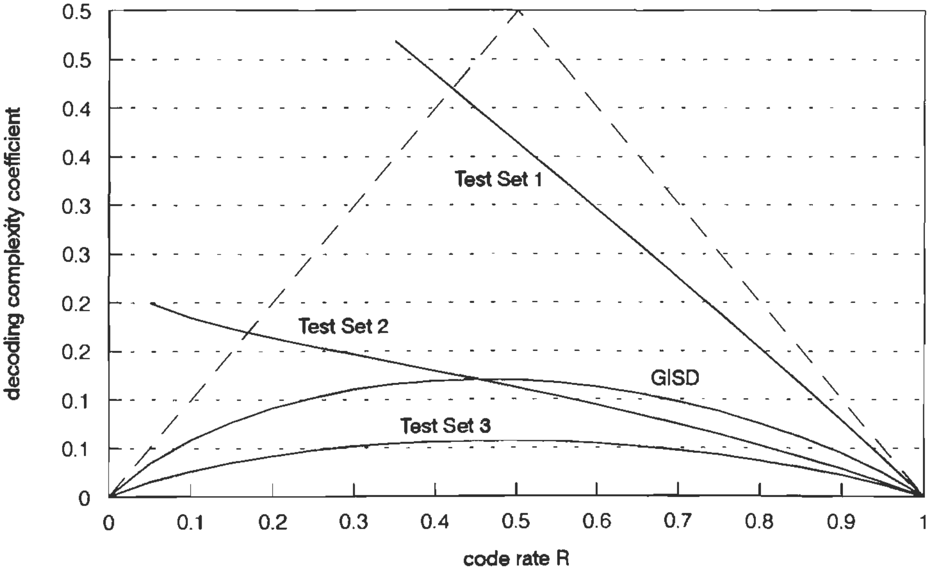


Figure 7.1: The binary complexity coefficients for BSDD.

Proof: According to Lemma 7.2.2, the complexity coefficient of Algorithm 7.2.1 using the ISD algorithm is

$$\mathrm{dcc}_{\mathrm{Chase},\mathcal{T}_i}(2,\mathrm{R}) = \mathrm{dcc}_{\mathrm{ISD}}(2,\mathrm{R}) + \lim_{n\to\infty} \frac{1}{n}\log_2|\mathcal{T}_i|, \quad (1 \le i \le 3). \quad (7.7)$$

Substituting Lemma 7.3.1 and (7.6) into (7.7) yields the complexity coefficient for each of the three test pattern sets.

For virtually all binary linear codes $\mathcal{C}$, with length n and rate R, the probability of erroneous decoding using test set $\mathcal{T}_1$ is asymptotically equal to Pe_{BSDD}. Therefore, from Theorem 7.3.2, it follows that for virtually all such codes the probability of erroneous decoding for the three test pattern sets tends to Pe_{BSDD} for $n \to \infty$. □

Chapter 8

A Class of Secret-Key Cryptosystems

This chapter discusses three locally-randomized, secret-key cryptosystems. The first is a secret-key variant of the McEliece public-key cryptosystem. This cryptosystem uses a set of errors that is restricted to vectors of weight at most $\lfloor (d-1)/2 \rfloor$ where d is the minimum distance of the code. Section 8.1 shows that this type of cryptosystem is vulnerable to Majority Voting analysis in which the secret encryption matrix is obtained in an efficient way. This Majority Voting analysis is not successful when the average Hamming weight of the n-bit error vectors in $\mathcal{Z}$ equals $n/2$. Therefore, the Rao-Nam secret-key cryptosystem [RN87] uses a set of predefined error vectors with distinct syndromes and average Hamming weight $n/2$. As shown in Section 8.2, several other algorithms are successful in obtaining the secret encryption matrix in the Rao-Nam scheme. Section 8.3 discusses the Li-Wang secret-key cryptosystem [LW91] which uses the *concealed channel* in the RN scheme for authentication purposes. Instead of randomly choosing an error vector from the set $\mathcal{Z}$, this scheme uses an error-selection function to assign for each plaintext only one fixed error vector from $\mathcal{Z}$. It is shown that this error-selection procedure makes the Li-Wang scheme less secure than the RN scheme.

This chapter is based on joint work with Joost Meijers [Mei90, MT91b, MT91a] and René Struik [Str87, ST88].

8.1 The Secret-Code Encryption Scheme

A secret-key variant of the McEliece scheme called *Secret-Code Encryptions* (SCE) was proposed by Jordan [Jor83] and by Rao [Rao84]. Their objective was to use simpler and smaller codes requiring less storage and enabling processing at higher speeds. The secret key of SCE consists of keeping both the private key and the public key of the McEliece scheme secret. This locally-randomized cryptosystem is defined as follows:

Definition 8.1.1 (The Secret-Code Encryption Scheme) *Let $\mathcal{C}$ be a binary linear $[n, k, d]$-code for which an efficient decoding algorithm exists. Define the set of error vectors as:*

$$\mathcal{Z} = \{\underline{z} \in GF(2)^n \mid w_H(\underline{z}) = \lfloor (d-1)/2 \rfloor\}.$$

Let G be a $k \times n$ generator matrix of $\mathcal{C}$. The encryption matrix is SGP, where S is a random binary $k \times k$ non-singular matrix, and P is a random $n \times n$ permutation matrix.

Encryption *A k-bit plaintext $\underline{x}$ is encrypted into an n-bit ciphertext $\underline{y}$ as:*

$$\underline{y} = \underline{x}SGP + \underline{z},$$

where $\underline{z}$ is randomly chosen from $\mathcal{Z}$.

Decryption *A ciphertext $\underline{y}$ is decrypted into a plaintext $\underline{x}$ by computing $\underline{y}P^T = \underline{x}SG - \underline{z}P^T$ (the vector $\underline{z}P^T$ also belongs to $\mathcal{Z}$). Next, the vector $\underline{x}S$ is obtained by applying the decoding algorithm of $\mathcal{C}$. The plaintext $\underline{x}$ is computed as $(\underline{x}S)S^{-1}$.*

Key *The secret key consists of: the matrices S, G and P, and the decoding algorithm of $\mathcal{C}$.*

The following security analysis is based on *Majority Voting* (MV) and shows that the secret matrix SGP can be obtained in an efficient way. By using a BHDD algorithm as described in Chapter 4, the value for the code parameters should be chosen, so that this decoding problem is infeasible to solve.

Let $\mathsf{M}(\mathcal{Z})$ be a matrix representation of a set $\mathcal{Z}$ of n-bit vectors, such that

$$\mathsf{M}(\mathcal{Z}) = (z_{ij}) \text{ with } 1 \le i \le |\mathcal{Z}| \text{ and } 1 \le j \le n,$$

where z_{ij} is the j-th bit of the i-th vector in $\mathcal{Z}$. In SCE, an error vector $\underline{z}$ is chosen according to a uniform distribution from the set

$$\mathcal{Z} = \{\underline{z} \in \mathrm{GF}(2)^n \mid \mathrm{w_H}(\underline{z}) = \lfloor (d-1)/2 \rfloor\},$$

so that

$$\mathsf{P}_{\mathcal{Z}}(\underline{z}) = |\mathcal{Z}|^{-1} = \binom{n}{\lfloor (d-1)/2 \rfloor}. \tag{8.1}$$

Let $\underline{\boldsymbol{z}} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n)$ denote a vector of random variables. From (8.1) it follows that the random variables $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n$ are not independent (i.e., $\mathsf{P}_{\mathcal{Z}}(\underline{z}) \neq \prod_{i=1}^{n} \mathsf{P}_{\mathcal{Z}}(z_i)$). The conditional probability distribution $\mathsf{P}_{\mathcal{Y}|\mathcal{X}}(\underline{y}|\underline{x})$ of the random variable $\underline{\boldsymbol{y}}$ given the event that $\underline{\boldsymbol{x}} = \underline{x}$ is

$$\mathsf{P}_{\mathcal{Y}|\mathcal{X}}(\underline{y}|\underline{x}) = \mathsf{P}_{\mathcal{Z}}(\underline{y} - \underline{x}E) = \mathsf{P}_{\mathcal{Z}}(\underline{z}).$$

Define $\lambda = \lfloor (d-1)/2 \rfloor / n$, then each column of $\mathsf{M}(\mathcal{Z})$ has exactly $\lambda |\mathcal{Z}|$ ones. Let $\mathcal{Z}_l$ be a random subset of $\mathcal{Z}$ with cardinality $|\mathcal{Z}_l| = l$. The rows of $\mathsf{M}(\mathcal{Z}_l)$ coincide with l independent drawings from the same probability distribution $\mathsf{P}_{\mathcal{Z}}(\underline{z})$. Then *The Law of Large Numbers* [Fel57] states that for sufficiently large l, the probability that a symbol occurs in a column $\mathsf{M}(\mathcal{Z}_l)$ converges (in probability) to the symbol's probability of occurrence. In other words, for sufficiently large l, every column of $\mathsf{M}(\mathcal{Z}_l)$ has approximately $\lambda |\mathcal{Z}_l|$ ones, and the number of ones satisfy the binomial distribution with parameters l and λ. Therefore, among l ciphertexts the (error) probability $\mathsf{Pe}(\lambda, l)$ that the number of 1s on the j-th coordinate of the error vector does not exceed the number of 0s is given by:

$$\mathsf{Pe}(\lambda, l) = \sum_{j=\lceil \frac{l}{2} \rceil}^{l} \binom{l}{j} \lambda^j (1-\lambda)^{l-j} \le \left[\sqrt{4\lambda(1-\lambda)}\right]^l. \tag{8.2}$$

As a result, a majority vote on the j-th coordinate of the received vectors yields an estimate for the j-th coordinate in the vector $\underline{x}SGP$.

The probability $\mathsf{Pc}(\lambda, l)$ of a correct estimate is $1 - \mathsf{Pe}(\lambda, l)$, and can be upper-bounded as follows:

$$\mathsf{Pc}(\lambda, l) \geq 1 - \left[\sqrt{4\lambda(1-\lambda)}\right]^l . \qquad (8.3)$$

When $\lambda \neq 1/2$, then $\mathsf{Pc}(\lambda, l)$ tends to 1 for sufficiently large l. Therefore, the bit on the j-th coordinate of the n-bit vector $\underline{x}SGP$ can be retrieved by Majority Voting on the bits in the j-th column of the matrix $\mathsf{M}(\mathcal{Y}_l)$.

Algorithm 8.1.2 (Majority Voting) *Consider the cryptosystem in Definition 8.1.1. The following two steps recover the secret matrix* $E = SGP$*:*

Step 1 *Choose an arbitrary k-bit vector $\underline{x}$, and obtain l distinct encryptions of $\underline{x}$ (i.e., $\underline{y}_i = \underline{x}E + \underline{z}_i$ with $1 \leq i \leq l$). Let $\mathcal{Y}_l$ denote the set $\{\underline{y}_1, \underline{y}_2, \cdots, \underline{y}_l\}$ of ciphertexts, and let $\mathcal{Z}_l$ denote the set $\{\underline{z}_1, \underline{z}_2, \cdots, \underline{z}_l\}$ distinct error vectors, such that*

$$\mathsf{M}(\mathcal{Y}_l) = (\underline{x}E) + \mathsf{M}(\mathcal{Z}_l).$$

Then, majority voting on each column of $\mathsf{M}(\mathcal{Y}_l)$ yields an estimate $\underline{x}E$ of $\underline{x}SGP$ (i.e., when the 1s dominate in a column, set the corresponding bit to 1, otherwise to 0).

Step 2 *Repeat Step 1 for k linearly independent vectors $\underline{x}$. Let the rows of the matrix X consist of these k vectors, and let the rows of XE consist of the corresponding k estimates. Then, an estimate E for the matrix SGP follows from*

$$E = X^{-1}(XE).$$

With probability $\mathsf{Pc}(\lambda, l)^{kn}$, the matrix E obtained in this algorithm is a correct estimate of the secret matrix SGP. Algorithm 8.1.2 requires k times l majority votes over n coordinates, so that, the work-factor W requires an average number of $O(knl)$ bit operations.

This MV analysis can also be formulated as a decoding problem for a repetition code. Suppose that l ciphertexts of length n with t errors are received, then each coordinate is *locally randomized* with probability $\approx t/n$. In essence, each column of $\mathsf{M}(\mathcal{Z}_l)$ corresponds to a codeword of a repetition code of length l which is transmitted over a BSC with crossover probability $\lambda = t/n$. Suppose each bit is transmitted l times. When MV decoding is used, then the resulting probability of erroneous decoding for each codeword equals $\mathsf{Pe}(\lambda, l)$ as given in (8.2). Since a repetition code improves the channel's reliability, the channel can be made as reliable as desired by increasing the code length. In a cryptographic sense, the MV analysis can be as powerful as desired. The objective of a MV analysis for the McEliece scheme differs, since the encryption matrix E is publicly known. Recall that the cryptanalyst can obtain the plaintext by repeating the majority voting several times. The MV analysis works for codes with high information rates and is even more successful when error vectors of lower Hamming weights are allowed in $\mathcal{Z}$. Therefore, with the same system parameters, the security of SCE is essentially equal to the security of the McEliece scheme. The original objective of SCE was to reduce the amount of computational overhead by using a code of smaller length and higher information rate. In the McEliece scheme, such a code renders the system insecure. Therefore, it can also be concluded that the same code when used in SCE results in an insecure system.

8.2 The Rao-Nam Scheme

A MV analysis is not successful when the average Hamming weight of the error vectors equals $n/2$. In this case, a bit on an arbitrary coordinate of the codeword is guessed correctly with probability $1/2$. In other words, guessing the symbol may as well be without any strategy. For the decoding approach this means that each BSC has a crossover probability of $\lambda = 1/2$ (i.e., the channel capacity is zero and no reliable transmission or decoding strategy is possible). From a cryptographic point of view this is a desirable situation. Therefore, Rao and Nam [RN87] proposed using a set $\mathcal{Z}$ of predefined error vectors with Hamming weight $\approx n/2$ (*weight constraint*) and all in different cosets of $\mathcal{C}$. They also proposed to use simple codes with a minimum distance of 3 or 4, and in their paper [RN87] is was concluded that their scheme is computationally secure even for $k \approx 50$. The RN scheme can be summarized as follows:

Definition 8.2.1 (The Rao-Nam Scheme) *Let $\mathcal{C}$ be a binary linear $[n,k]$-code $\mathcal{C}$. Let G be a $k \times n$ generator matrix of the code $\mathcal{C}$, and let H be an $(n-k) \times n$ parity-check matrix. Define the set of error vectors as:*

$$\mathcal{Z} = \{\underline{z} \in GF(2)^n \mid w_H(\underline{z}) \approx n/2\},$$

such that, each vector $\underline{z} \in \mathcal{Z}$ has a unique syndrome (i.e., each error vector lie in a different cosets of $\mathcal{C}$). Let $\mathcal{T}$ be the syndrome-error table: $\mathcal{T} = \{(\underline{z}H^T, \underline{z}) \mid \underline{z} \in \mathcal{Z}\}$. Let S be a binary $k \times k$ nonsingular matrix, and let P be an $n \times n$ permutation matrix.

Encryption *A k-bit plaintext $\underline{x}$ is encrypted into an n-bit ciphertext $\underline{y}$ as:*

$$\underline{y} = (\underline{x}SG + \underline{z})P,$$

where $\underline{z}$ is randomly chosen from $\mathcal{Z}$.

Decryption *A ciphertext $\underline{y}$ is decrypted by first computing $\underline{y}' = \underline{y}P^T = \underline{x}SG + \underline{z}$. Next, the syndrome $\underline{z}H^T$ is obtained by computing $\underline{y}'H^T$ and the corresponding error vector $\underline{z}$ is found in the syndrome-error table $\mathcal{T}$. Finally, the plaintext $\underline{x}$ is computed as $(\underline{y}' - \underline{z})(SG)^{-R}$.*

Key *The secret key consists of: the matrices S, P and G (H), and the syndrome-error table T $(\mathcal{Z})$.*

Although a universal cryptosystem has to protect the message without any assumption about the internal structure of the plaintext, not all plaintexts are allowed in the RN scheme. For example, for the plaintext $\underline{x} = \underline{0}$, each encryption yields an error vector. In this way, the set of permuted error vectors $\{\underline{z}P \mid \underline{z} \in \mathcal{Z}\}$ is obtained, and the average number of encryptions needed is given in the following lemma.

Lemma 8.2.2 *Let N be the cardinality of the set $\mathcal{Z}$ of error vectors. When the error vectors are selected according to a uniform distribution from $\mathcal{Z}$, then on the average $\sum_{i=0}^{l-1} N/(N-i)$ encryptions of the same plaintext are needed to obtain $l \leq N$ distinct ciphertexts. On* *the average $O(\mathsf{N}\log\mathsf{N})$ encryptions are needed to obtain all N distinct ciphertexts.*

Proof: Given $l < \mathsf{N}$ different ciphertexts. Then, with probability $(\mathsf{N} - l)/\mathsf{N}$ a new (distinct) ciphertext is found, so that, on the average $\mathsf{N}/(\mathsf{N}-l)$ encryptions are needed to obtain the $(l+1)$-th distinct ciphertext. Hence, in total $\sum_{i=0}^{l-1} \mathsf{N}/(\mathsf{N} - i)$ encryptions are needed to obtain l distinct ciphertexts. For $l = \mathsf{N}$, it follows that $\sum_{i=0}^{\mathsf{N}-1} \mathsf{N}/(\mathsf{N} - i) = \mathsf{N}\sum_{i=1}^{\mathsf{N}} i^{-1} = O(\mathsf{N}\log\mathsf{N})$ encryptions are needed to obtain all N distinct ciphertexts. □

The number of cosets equals $2^{n-k} = 2^{n(1-\mathrm{R})}$. Consequently, for a fixed code length, there is a trade-off between code rate and security. Moreover, since a syndrome-error table is used that requires $O(n2^{n-k})$ bits of storage, the value of N is restricted by memory limitations. To keep the secret key of moderate size, a simple algorithm for generating the possible error vectors is required. Therefore, Rao and Nam [RN87] first proposed to use a restricted set $\mathcal{Z}$ with so-called *Adjacent-t-Error* (ATE) vectors (i.e., a vector that consists of $n - t$ zeros and t consecutive ones). Since a codeword also has a distinct syndrome (i.e., $\underline{0}$), the additional restriction that an ATE should not be a codeword is unnecessary. Hin [Hin86a] analyzed the RN scheme with ATE vectors. His work was reported by Struik and van Tilburg [ST88] when his analysis was generalized for arbitrarily error vectors. They showed the following two main weaknesses of the RN scheme with ATEs as error vectors.

- The *burst-error structure* of $\mathcal{Z}$ (e.g., an ATE when cyclically shifted over one coordinate differs on two coordinates).
- The *low cardinality* of $\mathcal{Z}$ (e.g., for the binary case it equals n).

Rao and Nam's second proposal [RN87] used a randomly generated, predefined syndrome-error table as in Definition 8.2.1. In their analysis of this scheme, Struik and van Tilburg [ST88] observed that Definition 8.2.1 made no use of the error-correcting capability of the code. The only restrictions are that $k < n$ and that the matrix SG has full rank. As a result, the matrices S and P can be removed to facilitate easy analysis. In Chapter 9, it is shown that the set $\mathcal{Z}$ should be constructed such that it contains at most one randomly chosen vector from each coset of $\mathcal{C}$. The following secret-key cryptosystem is equivalent to the RN scheme:

Definition 8.2.3 (An Equivalent Rao-Nam Scheme) *Let $\mathcal{C}$ be a binary linear $[n,k]$-code $\mathcal{C}$. Let G be a $k \times n$ generator matrix of the code $\mathcal{C}$, and let H be an $(n-k) \times n$ parity-check matrix. Let the set of error vectors $\mathcal{Z}$ be a subset of $GF(2)^n$, such that, no distinct vectors in $\mathcal{Z}$ lie in the same coset of $\mathcal{C}$. Let $\mathcal{T}$ be the syndrome-error table: $\mathcal{T} = \{(\underline{z}H^T, \underline{z}) \mid \underline{z} \in \mathcal{Z}\}$.*

Encryption *A k-bit plaintext $\underline{x}$ is encrypted into an n-bit ciphertext $\underline{y}$ as:*

$$\underline{y} = \underline{x}G + \underline{z},$$

where $\underline{z}$ is randomly chosen from $\mathcal{Z}$.

Decryption *A ciphertext $\underline{y}$ is decrypted into a plaintext $\underline{x}$ by computing $\underline{y}H^T$ to obtain $\underline{z}H^T$. The corresponding error vector $\underline{z}$ is obtained from the syndrome-error table $\mathcal{T}$. Next, the plaintext $\underline{x}$ is computed as $\underline{x} = (\underline{y} - \underline{z})G^{-R}$.*

Key *The secret key consists of: the matrix G (H), and the syndrome-error table T $(\mathcal{Z})$.*

In the following analysis of this secret-key cryptosystem, the sum of two ciphertexts plays an essential role. Let $\underline{y}_1$ and $\underline{y}_2$ be two ciphertexts, then

$$\underline{y}_1 + \underline{y}_2 = (\underline{x}_1 G + \underline{z}_1) + (\underline{x}_2 G + \underline{z}_2) = (\underline{x}_1 + \underline{x}_2)G + (\underline{z}_1 + \underline{z}_2). \quad (8.4)$$

Thus, the sum of two ciphertexts is a codeword with the sum of two error vectors added. When $\underline{x}_1 = \underline{x}_2$, then the result is only the sum of two error vectors. And when $\underline{x}_1 + \underline{x}_2$ is a unit vector, then the result is a row of the secret generator matrix with the sum of two error vectors added to it. Therefore, first some properties of the set

$$\mathcal{Z} + \mathcal{Z} := \{\underline{z}_i + \underline{z}_j \mid 1 \leq i < j \leq \mathsf{N}\}$$

are proved before the algorithms that obtain the secret matrix G are described. Each n-bit vector in $\mathcal{Z}$ has a different syndrome and is randomly chosen from its corresponding coset. The next lemma gives the number of ways in which an arbitrarily n-bit vector can be written as the sum of two vectors of $\mathcal{Z}$ (i.e., the number of times the n-bit vector occurs in $\mathcal{Z}+\mathcal{Z}$).

Lemma 8.2.4 *Consider the cryptosystem in Definition 8.2.3. Let N_r denote the number of times an n-bit vector occurs r times in the set $\mathcal{Z}+\mathcal{Z}$. Then,*

$$N_r = \binom{N/2}{r} |\mathcal{C}|^{N/2}(|\mathcal{C}|-1)^{(N-2r)/2}. \tag{8.5}$$

Proof: Let $\underline{z}_i, \underline{z}_j \in \mathcal{Z}$ and $\underline{z}_i \neq \underline{z}_j$. The vector $\underline{z}_i + \underline{z}_j$ can not be a codeword since no error vectors lie in the same coset (i.e., the syndrome $(\underline{z}_i + \underline{z}_j)H^T \neq \underline{0}$). Denote the coset of the code $\mathcal{C}$ with parity-check matrix H and syndrome $\underline{s} \neq \underline{0}$ as:

$$\mathcal{Z}_{\underline{s}} = \{\underline{z} + \underline{c} \mid \underline{z}H^T = \underline{s} \text{ and } \underline{c} \in \mathcal{C}\}.$$

Consider two cosets $\mathcal{Z}_{\underline{s}_1}$ and $\mathcal{Z}_{\underline{s}_2}$ for which $\underline{s}_1 + \underline{s}_2 = \underline{s}$. Then, for exactly $|\mathcal{C}|$ pairs $(\underline{z}_1, \underline{z}_2) \in \mathcal{Z}_{\underline{s}_1} \times \mathcal{Z}_{\underline{s}_2}$, it holds that $\underline{z}_1 + \underline{z}_2 = \underline{z}$, and for the remaining $|\mathcal{C}|(|\mathcal{C}|-1)$ pairs the sum does not equal $\underline{z}$. For fixed $\underline{z}$, there are exactly $N/2$ ordered syndrome pairs $(\underline{s}_i, \underline{s}_j)$, such that $\underline{s} = \underline{s}_i + \underline{s}_j$ (no n-bit vector can occur more than $\lfloor N/2 \rfloor$ times in $\mathcal{Z}+\mathcal{Z}$). Hence, for a vector $\underline{z}$ with syndrome $\underline{s}$ occurring r times in the set $\mathcal{Z}+\mathcal{Z}$, there are exactly $\binom{N/2}{r}$ ordered syndrome pairs $(\underline{s}_i, \underline{s}_j)$, such that $\underline{s} = \underline{s}_i + \underline{s}_j$. From each of these r pairs, exactly $|\mathcal{C}|$ pairs $(\underline{z}_i, \underline{z}_j)$ can be chosen as error vectors. From the remaining $(N-2r)/2$ pairs, exactly $|\mathcal{C}|(|\mathcal{C}|-1)$ pairs can be chosen as error vectors, so that

$$N_r = \binom{N/2}{r} |\mathcal{C}|^r \{|\mathcal{C}|(|\mathcal{C}|-1)\}^{(N-2r)/2} = \binom{N/2}{r} |\mathcal{C}|^{N/2}(|\mathcal{C}|-1)^{(N-2r)/2},$$

which proves the lemma. □

This lemma is used in the proof of the next theorem where, for a rate R code, the probability is given that an arbitrarily n-bit vector is unique in the set $\mathcal{Z}+\mathcal{Z}$.

Theorem 8.2.5 *Consider the cryptosystem in Definition 8.2.3. Then, for sufficiently large n, the probability that an arbitrarily n-bit vector occurs exactly once in the set $\mathcal{Z} + \mathcal{Z}$ equals*

$$P_u = f(n, R)(e^{f(n,R)} - 1)^{-1}, \text{ where } f(n, R) = \frac{N}{2(|\mathcal{C}| - 1)}.$$

Proof: The number of times that an arbitrarily vector occurs once in the set $\mathcal{Z} + \mathcal{Z}$ equals N_1. Since the cardinality of $\mathcal{Z} + \mathcal{Z}$ is $\sum_{i=1}^{\mathsf{N}/2} \mathsf{N}_i$, the probability that an arbitrarily vector in $\mathcal{Z} + \mathcal{Z}$ occurs exactly once is $\mathsf{N}_1 / \sum_{i=1}^{\mathsf{N}/2} \mathsf{N}_i$. Substitute Equation (8.5), then

$$P_u = \frac{(\mathsf{N}/2)|\mathcal{C}|^{\mathsf{N}/2}(|\mathcal{C}| - 1)^{(\mathsf{N}-2)/2}}{|\mathcal{C}|^{\mathsf{N}} - (|\mathcal{C}|(|\mathcal{C}| - 1))^{\mathsf{N}/2}} = \frac{\mathsf{N}(|\mathcal{C}|(|\mathcal{C}| - 1))^{\mathsf{N}/2}}{2(|\mathcal{C}| - 1)|\mathcal{C}|^{\mathsf{N}} - (|\mathcal{C}|(|\mathcal{C}| - 1))^{\mathsf{N}/2}},$$

which further reduces to

$$\frac{\mathsf{N}}{2(|\mathcal{C}| - 1)(1 + \frac{1}{|\mathcal{C}|-1})^{(|\mathcal{C}|-1)\frac{\mathsf{N}}{2(|\mathcal{C}|-1)}} - 1}. \tag{8.6}$$

Next, substitute $f(n, \mathrm{R})$ for $\mathsf{N}/2(|\mathcal{C}| - 1)$, so that (8.6) becomes

$$\frac{f(n, \mathrm{R})}{(1 + \frac{1}{|\mathcal{C}|-1})^{(|\mathcal{C}|-1)f(n,\mathrm{R})} - 1}.$$

For sufficiently large n, this equals $f(n, \mathrm{R})(e^{f(n,\mathrm{R})} - 1)^{-1}$ which proves the theorem. □

For sufficiently large n, it follows that $f(n, \mathrm{R}) \approx \mathsf{N}/2|\mathcal{C}| \approx 2^{n(1-2\mathrm{R})}$. Then, when $\mathrm{R} > 1/2$, it follows that $f(n, \mathrm{R}) \to 0$ and its substitution in Theorem 8.2.5 shows that P_u is tightly approximated by $1 - f(n, \mathrm{R})/2$ when $n \to \infty$. When k vectors are chosen randomly then, with probability

$$\mathsf{P}_u^k = (1 - f(n, \mathrm{R})/2)^k \approx 1 - k2^{n(1-2\mathrm{R})} \to 1, \quad (n \to \infty),$$

these k vectors are unique in the set $\mathcal{Z} + \mathcal{Z}$. This proves the following corollary:

Corollary 8.2.6 *Consider the cryptosystem in Definition 8.2.3. For $[n,k]$-codes with information rate $R > 1/2$ and sufficiently large n, it holds with overwhelming probability that, k arbitrarily chosen n-bit vectors are unique in the set $\mathcal{Z}+\mathcal{Z}$.*

Moreover, the probability that N randomly chosen n-bit vectors are unique in the set $\mathcal{Z}+\mathcal{Z}$ equals

$$P_u^{\mathsf{N}} = (1 - f(n,\mathrm{R})/2)^{\mathsf{N}} \approx 1 - \mathsf{N}2^{n(1-2\mathrm{R})} = 1 - 2^{n(2-3\mathrm{R})},$$

which tends to 1 for codes with code rate $\mathrm{R} > 2/3$ and $n \to \infty$. This proves the following corollary:

Corollary 8.2.7 *Consider the cryptosystem in Definition 8.2.3. For $[n,k]$-codes with information rate $R > 2/3$ and sufficiently large n, it holds with overwhelming probability that, all vectors in $\mathcal{Z}+\mathcal{Z}$ are unique.*

The following algorithm is an improved version of the one presented in [ST88] (the reader is also referred to [Mei90]). To date, it is the only algorithm known that always recovers for the RN scheme an equivalent secret-key cryptosystem for all code rates R with a work- and memory-factor polynomially bounded in N (see Lemma 8.2.9). This algorithm uses the property that the difference of two ciphertexts results in a codeword with an added vector from $\mathcal{Z}+\mathcal{Z}$.

Algorithm 8.2.8 (Struik-Tilburg) *Consider the cryptosystem in Definition 8.2.3. The following three steps obtain a matrix E equivalent to the secret matrix G and a complete set of error vectors:*

Step 1 *Select at random a plaintext $\underline{x}$ and obtain the set $\mathcal{Y}^{(\underline{x})}$ of all N distinct ciphertexts. Let $\mathcal{Z}^{(\underline{x})}$ be the corresponding set of distinct error vectors, so that $\mathcal{Y}^{(\underline{x})} = \{\underline{x}G + \underline{z} \mid \underline{z} \in \mathcal{Z}^{(\underline{x})}\}$. Next, construct the (labelled) graph*

$$\Gamma^{(\underline{x})} = \left(\mathcal{Y}^{(\underline{x})}, \mathcal{Z}^{(\underline{x})} + \mathcal{Z}^{(\underline{x})}\right),$$

with vertices $\underline{y}_1^{(\underline{x})}, \underline{y}_2^{(\underline{x})}, \ldots, \underline{y}^{(\underline{x})}$, and the edge from $\underline{y}_i^{(\underline{x})}$ to $\underline{y}_j^{(\underline{x})}$ labelled with the sum of these vectors (i.e., with the vector $\underline{y}_i^{(\underline{x})} + \underline{y}_j^{(\underline{x})} = \underline{z}_i^{(\underline{x})} + \underline{z}_j^{(\underline{x})}$).

Step 2 *Let $\underline{u}_\nu$ be the ν-th, k-bit unit vector $(1 \leq \nu \leq k)$. Compute the plaintext $\underline{x}_\nu$ as $\underline{x}_\nu = \underline{x} + \underline{u}_\nu$. For $\nu \in \{1, 2, \ldots, k\}$, encipher each $\underline{x}_\nu$ until the set $\mathcal{Y}^{(\underline{x}_\nu)}$ of all N distinct ciphertexts are obtained. Next, construct for each ν the labelled graph*

$$\Gamma^{(\underline{x}_\nu)} = (\mathcal{Y}^{(\underline{x}_\nu)}, \mathcal{Z}^{(\underline{x}_\nu)} + \mathcal{Z}^{(\underline{x}_\nu)}), \quad 1 \leq \nu \leq k.$$

For each $\nu \in \{1, 2, \ldots, k\}$, compute a mapping ϕ^ν of the graph Γ^ν onto the graph Γ that leaves the edges invariant, that is, for all $\underline{y}_i^{(\underline{x}_\nu)}, \underline{y}_j^{(\underline{x}_\nu)} \in \mathcal{Y}^{(\underline{x}_\nu)}$:

$$\phi^\nu(\underline{y}_i^{(\underline{x}_\nu)}) + \phi^\nu(\underline{y}_j^{(\underline{x}_\nu)}) = \underline{y}_i^{(\underline{x}_\nu)} + \underline{y}_j^{(\underline{x}_\nu)}.$$

Step 3 *For each $\nu \in \{1, 2, \ldots, k\}$, compute the ν-th row of the E as:*

$$\underline{e}_\nu = \underline{y}_i^{(\underline{x}_\nu)} + \phi_\nu(\underline{y}_i^{(\underline{x}_\nu)}),$$

where $\underline{e}_\nu$ is the ν-th row of the matrix E. Compute the set $\mathcal{Z} = \{\underline{y} - \underline{x}E \mid \underline{y} \in \mathcal{Y}^{(\underline{x})}\}$.

Exit with the $k \times n$ matrix $E = (\underline{e}_1^T, \underline{e}_2^T, \ldots, \underline{e}_k^T)^T$ and the set of error vectors $\mathcal{Z}$.

The work-factor of the algorithm is dominated by the computation of the mapping (translation) ϕ^ν in Step 2. To obtain a translation ϕ^ν, proceed as follows. First, sort the set $\mathcal{Y}^{(\underline{x})}$, which requires $O(\mathsf{N} \log \mathsf{N})$ n-bit vector operations. Second, select a vector $\underline{y}_1^{(\underline{x}_\nu)}$ of the set $\mathcal{Y}^{(\underline{x}_\nu)}$. The vector $\underline{y}_1^{(\underline{x}_\nu)}$ can be mapped onto N different vectors of the set $\mathcal{Y}^{(\underline{x})}$. Choose one of these vectors, for example $\underline{y}_i^{(\underline{x})}$. Let ψ be this possible candidate mapping:

$$\psi(\underline{y}^{(\underline{x}_\nu)}) = \underline{y}^{(\underline{x}_\nu)} + (\underline{y}_i^{(\underline{x})} + \underline{y}_1^{(\underline{x}_\nu)}).$$

Since ψ is a translation, ψ is injective. Check if ψ is also surjective by verifying if the remaining $\mathsf{N} - 1$ vectors of $\mathcal{Y}^{(\underline{x}_\nu)}$ are also mapped onto $\mathcal{Y}^{(\underline{x})}$. If ψ is surjective, then a usable mapping ϕ is found, otherwise, map $\underline{y}_1^{(\underline{x}_\nu)}$ onto another vector in $\mathcal{Y}^{(\underline{x})}$ and verify again. This procedure requires $\mathsf{N}(\mathsf{N} - 1) \log \mathsf{N}$ n-bit vector operations in the worst case. Since the algorithm requires k such mappings, an average work-factor W of $O(kn\mathsf{N}^2 \log \mathsf{N})$ bit operations is needed. Since the sets $\mathcal{Y}^{(\underline{x}_\nu)}$ and $\mathcal{Y}^{(\underline{x})}$ need to be stored, the memory-factor M equals $O(n\mathsf{N})$ bits. For each column a plaintext needs to be encrypted $O(\mathsf{N} \log \mathsf{N})$ times, so that a total of $O(k\mathsf{N} \log \mathsf{N})$ encryptions are needed. This proves the following lemma:

Lemma 8.2.9 *Algorithm 8.2.8 requires $O(k\mathsf{N} \log \mathsf{N})$ encryptions. It has a work-factor W of $O(kn\mathsf{N}^2 \log \mathsf{N})$ bit operations, and a memory-factor M that equals $O(n\mathsf{N})$ bits.*

When a vector in $\mathcal{Y}^{(\underline{x}_\nu)}$ is found that does not map onto the set $\mathcal{Y}^{(\underline{x})}$, the function ψ is rejected. For codes with a high information rate, Corollary 8.2.7 shows that, with overwhelming probability, all vectors of $\mathcal{Z} + \mathcal{Z}$ are unique. This implies that the decision whether ψ is suitable or not is almost instantaneous. Therefore, the work-factor of Algorithm 8.2.8 becomes $\mathsf{W} = O(kn\mathsf{N} \log \mathsf{N})$ bit operations. This yields the following:

Corollary 8.2.10 *For codes with information rate $R > 2/3$, Algorithm 8.2.8 requires an average number of $O(k\mathsf{N} \log \mathsf{N})$ encryptions, has a work-factor W of $O(kn\mathsf{N} \log \mathsf{N})$ bit operations and a memory-factor M that equals $O(n\mathsf{N})$ bits.*

Based on Corollary 8.2.7, Meijers and van Tilburg [MT91a] proposed the following improvement of Algorithm 8.2.8. When the set $\mathcal{Z}$ is unique, only three ciphertexts are needed in Step 2 of Algorithm 8.2.8 in order to find a vector (translation) that maps the three ciphertexts onto the set $\mathcal{Y}^{(\underline{x})}$. Since only a few of the N ciphertexts are needed, the work-factor and the number of encryptions are drastically reduced. The costs associated with this modified Algorithm 8.2.8 are summarized in the following:

Corollary 8.2.11 *For codes with information rate $R > 2/3$, the modified Algorithm 8.2.8 requires an average number of $O(\mathsf{N} \log \mathsf{N})$ encryptions, and has a work-factor W of $O(kn\mathsf{N} \log \mathsf{N})$ bit operations and a memory-factor M that equals $O(n\mathsf{N})$ bits.*

Heiman [Hei87] described an algorithm based on the following observation: When $\underline{y}_1$ and $\underline{y}_2$ are ciphertexts of the same (unknown) plaintext, then

$$\underline{y}_1 + \underline{y}_2 = (\underline{x}G + \underline{z}_1) + (\underline{x}G + \underline{z}_2) \in \mathcal{Z} + \mathcal{Z}.$$

For codes with rate R > 2/3 (Corollary 8.2.7), the set $\mathcal{Z} + \mathcal{Z}$ can be obtained by encrypting $O(\mathsf{N}^2 \log \mathsf{N})$ plaintexts two times. Then, from the set $\mathcal{Z} + \mathcal{Z}$, the labelled graph $\Gamma^{\underline{x}}$ is computed for some unknown plaintext $\underline{x}$. From $\Gamma^{\underline{x}}$ and three encryptions of two known plaintexts, each row of the encryption matrix can be computed. Meijers [Mei90] gave a detailed analysis of Heiman's algorithm and proved the following:

Lemma 8.2.12 (Meijers [Mei90]) *Heiman's method [Hei87] is successful for codes with rate $R > 2/3$. It needs $O(N^2 \log N)$ encryptions, requires a work-factor of $O(nN^3 \log N)$ bit operations, and needs a memory-factor of $O(nN^2)$ bits.*

In order to prevent the previously mentioned cryptanalysis, Rao and Nam [RN89] proposed using error-correcting codes with minimum distance of ≤ 6 and length of at most 250 bits. They also proposed to use byte error-correcting codes such as Reed-Solomon codes or nonlinear codes such as Preparata codes. Note that the use of non-linear codes renders the RN scheme less efficient. The RN scheme with Preparata codes was described in [Den88]. Struik [Str91] demonstrated that this non-linear scheme is also vulnerable to cryptanalysis. Another system was proposed by Hwang and Rao [HR90b] and also suffers from a similar cryptanalysis.

Despite the remark of Rao [Rao88] about the analysis in [ST88], Algorithm 8.2.8 is robust in the sense that the algorithm always finds the secret matrix and the complete set of error vectors for any linear code. For large code lengths of 250 bits and high information rates, Algorithm 8.2.8 requires a unnecessarily high work-factor. However, the objective of the RN *secret-key* cryptosystem was to use simpler and smaller codes that require less storage and process at higher speeds than used in the McEliece *public-key* cryptosystem (i.e., a 1024 bits binary Goppa code with rate approximately 0.64). Therefore, the gain of using codes of length 250 bits in the RN scheme seems questionable.

Also, in this case, a simple error generating function is needed to make an implementation feasible. Despite all these problems, Meijers and van Tilburg [MT91b] showed that, even for these high code lengths, the security level of the RN cryptosystem is not sufficient. They used the following probabilistic approach based on the Birthday Paradox [Sim92], in which it is beneficial to have a large cardinality of the set of error vectors $\mathcal{Z}$.

Algorithm 8.2.13 (Meijers-Tilburg) *Consider the cryptosystem in Definition 8.2.3. Let a and b be two positive integers with values approximately $\sqrt{N}$. The following four steps obtain a matrix E equivalent to the secret matrix G and a set $\mathcal{E}$ of $a + kb = O(k\sqrt{N})$ error vectors:*

Step 1 *Select at random a k-bit plaintext and obtain a set $\mathcal{Y}_a^{(\underline{x})}$ of $|\mathcal{Y}_a^{(\underline{x})}| = a$ distinct n-bit ciphertexts. Let $\mathcal{Z}_a^{(\underline{x})}$ be the corresponding set of distinct error vectors, so that*

$$\mathcal{Y}_a^{(\underline{x})} = \{\underline{x}G + \underline{z} \mid \underline{z} \in \mathcal{Z}_a^{(\underline{x})}\}.$$

Next, construct the labelled graph

$$\Gamma_a^{(\underline{x})} = \left(\mathcal{Y}_a^{(\underline{x})}, \mathcal{Z}_a^{(\underline{x})} + \mathcal{Z}_a^{(\underline{x})}\right),$$

with vertices $\underline{y}_1^{(\underline{x})}, \underline{y}_2^{(\underline{x})}, \ldots, \underline{y}_a^{(\underline{x})}$, and the edge from $\underline{y}_i^{(\underline{x})}$ to $\underline{y}_j^{(\underline{x})}$ labelled with the sum of these vectors (i.e., with the vector $\underline{y}_i^{(\underline{x})} + \underline{y}_j^{(\underline{x})} = \underline{z}_i^{(\underline{x})} + \underline{z}_j^{(\underline{x})}$).

Step 2 *Let $\underline{u}_\nu$ be the ν-th, k-bit unit vector $(1 \leq \nu \leq k)$. Compute the plaintext $\underline{x}_\nu$ as $\underline{x}_\nu = \underline{x} + \underline{u}_\nu$. For $\nu \in \{1, 2, \ldots, k\}$, encipher each $\underline{x}_\nu$ until a set $\mathcal{Y}_b^{(\underline{x}_\nu)}$ of b distinct ciphertexts are obtained. Next, construct the corresponding labelled graphs*

$$\Gamma_b^{(\underline{x}_\nu)} = (\mathcal{Y}_b^{(\underline{x}_\nu)}, \mathcal{Z}_b^{(\underline{x}_\nu)} + \mathcal{Z}_b^{(\underline{x}_\nu)}), \quad 1 \leq \nu \leq k.$$

In this way, k labelled graphs $\Gamma_b^{(\underline{x}_\nu)}$ are obtained.

Step 3 *With overwhelming probability, there exist integers i, j and h, such that*

$$\underline{y}_i^{(\underline{x}_\nu)} + \underline{y}_h^{(\underline{x}_\nu)} \in \mathcal{Z}_a^{(\underline{x})} + \mathcal{Z}_a^{(\underline{x})} \text{ and } \underline{y}_j^{(\underline{x}_\nu)} + \underline{y}_h^{(\underline{x}_\nu)} \in \mathcal{Z}_a^{(\underline{x})} + \mathcal{Z}_a^{(\underline{x})}.$$

Moreover, with overwhelming probability, there exist integers r, s and t, such that

$$\underline{z}_h^{(\underline{x}_\nu)} + \underline{z}_i^{(\underline{x}_\nu)} = \underline{z}_r^{(\underline{x})} + \underline{z}_s^{(\underline{x})}, \text{ and } \underline{z}_h^{(\underline{x}_\nu)} + \underline{z}_j^{(\underline{x}_\nu)} = \underline{z}_r^{(\underline{x})} + \underline{z}_t^{(\underline{x})},$$

So that $\underline{z}_h^{(\underline{x}_\nu)} = \underline{z}_r^{(\underline{x})}$. For $1 \leq \nu \leq k$, compute the ν-th row of the E as:

$$\underline{e}_\nu = \underline{y}_h^{(\underline{x}_\nu)} + \underline{y}_r^{(\underline{x})} = \underline{g}_\nu + (\underline{z}_h^{(\underline{x}_\nu)} + \underline{z}_r^{(\underline{x})}).$$

Compute the set

$$\mathcal{E} = \{\underline{y} - \underline{x}E \mid \underline{y} \in \mathcal{Y}^{(\underline{x})} \cup \mathcal{Y}^{(\underline{x}_1)} \cup \mathcal{Y}^{(\underline{x}_2)} \cup \cdots \cup \mathcal{Y}^{(\underline{x}_k)}\}.$$

Step 4 *Exit with the $k \times n$ matrix $E = (\underline{e}_1^T, \underline{e}_2^T, \ldots, \underline{e}_k^T)^T$ and the set of error vectors $\mathcal{E}$.*

The choice of the values of a and b depends on the parameters n and k of the underlying code. If $a, b = O(\sqrt{\mathsf{N}})$, then based on the Birthday Paradox, the sets $\mathcal{Z}_a^{(\underline{x})}$ and $\mathcal{Z}_b^{(\underline{x}_\nu)}$ have with overwhelming probability at least three vectors in common (see also [Mei90]). For codes with high information rate, the vectors of $\mathcal{Z}_a^{(\underline{x})} + \mathcal{Z}_a^{(\underline{x})}$ and $\mathcal{Z}_b^{(\underline{x}_\nu)} + \mathcal{Z}_b^{(\underline{x}_\nu)}$ are unique with overwhelming probability. This means that a vector of $\mathcal{Z}_b^{(\underline{x}_\nu)} + \mathcal{Z}_b^{(\underline{x}_\nu)}$ is the sum of two vectors $\underline{z}_i$ and $\underline{z}_j$ in the set $\mathcal{Z}_b^{(\underline{x}_\nu)}$.

From Lemma 8.2.2 it follows that on the average

$$\sum_{i=0}^{\sqrt{\mathsf{N}}-1} \frac{\mathsf{N}}{\mathsf{N}-i} < \frac{\mathsf{N}\sqrt{\mathsf{N}}}{\mathsf{N}-(\sqrt{\mathsf{N}}-1)} = O(\sqrt{\mathsf{N}})$$

encryptions of the same plaintext are needed to obtain $\sqrt{\mathsf{N}}$ distinct ciphertexts. An efficient implementation of this algorithm is obtained when the labelled graph in Step 1 is stored as a sorted list. Then Step 1 dominates the work-factor and requires $O(\mathsf{N} \log \mathsf{N})$ n-bit vector operations for the sorting procedure and $O(n\mathsf{N})$ bits of memory for storage of the sorted list.

Lemma 8.2.14 *For codes with rate $R > 1/2$, Algorithm 8.2.13 is successful with overwhelming probability. It requires $O(k\sqrt{\mathsf{N}})$ encryptions on the average, has a work-factor W of $O(kn\mathsf{N} \log \mathsf{N})$ bit operations, and a memory-factor M of $O(n\mathsf{N})$ bits.*

Not all possible error patterns are obtained in this way. However, to keep the secret key of moderate size, a simple algorithm that generates the possible error vectors must be used. In general, it is reasonable to assume that $O(k\sqrt{\mathsf{N}})$ known error vectors are sufficient to reconstruct this algorithm. However, if this is not the case, then the system's security is based on this error-vector generating algorithm. In the worst case, $O(\mathsf{N}\log\mathsf{N})$ known encryptions are needed to reconstruct the complete syndrome-error table.

Similar proposals using codes that are capable of correcting burst-errors, can be found in [Roc92, ALCdS93] and [CdSCdS94]. These schemes are a combination of the SCE scheme and the RN scheme, and therefore suffer from similar cryptanalysis to that previously mentioned. In these papers, it was not observed that since P is part of the secret key, it is not necessary for this matrix to be a permutation matrix. Only when the cryptosystem must be able to correct channel errors, then P should be a permutation matrix.

To avoid analysis that make use of the linearity as in Equation (8.4), Struik and van Tilburg [ST88] proposed to combine the plaintext with the error vector by a non-linear function f which may also be key dependent. In this case, the encryption function in Definition 8.2.3 becomes:

$$\underline{y} = f_{\underline{z}}(\underline{x})G + \underline{z}, \text{ where } f_{\underline{z}}(\underline{x}) = f(\underline{x}, \underline{z}). \qquad (8.7)$$

Since in the decryption procedure the error vector $\underline{z}$ is obtained, the plaintext can be retrieved by computing $\underline{x} = f_{\underline{z}}^{-1}((\underline{y} - \underline{z})G^{-R})$. Hwang and Rao [HR90a] extended this idea by also replacing the syndrome-error table with non-linear function. In conclusion, these non-linear functions should be chosen carefully and they do not guarantee an improved security. For example, different analysis might be found that is based on its non-linearity as shown for the Li-Wang scheme.

8.3 The Li-Wang Scheme

Li and Wang [LW91] proposed a secret-key cryptosystem that uses the concealed channel in the RN scheme for authentication purposes. Instead of randomly choosing an error vector from the set $\mathcal{Z}$ (as in the RN scheme), an error-selection function is used which assigns to each plaintext only one fixed error vector from $\mathcal{Z}$. As noted, redundancy in the concealed channel degrades the security of the system. In [Til93c], the author showed how to take advantage of this redundancy to make the Li-Wang (LW) scheme less secure than the RN scheme. This section describes the LW scheme (for code rates R $\geq 1/2$) in a standardized form in order to facilitate easy analysis (see also [Til93c]). In Definition 8.3.2, the following error function is used:

Definition 8.3.1 (Error-Selection Function) *Let $n-k<k$, and let $\mathcal{A}$ be a given subset $\{a_1, a_2, \ldots, a_{n-k}\}$ of $\{1, 2, \ldots, k\}$ with $1 \leq a_1 < a_2 < \ldots < a_{(n-k)} \leq k$. The subvector $\underline{x}_{\mathcal{A}}$ of the n-bit vector $\underline{x}$ is defined as:*

$$\underline{x}_{\mathcal{A}} := \sigma_{\mathcal{A}}(\underline{x}) = (x_{a_1}, x_{a_2}, \ldots, x_{a_{(n-k)}}).$$

Then, the error-selection function $f_{\mathcal{A}}$ assigns to each k-bit plaintext $\underline{x}$ an n-bit error vector $\underline{z}$ from $\mathcal{Z}$:

$$f_{\mathcal{A}}(\underline{x}) = \underline{z} \in \mathcal{Z}, \quad \underline{z}H^T = \sigma_{\mathcal{A}}(\underline{x}),$$

and no distinct vectors in $\mathcal{Z}$ lie in the same coset of C.

From this definition, it follows that $f_{\mathcal{A}}(\underline{x})H^T = \underline{z}H^T = \sigma_{\mathcal{A}}(\underline{x})$. It is important to note that two different plaintexts might be mapped on the same error vector as $\underline{z}$ do not uniquely determine $\underline{x}$.

The LW scheme is obtained when the random selection of the error vector $\underline{z}$, in Definition 8.2.1 (RN scheme), is replaced by the above error-selection function $f_{\mathcal{A}}$. It can be shown that the following secret-key cryptosystem is equivalent to the LW scheme (i.e., using similar reasoning as used for the RN scheme). This equivalent scheme is then used in the cryptanalysis of the LW scheme.

Definition 8.3.2 (An Equivalent Li-Wang Scheme) *Let C be a binary linear $[n,k]$-code. Let G be a generator matrix of code C with a right inverse G^{-R} (i.e., $GG^{-R} = I_k$). Let H be a parity-check matrix for C. Define the error-selection function $f_{\mathcal{A}}$ as in Definition 8.3.1. Let $\mathcal{T}$ be the syndrome-error table: $\mathcal{T} = \{(\underline{z}H^T, \underline{z}) \mid \underline{z} \in \mathcal{Z}\}$.*

Encryption *A k-bit plaintext $\underline{x}$ is encrypted into an n-bit ciphertext $\underline{y}$ as:*

$$\underline{y} = \underline{x}G + f_{\mathcal{A}}(\underline{x}).$$

Decryption *A ciphertext $\underline{y}$ is decrypted into a plaintext $\underline{x}$ by computing $\underline{y}H^T = \sigma_{\mathcal{A}}(\underline{x})$ which uniquely determines $\underline{z}$ by using the syndrome-error table $\mathcal{T}$. Next, the error vector $\underline{z}$ is subtracted from $\underline{y}$ to obtain $\underline{x}G$. Then, the plaintext $\underline{x}$ is computed as $\underline{x} = (\underline{x}G)G^{-R}$.*

Key *The secret key consists of: the matrix G (H) and the error-selection function $f_{\mathcal{A}}(\underline{x})$.*

First, the following analysis shows how to obtain a matrix E equivalent to the secret generator matrix G. The analysis uses $O(k)$ chosen encryptions and is successful with overwhelming probability. Second, depending on the way the set of error vectors $\mathcal{Z}$ was generated, at most $|\mathcal{Z}|$ chosen encryptions are required to construct a corresponding error-selection function $e_{\mathcal{A}}$. The matrix E and the error-selection function $e_{\mathcal{A}}$ together define a secret-key cryptosystem equivalent to the LW scheme. The first algorithm used in this analysis is based on the following theorem. This theorem proves that the sum of three ciphertexts, obtained from two randomly chosen plaintexts, yields a codeword.

Theorem 8.3.3 *Consider the cryptosystem in Definition 8.3.2. Let $\underline{y}_i$, $\underline{y}_j$ and $\underline{y}_{ij}$ be ciphertexts of $\underline{x}_i$, $\underline{x}_j$ and $(\underline{x}_i + \underline{x}_j)$, respectively. Then,*

$$\underline{y}_i + \underline{y}_j + \underline{y}_{ij} \in \mathcal{C}.$$

Proof: Since $\sigma_{\mathcal{A}}$ is a linear function, the syndrome $(\underline{y}_i + \underline{y}_j + \underline{y}_{ij})H^T = f_{\mathcal{A}}(\underline{x}_i)H^T + f_{\mathcal{A}}(\underline{x}_j)H^T + f_{\mathcal{A}}(\underline{x}_i + \underline{x}_j)H^T = \sigma_{\mathcal{A}}(\underline{x}_i) + \sigma_{\mathcal{A}}(\underline{x}_j) + \sigma_{\mathcal{A}}(\underline{x}_i + \underline{x}_j) = \underline{0}$. A vector $\underline{y}$ is in $\mathcal{C} \iff \underline{y}H^T = \underline{0}$, hence $\underline{y}_i + \underline{y}_j + \underline{y}_{ij}$ is a codeword. □

Recall that any set of k linearly independent codewords can be used as a generator matrix for $\mathcal{C}$. Therefore, the following algorithm exits with a set of k linearly independent codewords. These codewords form a $k \times n$ matrix E which generates the same code $\mathcal{C}$ as G does.

Algorithm 8.3.4 *Using the ciphertexts as defined in Theorem 8.3.3, the following four steps obtain a matrix E equivalent to the secret matrix G:*

Step 1 *Select at random two different plaintexts $\underline{x}_i$ and $\underline{x}_j$, and obtain the corresponding ciphertexts: $\underline{y}_i, \underline{y}_j$ and $\underline{y}_{ij}$.*

Step 2 *Compute the codeword $\underline{c} = \underline{y}_i + \underline{y}_j + \underline{y}_{ij}$.*

Step 3 *Repeat Steps 1 and 2 until k linearly independent codewords $(\underline{c}_1, \underline{c}_2, \ldots, \underline{c}_k)$ are obtained.*

Step 4 *Exit with the $k \times n$ matrix $E = (\underline{c}_1^T, \underline{c}_2^T, \ldots, \underline{c}_k^T)^T$.*

This algorithm is not successful when applied to the RN scheme, because in their scheme it is impossible to choose three ciphertexts that guarantee a codeword.

Next, for Algorithm 8.3.4, estimate the average number of necessary iterations in order to exit with the matrix E. Consider k randomly selected k-bit vectors $\underline{x}_i$ and k corresponding n-bit codewords $\underline{c}_i$, such that $\underline{x}_i E = \underline{c}_i$ $(1 \leq i \leq k)$. Let $X = (\underline{x}_1^T, \underline{x}_2^T, \ldots, \underline{x}_k^T)^T$ and $C = (\underline{c}_1^T, \underline{c}_2^T, \ldots, \underline{c}_k^T)^T$. If matrix X is non-singular, then the full-rank $k \times n$ matrix E is computed from $E = X^{-1}C$. The probability q_k that X has full rank, given k randomly chosen vectors $\underline{x}_i$, equals $\prod_{i=1}^{k}(1 - 2^{-i})$. Matrix C has full rank if, and only if, matrix X has full rank. Thus, if k codewords are randomly selected, then with probability $q_k \approx 0.289$ $(k > 10)$ a linearly independent set of k codewords will be obtained. Therefore, the expected number of vectors to choose before their rank equals k is less than $q_k^{-1}k = O(k)$. In Step 1 of Algorithm 8.3.4, two different plaintexts $\underline{x}_i$ and $\underline{x}_j$ are randomly selected. Since set $\mathcal{Z}$ is randomly chosen, it follows from Theorem 8.3.3 that Step 2 yields a random codeword. If Steps 1 and 2 are repeated $O(k)$ times, then with overwhelming probability $(|\mathcal{Z}| \gg k)$ k linearly independent codewords are obtained. In Step 1, exactly three ciphertexts are required. Therefore, $3O(k) = O(k)$ encryptions are needed for Algorithm 8.3.4 to be successful with overwhelming probability. Step 2 requires a work-factor of $O(kn^2)$ bit operations and uses $O(kn)$ bits of memory. This proves the following lemma:

Lemma 8.3.5 *With overwhelming probability Algorithm 8.3.4 terminates in $O(k)$ iterations. Then on the average, the algorithm requires $O(k)$ chosen encryptions, has a work-factor W of $O(kn^2)$ bit operations and has a memory-factor M of $O(kn)$ bits.*

To date, no algorithm is known to exist for the RN scheme that recovers a secret matrix G, or equivalent, within $O(k)$ chosen encryptions. As stated in [RN89], this is the basic problem of the RN scheme. For example, once a generator matrix is obtained, it takes at most $O(|\mathcal{Z}|\log|\mathcal{Z}|)$ steps before the set of error vectors follows.

Once the basic problem has been solved, the other secret-key elements can be obtained with less effort. For example, to recover the secret set $\mathcal{A}$ in Definition 8.3.1, proceed as follows. Solving $ED^T = O_{k,(n-k)}$ yields a parity-check matrix D for $\mathcal{C}$, so that $GD^T = O_{k,(n-k)}$. Let the n-bit vector $\underline{u}_i$ denote the i-th unit vector. If $\underline{s}_i = \underline{s}$, then it follows that $f_{\mathcal{A}}(\underline{x}) = f_{\mathcal{A}}(\underline{x}+\underline{u}_i)$. As a result, $\underline{y}+\underline{y}_i = \underline{u}_i G = \underline{g}_i$ (i.e, the i-th row of the generator matrix G). The following algorithm recovers the set $\mathcal{A}$ in Definition 8.3.2. In addition, it recovers $2k-n$ rows of the secret matrix G.

Algorithm 8.3.6 *Consider the cryptosystem in Definition 8.3.2. The following two steps recover the set $\mathcal{A}$ and obtain $2k-n$ rows of the secret matrix G:*

Step 1 *Let $\mathcal{A} := \emptyset$. Select at random a plaintext $\underline{x}$ and obtain its corresponding ciphertext $\underline{y} = \underline{x}G + f_{\mathcal{A}}(\underline{x})$. Next, compute the syndrome $\underline{s}$ as $\underline{y}D^T$.*

Step 2 *For $1 \leq i \leq k$, obtain for each plaintext $(\underline{x}+\underline{u}_i)$ the corresponding ciphertext $\underline{y}_i = (\underline{x}+\underline{u}_i)G + f_{\mathcal{A}}(\underline{x}+\underline{u}_i)$. Next, compute for each $\underline{y}_i$ the syndrome: $\underline{s}_i = \underline{y}_i D^T$.*

If $\underline{s}_i = \underline{s}$, then $\underline{g}_i = \underline{y} + \underline{y}_i$, otherwise let $\mathcal{A} = \mathcal{A} \cup \{i\}$.

The matrix E obtained in Algorithm 8.3.4 can be updated, such that, the rows $\underline{e}_i$ with $i \in \mathcal{A}$ are equal to the rows $\underline{g}_i$. With knowledge of the set $\mathcal{A}$, the corresponding set of error vectors can be obtained in at most $|\mathcal{Z}|$ chosen-plaintexts:

$$\underline{y} = \underline{x}G + f_{\mathcal{A}}(\underline{x}), \text{ so that } e_{\mathcal{A}}(\underline{x}) = \underline{y} - \underline{x}E.$$

Note that with only r error vectors, $r2^{2k-n}$ plaintexts can always be encrypted and decrypted.

In [Til93c], the author gave another algorithm for analyzing the LW scheme. This algorithm does not use the linearity of the $\sigma_{\mathcal{A}}$-function:

$$\sigma_{\mathcal{A}}(\underline{x}_i + \underline{x}_j) = \sigma_{\mathcal{A}}(\underline{x}_i) + \sigma_{\mathcal{A}}(\underline{x}_j).$$

However, it uses the non-linearity of the error-selection function:

$$f_{\mathcal{A}}(\underline{x}_i + \underline{x}_j) \neq f_{\mathcal{A}}(\underline{x}_i) + f_{\mathcal{A}}(\underline{x}_j).$$

This algorithm obtains $2k - n$ rows of the $k \times n$ secret matrix G and needs at most k^2 chosen encryptions. It is based on the following:

Lemma 8.3.7 *Consider the cryptosystem in Definition 8.3.2. Let the k-bit vectors $\underline{u}_i$ and $\underline{u}_j$ denote the i-th and j-th unit vector, respectively. Let $\underline{g}_j$ be the j-th row of the secret $k \times n$ matrix G. Let $\underline{y}_i$ and $\underline{y}_{ij}$ be ciphertexts of the plaintexts $\underline{u}_i$ and $(\underline{u}_i + \underline{u}_j)$, respectively. Then, $\sigma_{\mathcal{A}}(\underline{u}_j) = \underline{0}$ if, and only if,*

$$\forall_{i,1 \leq i \leq k} \quad \underline{y}_{ij} + \underline{y}_i = \underline{g}_j.$$

There are exactly $2k - n$ vectors $\underline{u}_j$ that satisfy $\sigma_{\mathcal{A}}(\underline{u}_j) = \underline{0}$.

Proof: Compute $\underline{y}_{ij} + \underline{y}_i = \underline{u}_j G + f_{\mathcal{A}}(\underline{u}_i + \underline{u}_j) + f_{\mathcal{A}}(\underline{u}_i)$. Because $\sigma_{\mathcal{A}}(\underline{u}_j) = \underline{0} \iff f_{\mathcal{A}}(\underline{u}_i + \underline{u}_j) + f_{\mathcal{A}}(\underline{u}_i) = \underline{0}$ for all i $(1 \leq i \leq k)$, it follows that, $\forall_{i,1 \leq i \leq k} \; [\underline{y}_{ij} + \underline{y}_i = \underline{u}_j G = \underline{g}_j] \iff \sigma_{\mathcal{A}}(\underline{u}_j) = \underline{0}$.

The function $\sigma_{\mathcal{A}}(\underline{u}_j)$ results in the $(n-k)$-bit all-zero vector if, and only if, the set $\mathcal{A}$ selects $n - k$ coordinates from $\underline{u}_j$ that are all equal to zero. Therefore, $k - (n - k) = 2k - n$ unit vectors $\underline{u}_j$ exist with $\sigma_{\mathcal{A}}(\underline{u}_j) = \underline{0}$. □

The next theorem provides an efficient method to independently recover $2k - n$ rows of the encryption matrix G.

Theorem 8.3.8 *Consider the cryptosystem in Definition 8.3.2. Let the ciphertexts be the same as defined in Lemma 8.3.7. The vector $\underline{e}_j = \underline{y}_{1j} + \underline{y}_1$ is equal to the j-th row $\underline{g}_j$ of the secret matrix G if*

$$\forall_{i,2 \leq i \leq k} \quad \underline{y}_{ij} + \underline{y}_i = \underline{e}_j. \tag{8.8}$$

There are exactly $2k - n$ rows $\underline{g}_j$ that can be obtained in this way.

Proof: If $\sigma_{\mathcal{A}}(\underline{u}_j) = \underline{0}$, then for all unit vectors $\underline{u}_i$ $(1 \le i \le k)$, it follows from Lemma 8.3.7 that $\underline{y}_i + \underline{y}_{ij} = \underline{g}_j + f_{\mathcal{A}}(\underline{u}_i) + f_{\mathcal{A}}(\underline{u}_i + \underline{u}_j) = \underline{g}_j$. Hence, $\underline{e}_j = \underline{g}_j$.

Suppose $\underline{u}_j$ is a unit vector, such that $\sigma_{\mathcal{A}}(\underline{u}_j) \neq \underline{0}$. According to Lemma 8.3.7, exactly $2k - n$ vectors $\underline{u}_i$ can be selected, such that $\underline{y}_i + \underline{y}_{ij} = \underline{g}_j + f_{\mathcal{A}}(\underline{0}) + f_{\mathcal{A}}(\underline{u}_j) = \underline{e}_j$. All other unit vectors, for example $\underline{u}_s$, must satisfy $\underline{y}_s + \underline{y}_{sj} = \underline{g}_j + f_{\mathcal{A}}(\underline{u}_s) + f_{\mathcal{A}}(\underline{u}_s + \underline{u}_j)$. Because $f_{\mathcal{A}}$ is a non-linear function, $f_{\mathcal{A}}(\underline{0}) + f_{\mathcal{A}}(\underline{u}_j)$ can not match $f_{\mathcal{A}}(\underline{u}_s) + f_{\mathcal{A}}(\underline{u}_s + \underline{u}_j)$ for all values of s.

In accordance with Lemma 8.3.7, exactly $2k - n$ unit vectors have $\sigma_{\mathcal{A}}(\underline{u}_j) = \underline{0}$. So exactly $2k - n$ rows of G can be obtained in this way. □

Theorem 8.3.8 states that if a row estimate $\underline{e}_j$ satisfies (8.8), then $\underline{e}_j$ is the j-th row of the secret matrix G. The next algorithm uses (8.8) to verify whether or not a row estimate is correct.

Algorithm 8.3.9 *Using the ciphertexts as defined in Theorem 8.3.8, let* $r = 1$, $j = 1$, *and obtain the ciphertext* $\underline{y}_1$. *The following three steps obtain* $2k - n$ *rows of the secret matrix* G*:*

Step 1 *Obtain the ciphertext* $\underline{y}_{1j}$, *and compute* $\underline{e}_j = \underline{y}_{1j} + \underline{y}_1$. *Obtain the ciphertext* $\underline{y}_{ij}$, *and if* $\underline{y}_{ij} + \underline{y}_i = \underline{e}_j$ *holds for all* $i \in \{2, 3, \ldots k\}$, *then let* $b_r = j$, $\underline{g}_{b_r} = \underline{e}_j$ *and* $r = r + 1$.

Step 2 *If* $r \le 2k - n$, *then let* $j = j + 1$ *and return to Step 1.*

Step 3 *Exit with the* $(2k - n) \times n$ *matrix* $G_{\mathcal{B}} = (\underline{g}_{b_1}^T, \underline{g}_{b_2}^T, \ldots, \underline{g}_{b_{2k-n}}^T)^T$ *and the corresponding set* $\mathcal{B} = \{b_1, b_2, \ldots, b_{2k-n}\}$.

Note that $|\mathcal{A}| + |\mathcal{B}| = k$ and $\mathcal{A} \cap \mathcal{B} = \emptyset$. Since there are exactly $2k - n$ rows of the secret matrix G that satisfy the verification condition (8.8), at most k row estimates $\underline{e}_j$ must be considered (i.e., Step 1 is repeated $\le k$ times). In Step 1, at most k encryptions are required, so that Algorithm 8.3.9 requires in the worst case $O(k^2)$ chosen encryptions. For each ciphertexts an n-bit vector addition is needed. The storage of the matrix $G_{\mathcal{B}}$ requires $n(2k - n) = O(kn)$ bits of memory. This proves the following lemma:

Lemma 8.3.10 *Algorithm 8.3.9 terminates after at most k iterations and exists with $2k - n$ rows of the secret matrix G. The algorithm requires a work-factor W of $O(k^2n)$ bit operations and a memory-factor M of $O(kn)$ bits.*

Once $2n - k$ rows of the secret matrix G have been found, a secret-key cryptosystem equivalent to the LW scheme can be obtained as follows. Split the k-bit vector $\underline{x}$ into an $(n-k)$-bit subvector $\underline{x}_{\mathcal{A}} = \sigma_{\mathcal{A}}(\underline{x})$ and a $(2k-n)$-bit subvector $\underline{x}_{\mathcal{B}} = \sigma_{\mathcal{B}}(\underline{x})$, such that, the concatenation $(\underline{x}_{\mathcal{A}}||\underline{x}_{\mathcal{B}})$ is a coordinate permutation of the k-bit vector $\underline{x}$. Without loss of generality, it is assumed that the set $\mathcal{A}$ is the same as in Definition 8.3.1.

First, obtain an equivalent encryption procedure by using chosen encryptions to find ciphertexts of the form $\underline{y} = (\underline{x}_{\mathcal{A}}||\underline{0})G + f_{\mathcal{A}}(\underline{x}_{\mathcal{A}}||\underline{0})$. Define the set $\mathcal{T}$ as:

$$\mathcal{T} = \{(\underline{x}_{\mathcal{A}}, \underline{y}) \mid \underline{y} = \underline{x}G + f_{\mathcal{A}}(\underline{x}),\ \underline{x}_{\mathcal{B}} = \underline{0} \text{ and } \underline{x}_{\mathcal{A}} \in \mathrm{GF}(2)^{n-k}\}.$$

The subvector $\underline{x}_{\mathcal{A}}$ uniquely determines $f_{\mathcal{A}}(\underline{x})$ and, given $\underline{x}_{\mathcal{B}} = \underline{0}$, also $\underline{y}$. Rewrite the set $\mathcal{T}$ as a substitution table: $\underline{y} = \mathsf{T}(\underline{x}_{\mathcal{A}})$. Next, split the matrix G, in the encryption function of Definition 8.3.2, into two sub-matrices $G_{\mathcal{A}}$ and $G_{\mathcal{B}}$, such that

$$\underline{y} = \underline{x}G + f_{\mathcal{A}}(\underline{x}) = \sigma_{\mathcal{A}}(\underline{x})G_{\mathcal{A}} + \sigma_{\mathcal{B}}(\underline{x})G_{\mathcal{B}} + f_{\mathcal{A}}(\underline{x}),$$

where $G_{\mathcal{B}}$ equals the matrix as obtained in Algorithm 8.3.9. Replace $G_{\mathcal{A}}$ and the error function $f_{\mathcal{A}}(\underline{x})$ with the substitution table T:

$$\underline{y} = \sigma_{\mathcal{B}}(\underline{x})G_{\mathcal{B}} + \mathsf{T}(\sigma_{\mathcal{A}}(\underline{x})).$$

Second, obtain an equivalent decryption procedure by computing a parity-check matrix $H_{\mathcal{B}}$, such that $G_{\mathcal{B}}H_{\mathcal{B}}^T = O_{2k-n,2(n-k)}$. Next, define the set $\mathcal{T}^{-1}$ as:

$$\mathcal{T}^{-1} = \{(\underline{x}_{\mathcal{A}}, \underline{s}_{\mathcal{B}}) \mid \underline{s}_{\mathcal{B}} = \mathsf{T}(\underline{x}_{\mathcal{A}})H_{\mathcal{B}}^T,\ \underline{x}_{\mathcal{A}} \in \mathrm{GF}(2)^{n-k}\}.$$

It can be verified that $\underline{s}_{\mathcal{B}}$ uniquely determines $\underline{x}_{\mathcal{A}}$. Rewrite the set $\mathcal{T}^{-1}$ as a substitution table: $\underline{x}_{\mathcal{A}} = \mathsf{T}^{-1}(\underline{s}_{\mathcal{B}})$. These encryption and decryption procedures are summarized in the following secret-key cryptosystem which is equivalent to the LW scheme:

Definition 8.3.11 *Let $G_{\mathcal{B}}$ be a $(2n-k)\times n$ binary matrix with a right inverse $G_{\mathcal{B}}^{-R}$ (i.e., $G_{\mathcal{B}}G_{\mathcal{B}}^{-R} = I_{2k-n}$). Let $H_{\mathcal{B}}$ be a parity-check matrix $H_{\mathcal{B}}$, such that $G_{\mathcal{B}}H_{\mathcal{B}}^T = O_{2k-n,2(n-k)}$. Compute the set $\mathcal{T}$ as:*

$$\mathcal{T} = \{(\underline{x}_{\mathcal{A}}, \underline{y}) \mid \underline{y} = \underline{x}G + f_{\mathcal{A}}(\underline{x}),\ \underline{x}_{\mathcal{B}} = \underline{0} \text{ and } \underline{x}_{\mathcal{A}} \in GF(2)^{n-k}\}.$$

Rewrite the set $\mathcal{T}$ as a substitution table: $\underline{y} = \mathcal{T}(\underline{x}_{\mathcal{A}})$.

Encryption *A k-bit plaintext $\underline{x}$ is encrypted into an n-bit ciphertext $\underline{y}$ as*

$$\underline{y} = \sigma_{\mathcal{B}}(\underline{x})G_{\mathcal{B}} + \mathcal{T}(\sigma_{\mathcal{A}}(\underline{x})).$$

Decryption *A ciphertext $\underline{y}$ is decrypted into a plaintext $\underline{x}$ by computing $\underline{s}_{\mathcal{B}} = \underline{y}H_{\mathcal{B}}^T$, which uniquely determines $\underline{x}_{\mathcal{A}} = \mathcal{T}^{-1}(\underline{s}_{\mathcal{B}})$. Next, compute the subvector $\underline{x}_{\mathcal{B}} = (\underline{y} - \mathcal{T}(\underline{x}_{\mathcal{A}}))G_{\mathcal{B}}^{-R}$. Then, compute the plaintext $\underline{x}$ as $\underline{x} = \sigma_{\mathcal{A}}^{-1}(\underline{x}_{\mathcal{A}}) + \sigma_{\mathcal{B}}^{-1}(\underline{x}_{\mathcal{B}})$.*

Key *The secret key consists of: the matrix $G_{\mathcal{B}}$ ($H_{\mathcal{B}}$) and the table $\mathcal{T}$ $\mathcal{T}^{-1}$ and the sets $\mathcal{A}$ and $\mathcal{B}$.*

Algorithm 8.3.9 can also be successfully applied to the following extension of the LW scheme, where the encryption function is:

$$\underline{y} = \underline{x}G + \underline{z} \text{ with } \underline{z}H^T = e(\sigma_{\mathcal{A}}(\underline{x})),$$

and e: $\mathrm{GF}(2)^k \to \mathrm{GF}(2)^{n-k}$ is a nonlinear function. A logical extension is to use the nonlinear function e, such that $\underline{z}H^T = e(\underline{x})$. In this way the encryption scheme becomes

$$\underline{y}_i = \underline{x}_i G + \underline{z}_i, \text{ such that } \underline{z}_i H^T = e(\underline{x}_i).$$

However, an algorithm exists that obtains the secret matrix in time polynomially bounded in $|\mathcal{Z}|$. For example, Algorithm 8.2.13 can be used with the following observation:

If $\underline{u}_s = \underline{x}_i + \underline{x}_j$, then $\underline{e}_s = \underline{y}_i + \underline{y}_j = (\underline{x}_i G + \underline{z}_i) + (\underline{x}_j G + \underline{z}_j) = \underline{g}_s + \underline{z}_i + \underline{z}_j$.

So instead of encrypting the same plaintext $O(|\mathcal{Z}|^{1/2})$ times, now encrypt $O(|\mathcal{Z}|^{1/2})$ different plaintexts $\underline{x}_i$ and $\underline{x}_j$ for which it holds that

$\underline{x}_i + \underline{x}_j = \underline{u}_s$. It can be shown that with high probability, a $k \times n$ matrix equivalent to the encryption matrix G can be obtained within $O(k \times |\mathcal{Z}|^{1/2})$ encryptions.

If it is not used for authentication, Li and Wang suggested to use the set $\mathcal{Z}$ for encryption to improve the code rate of their scheme. However, their scheme is not based on code $\mathcal{C}$'s error-correcting capability to recover random errors, so that, the cryptanalyst obtains additional information about the used set of error vectors. Therefore, this is an ineffective modification of this type of secret-key cryptosystem.

Chapter 9

Beyond Majority Voting

In Chapter 8, Majority was introduced as a method to analyze the security of the secret-code encryption scheme. An extension of MV is *Local Majority Voting* (LMV) which simultaneously considers more than one coordinate. *Global Majority Voting* (GMV) goes one step further and aligns all of these local estimates of the subvector to obtain a global estimate of the vector. Finally, *Extended Majority Voting* (EMV) uses these estimates to recover the secret matrix. These four types of MV algorithms are discussed in the sections 9.2, 9.3, 9.4 and 9.5, respectively. Section 9.6 derives the number of coordinates that are considered in LMV in order to minimize the error probability. In conclusion, Section 9.7 describes a potential improvement to EMV.

9.1 Preliminaries

The set of error vectors $\mathcal{Z}$ used in Definition 9.1.2 is a subset of $\mathcal{Z}^a$, and is defined as follows:

Definition 9.1.1 *Without loss of generality, assume n to be an even integer. Given $a \in \{0, 1, \ldots, n/2\}$, the set $\mathcal{Z}^a$ is defined as:*

$$\mathcal{Z}^a = \{\underline{z} \in GF(2)^n \mid |w_H(\underline{z}) - \frac{n}{2}| \leq a\}. \tag{9.1}$$

This chapter is based on joint work with Joost Meijers [MT93].

Let $\mathcal{C}$ be a binary $[n,k]$-code with N cosets denoted by $\Gamma_1, \Gamma_2, \ldots, \Gamma_{\mathsf{N}}$. Define a subset Υ_j^a of each coset Γ_j as $\Gamma_j \cap \mathcal{Z}^a$. An EMV analysis is applied to the following locally-randomized cryptosystem:

Definition 9.1.2 *Let $\mathcal{C}$ be a binary linear $[n,k]$-code $\mathcal{C}$. Let G be a generator matrix of the code $\mathcal{C}$, and let H be a parity check matrix. Given the integer a from $\{0, 1, \ldots, n/2\}$, the set of error vectors $\mathcal{Z}$ is a subset of $GF(2)^n$, such that*

$$\mathcal{Z} = \{\underline{z} \in \Upsilon_j^a \mid 1 \leq j \leq \mathsf{N}\} \textit{ and } \forall_{j, 1 \leq j \leq \mathsf{N}} \, |\Upsilon_j^a \cap \mathcal{Z}| = 1.$$

Compute the syndrome-error table $\mathcal{T} = \{(\underline{z}H^T, \underline{z}) \mid \underline{z} \in \mathcal{Z}\}$.

Encryption *A k-bit plaintext $\underline{x}$ is encrypted into an n-bit ciphertext $\underline{y}$ as:*

$$\underline{y} = \underline{x}G + \underline{z},$$

where $\underline{z}$ is randomly chosen from $\mathcal{Z}$.

Decryption *A ciphertext $\underline{y}$ is decrypted into a plaintext $\underline{x}$ by computing $\underline{y}H^T$ to obtain $\underline{z}H^T$. The corresponding error vector $\underline{z}$ is obtained from the syndrome-error table $\mathcal{T}$. Next, the plaintext $\underline{x}$ is computed as $\underline{x} = (\underline{y} - \underline{z})G^{-R}$.*

Key *The secret key consists of: the matrix G (H) and the syndrome-error table T $(\mathcal{Z})$.*

For this cryptosystem, Section 9.2 shows that more than one condition exists for which an MV algorithm is not successful in obtaining the secret matrix G. In particular, when the n-bit error vectors $\underline{z}$ in $\mathcal{Z}$ satisfy

$$|\mathsf{w_H}(\underline{z}) - \frac{n}{2}| \leq a,$$

where a is an integer from $\{0, 1, \ldots, n/2\}$. Then, it is shown that the EMV algorithm, based on a simultaneous four-symbol MV analysis, does successfully recover the secret matrix G when the error vectors are not selected at random (i.e., $a = n/2$). As a result, the set of error vectors in the secret-key cryptosystems defined in Chapter 8 should be randomly chosen (i.e., without the weight constraint as in Definition 9.1.1).

9.2 Majority Voting

This section describes a MV analysis which considers only one coordinate of each vector for voting. This analysis is presented in a standardized way in order to facilitate generalization to more than one coordinate and to q-ary alphabets. This MV analysis is based on the following definition of a symbol counter, which is a special case of the vector counter defined in the next section.

Definition 9.2.1 (Symbol Counter) *Let z_i be the i-th coordinate of a vector $\underline{z} \in GF(2)^n$. For $1 \leq i \leq n$ and $x \in GF(2)$, the symbol counter is defined as*

$$SC_{i,x}(\underline{z})= \begin{cases} 0 & \text{if } x \neq z_i; \\ 1 & \text{if } x = z_i. \end{cases}$$

With little ambiguity in the notation, the symbol counter for the set $\mathcal{Z}$ is defined by

$$\mathsf{SC}_{i,x}(\mathcal{Z}) = \sum_{\underline{z} \in \mathcal{Z}} \mathsf{SC}_{i,x}(\underline{z}).$$

For all a and i, the value of the symbol counter $\mathsf{SC}_{i,0}(\mathcal{Z}^a)$ equals $\mathsf{SC}_{i,1}(\mathcal{Z}^a)$. It is of interest to examine the case $\mathsf{SC}_{i,0}(\mathcal{Z}) \neq \mathsf{SC}_{i,1}(\mathcal{Z})$ for $1 \leq i \leq n$ with respect to the Algorithm 8.1.2 (MV). Consider l realizations of the random vector variable $\underline{\boldsymbol{y}} = \underline{x}G + \underline{\boldsymbol{z}}$. For $\lambda_i = \mathsf{SC}_{i,0}(\mathcal{Z})/|\mathcal{Z}| < 1/2$, the (error) probability $\mathsf{Pe}(\lambda_i, l)$ that among the l realizations the number of 1s on the i-th coordinate does not exceed the number of 0s is given by (8.2). Majority voting yields an estimate for the i-th coordinate in the vector $\underline{x}G$. The probability of a correct estimate is:

$$\mathsf{Pc}(\lambda_i, l) = 1 - \mathsf{Pe}(\lambda_i, l) \geq 1 - \left[\sqrt{4\lambda_i(1-\lambda_i)}\right]^l .$$

Since $\lambda_i < 1/2$, it follows that $\mathsf{Pc}(\lambda_i, l) \to 1$ for $l \to \infty$. To obtain an estimate $\underline{x}E$ for $\underline{x}G$, repeat this for all n coordinates. Repeat the same procedure for k independent plaintexts $\underline{x}$. Let the rows of matrix X consist of the k plaintexts and the rows of XE consist of the k estimates. Then an estimate E for G follows from

$$E = X^{-1}(XE).$$

With probability $\prod_{i=1}^{n} \mathsf{Pc}(\lambda_i, l)^k$, the matrix E is a correct estimate for the matrix G.

The following theorem shows that for a sufficiently large number of cosets N (i.e, different error vectors), the symbol counter satisfies $\mathsf{SC}_{i,0}(\mathcal{Z}) \approx \mathsf{SC}_{i,1}(\mathcal{Z})$ for all a and i. In other words, with probability $\approx 1/2$ a MV on an arbitrary coordinate of the ciphertexts yields a correct estimate of the codeword (i.e., guessing may as well be without any strategy).

Theorem 9.2.2 *Let* $\Pr\{z_i^{(j)} = z\}$ *denote the probability that the* i*-th coordinate of* $\underline{z} \in \Upsilon_j^a$ *equals the value* $z \in GF(2)$. *Define* z_i *as the integer sum* $\sum_{j=1}^{N} z_i^{(j)}$. *For* $\epsilon > 0$,

$$\Pr\{|\frac{z_i}{N} - \frac{1}{2}| \geq \epsilon\} \leq \frac{1}{4N\epsilon^2}, \quad 1 \leq i \leq n.$$

Proof: For each j there exists exactly one (not necessarily different) k $(1 \leq k \leq m)$, such that $\Upsilon_j^a = \Upsilon_k^a + \underline{1}$. As a consequence,

$$\mathsf{Exp}(\frac{z_i^{\mathsf{N}}}{\mathsf{N}}) = \frac{1}{\mathsf{N}} \sum_{j=1}^{\mathsf{N}} \mathsf{Exp}(z_i^{(j)}) = \frac{1}{2}, \quad 1 \leq i \leq n.$$

For all i and j, the variance of $z_i^{(j)}$ can be upper-bounded by $\sigma^2 = 0.25$. Hence,

$$\mathsf{Var}(\frac{z_i}{\mathsf{N}}) = \sum_{j=1}^{\mathsf{N}} \frac{\mathsf{Var}(z_i^{(j)})}{\mathsf{N}^2} \leq \frac{\sigma^2}{\mathsf{N}} = \frac{1}{4\mathsf{N}}.$$

Application of Chebyshev's inequality proves this Theorem. □

Note that when $\underline{1} \in C$, then all sets Υ_j^a are balanced (i.e., for all j it holds that $\Upsilon_j^a = \Upsilon_j^a + \underline{1}$). In this case, for each set Υ_j^a and for all i it holds that $\mathsf{Exp}(z_i^{(j)}) = 0.5$ and $\mathsf{Var}(z_i^{(j)}) = 0.25$.

9.3 Local Majority Voting

In this section, the symbol counter is generalized to a vector counter which simultaneously considers more than one coordinate for voting.

Then a LMV strategy is developed that obtains useful estimates from subvectors of each row of the secret matrix G. In the following sections, it is shown how these estimates can be combined to obtain an estimate E for the matrix G.

Definition 9.3.1 (Vector Counter) *Let* $1 \leq s \leq n$. *Let* $\mathcal{A}$ *be an* s*-subset of* $\{1, 2, \ldots, n\}$ *(i.e.,* $|\mathcal{A}| = s$*). Let* $\underline{z}_{\mathcal{A}}$ *be the subvector of the* $\underline{z} \in GF(2)^n$ *that agrees with the coordinates of* $\mathcal{A}$. *For a given* $\mathcal{A}$ *and* $\underline{v}$*, the vector counter* $VC_{\mathcal{A},\underline{v}}(\underline{z})$ *is defined as:*

$$VC_{\mathcal{A},\underline{v}}(\underline{z}) = \begin{cases} 0 & \text{if } \underline{v} \neq \underline{z}_{\mathcal{A}}; \\ 1 & \text{if } \underline{v} = \underline{z}_{\mathcal{A}}. \end{cases}$$

With little ambiguity in the notation, the vector counter for the set $\mathcal{Z}$ is defined by

$$\mathsf{VC}_{\mathcal{A},\underline{v}}(\mathcal{Z}) = \sum_{\underline{z} \in \mathcal{Z}} \mathsf{VC}_{\mathcal{A},\underline{v}}(\underline{z}).$$

The set $\mathcal{A}$ extracts s coordinates of the vector $\underline{z}$ for which the vector counter $\mathsf{VC}_{\mathcal{A},\underline{v}}(\underline{z})$ looks for a match with the subvector $\underline{v}$.

Lemma 9.3.2 *For* $\mathcal{Z}^0$ *and all* $\underline{v} \in GF(2)^s$*, the probability* $P_{\mathcal{Z}^a_{\mathcal{A}}}(\underline{v})$ *that a realization of the random vector variable* $\underline{\boldsymbol{z}}^a_{\mathcal{A}}$ *equals* $\underline{v} \in GF(2)^s$ *is given by*

$$P_{\mathcal{Z}^0_{\mathcal{A}}}(\underline{v}) = \frac{VC_{\mathcal{A},\underline{v}}(\mathcal{Z}^0)}{|\mathcal{Z}^0|} = \binom{n - |\mathcal{A}|}{n/2 - w_H(\underline{v})} \bigg/ \binom{n}{n/2}. \tag{9.2}$$

Proof: The vector $\underline{v}$ can be extended to a vector of length n and weight $n/2$ by inserting a vector of length $n - s$ and weight $n/2 - \mathsf{w}_\mathsf{H}(\underline{v})$ at the appropriate coordinates. There are exactly $\binom{n}{n/2}$ vectors $\underline{z}$ in $\mathcal{Z}^0$ of weight $n/2$. □

For an n-bit vector $\underline{z}$ and an s-subset $\mathcal{A}$, the probability function $\mathsf{P}_{\mathcal{Z}^0_{\mathcal{A}}}(\underline{v})$ only depends on the weight of $\underline{v}$. If $\underline{v}$ and $\underline{w}$ are two s-bit vectors with $\mathsf{w}_\mathsf{H}(\underline{v}) = s - \mathsf{w}_\mathsf{H}(\underline{w})$, then $\mathsf{P}_{\mathcal{Z}^0_{\mathcal{A}}}(\underline{v})$ equals $\mathsf{P}_{\mathcal{Z}^0_{\mathcal{A}}}(\underline{w})$. Furthermore,

$$\mathsf{P}_{\mathcal{Z}^0_{\mathcal{A}}}(\underline{v}) < \mathsf{P}_{\mathcal{Z}^0_{\mathcal{A}}}(\underline{w}) \quad \begin{cases} \text{if } 0 \leq \mathsf{w}_\mathsf{H}(\underline{v}) < \mathsf{w}_\mathsf{H}(\underline{w}) \leq \lfloor s/2 \rfloor; \\ \textit{or} \text{ if } s \geq \mathsf{w}_\mathsf{H}(\underline{v}) > \mathsf{w}_\mathsf{H}(\underline{w}) \geq \lceil s/2 \rceil. \end{cases} \tag{9.3}$$

When the set $\mathcal{Z}^{n/2}$ is taken instead of the set $\mathcal{Z}^0$, then the probability $\mathsf{P}_{\mathcal{Z}_{\mathcal{A}}^{n/2}}(\underline{v})$ equals $\mathsf{P}_{\mathcal{Z}_{\mathcal{A}}^{n/2}}(\underline{w})$ for all s-bit vectors $\underline{v}$. Due to unequal weight distribution (9.3), the all-zero and all-one subvectors are expected to occur less frequently than any other subvector. This unequal weight distribution will be the basis of the following LMV analysis.

Let $\mathsf{Pe}_{\mathcal{A}}$ denote the (error) probability that a majority vote on the coordinates, which correspond to $\mathcal{A}$ of m realizations of the random vector variable $\underline{\boldsymbol{y}}_{\mathcal{A}} = (\underline{x}G + \underline{\boldsymbol{z}})_{\mathcal{A}}$ yields a vector other than the all-zero or all-one vector. Let $\mathcal{Z}_l$ be a set of l distinct error vectors, then the corresponding set of distinct ciphertexts of a plaintext $\underline{x}$ is defined as:

$$\mathcal{Y}_l^{(\underline{x})} = \{\underline{x}G + \underline{z} \mid \underline{z} \in \mathcal{Z}_l\}.$$

Note that the set $\mathcal{Y}_l^{(\underline{x})}$ is a translation of the set $\mathcal{Z}_l$ by the vector $\underline{x}G$, therefore

$$\mathsf{VC}_{\mathcal{A},(\underline{v}+(\underline{x}G)_{\mathcal{A}})}(\mathcal{Y}_l^{(\underline{x})}) = \mathsf{VC}_{\mathcal{A},\underline{v}}(\mathcal{Z}_l).$$

For sufficiently large l it is expected that $\mathsf{VC}_{\mathcal{A},\underline{v}}(\mathcal{Z}_l)$ will be minimal for either $\underline{v} = \underline{0}$ or $\underline{v} = \underline{1}$. As a consequence, with probability $1 - \mathsf{Pe}_{\mathcal{A}}$, the vector counter $\mathsf{VC}_{\mathcal{A},(\underline{v}+(\underline{x}G)_{\mathcal{A}})}(\mathcal{Y}_l^{(\underline{x})})$ will be minimal for $(\underline{x}G)_{\mathcal{A}}$ or $(\underline{x}G)_{\mathcal{A}} + \underline{1}$. Let $(\underline{x}E)_{\mathcal{A}}$ denote the vector $\underline{w}$ for which $\mathsf{VC}_{\mathcal{A},\underline{w}}(\mathcal{Y}_l^{(\underline{x})})$ is minimal.

Algorithm 9.3.3 (Local Majority Voting) *Consider the cryptosystem in Definition 9.1.2. Let r and s be integer, and let $\mathfrak{A}$ be a collection of r different subsets $\mathcal{A} = \{a_1, a_2, \ldots, a_s\}$ of $\{1, 2, \ldots, n\}$. Label these subsets as $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_r$. Then the collection $\mathfrak{A}$ determines r estimates $(\underline{x}E)_{\mathcal{A}_i}$ for $(\underline{x}G)_{\mathcal{A}_i}$ where $(1 \leq i \leq r)$.*

Step 1 *Select at random a k-bit plaintext and obtain a set $\mathcal{Y}_l^{(\underline{x})}$ of l distinct n-bit ciphertexts.*

Step 2 *For all $\mathcal{A} \in \mathfrak{A}$, set the estimate $(\underline{x}E)_{\mathcal{A}}$ for $(\underline{x}G)_{\mathcal{A}}$ equal to an s-bit vector $\underline{v}$ for which $VC_{\mathcal{A},\underline{v}}(\mathcal{Y}_l^{(\underline{x})})$ is minimal. Then,*

$$(\underline{x}E)_{\mathcal{A}} \in \left\{\underline{v} \in GF(2)^s \mid \forall_{\underline{w} \in GF(2)^s} \left[VC_{\mathcal{A},\underline{v}}(\mathcal{Y}_l^{(\underline{x})}) \leq VC_{\mathcal{A},\underline{w}}(\mathcal{Y}_l^{(\underline{x})})\right]\right\}.$$

For a large enough l, with overwhelming probability, one of the following two possibilities occurs:

$$(\underline{x}E)_{\mathcal{A}}= \left\{ \begin{array}{l} (\underline{x}G)_{\mathcal{A}}; \\ or\ (\underline{x}G)_{\mathcal{A}} + \underline{1}. \end{array} \right. \tag{9.4}$$

The Algorithm 9.3.3 (LMV) finds r estimates $(\underline{x}E)_{\mathcal{A}_i}$ for $(\underline{x}G)_{\mathcal{A}_i}$ from the l observations of the random vector variable $(\underline{x}G+\boldsymbol{z})_{\mathcal{A}_i}$. These estimates are with overwhelming probability equal to $(\underline{x}G)_{\mathcal{A}_i}$ or $(\underline{x}G)_{\mathcal{A}_i}+\underline{1}$.

Remark 9.3.4 *Since all s-bit vectors $\underline{v}$ can be considered as estimates, let the best estimate for $(\underline{x}G)_{\mathcal{A}}$ be est_1 and set it equal to $\min_{\underline{v}} VC_{\mathcal{A},\underline{v}}(\mathcal{Y}_l^{(\underline{x})})$. Next, label in a similar way all s-bit vectors $\underline{v}$ as candidate estimates, such that*

$$VC_{\mathcal{A},est_1}(\mathcal{Y}_l^{(\underline{x})}) \leq VC_{\mathcal{A},est_2}(\mathcal{Y}_l^{(\underline{x})}) \leq \ldots \leq VC_{\mathcal{A},est_{2^s}}(\mathcal{Y}_l^{(\underline{x})}).$$

As observed, the expected result is

$$est_1 = est_2 + \underline{1}. \tag{9.5}$$

If this is the case, then accept est_1 as a correct estimate. If (9.5) does not hold, then a decision not to trust est_1 may be appropriate and the estimate est_3 might also be considered.

9.4 Global Majority Voting

This section describes how to choose the collection $\mathfrak{A}$ of subsets $\mathcal{A}_i$ in Algorithm 9.3.3 (LMV). Choose two integers r and s so that the subsets $\mathcal{A}_i \in \mathfrak{A}$ satisfy

$$\left\{ \bigcup_{1\leq i\leq r} \mathcal{A}_i \right\} = \{1,2,\ldots,n\} \text{ and} \tag{9.6}$$

$$\forall_{i,\ 1\leq i<r}\ \left\{\mathcal{A}_i \bigcap \mathcal{A}_{i+1}\right\} \neq \emptyset. \tag{9.7}$$

When (9.6) is satisfied, these r subvectors $\underline{z}_{\mathcal{A}_i}$ cover a vector of length n. By satisfying (9.7), each subvector $\underline{z}_{\mathcal{A}_i}$ has some coordinates in common (on which their values might differ) with at least one other subvector

$\underline{z}_{\mathcal{A}_j}$. Consequently, $n \leq r(s-1)+1$. note by choosing $\mathfrak{A}$, such that $\mathcal{A}_i$ and $\mathcal{A}_{i+1}$ share more than one coordinate, it is possible to detect incorrect estimates.

The following GMV algorithm aligns the r estimates $(\underline{x}E)_{\mathcal{A}_i}$ such that they overlap the coordinate and cover the complete vector $\underline{x}G$. The GMV algorithm provides an estimate $\underline{x}E$ for the row $\underline{x}G$ which might differ by the all-one vector.

Algorithm 9.4.1 (Global Majority Voting) *Let r, s and $\mathfrak{A}$ be as defined in (9.6) and (9.7). Let $(\underline{x}E)_{\mathcal{A}_1}$, $(\underline{x}E)_{\mathcal{A}_1}$, $\ldots$, $(\underline{x}E)_{\mathcal{A}_r}$ be the obtained estimates in Algorithm 9.3.3 (LMV). Then, the following two steps provide an estimate $\underline{x}E$ for $\underline{x}G$ or $\underline{x}G+\underline{1}$.*

Step 1 *Set the s bits on the coordinates $\mathcal{A}_1$ of the vector $\underline{x}E$ equal to the estimate $(\underline{x}E)_{\mathcal{A}_1}$. Let $i=2$.*

Step 2 *The subvectors $(\underline{x}E)_{\mathcal{A}_{i-1}}$ and $(\underline{x}E)_{\mathcal{A}_i}$ must have at least one coordinate in common. When the values in the overlapping coordinates disagree, add the all-one subvector to the subvector $(\underline{x}E)_{\mathcal{A}_i}$. Set the bits on the coordinates $\mathcal{A}_i$ of the vector $\underline{x}E$ equal to the (updated) estimate $(\underline{x}E)_{\mathcal{A}_i}$.*

Step 3 *If $i<r$, then let $i=i+1$ and return to Step 2, otherwise, exit with the estimate $\underline{x}E$ for $\underline{x}G$.*

With overwhelming probability, the algorithm exits in Step 3 with either

$$\underline{x}E = \begin{cases} \underline{x}G; \\ \text{or } \underline{x}G+\underline{1}. \end{cases}$$

9.5 Extended Majority Voting

Let $\underline{x}_\nu$ be a k-bit plaintext equal to $\underline{x}+\underline{u}_\nu$ where $\underline{u}_\nu$ is the ν-th unit vector. First, the following algorithm obtains k vectors $\underline{x}_\nu E$ from which an estimate for the ν^{th}-row of G is computed (i.e., $\underline{e}_\nu = \underline{x}E + \underline{x}_\nu E$). The estimates are obtained using the Algorithm 9.4.1 (GMV). Since it is not known whether or not the estimates obtained in this way are disturbed with the all-one vector, the obtained estimate of the secret matrix G is of the form

$$E = G + A,$$

where A is a $k \times n$ binary matrix, and where the set of all rows of A consists only of all-one vectors and all-zero vectors. Second, depending on the structure of $\mathcal{Z}$, the matrix A can be removed by either defining an equivalent cryptosystem or by removing the all-one vectors.

Algorithm 9.5.1 (Extended Majority Voting) *Consider the cryptosystem in Definition 9.1.2 and the Algorithm 9.4.1 (GMV).*

Step 1 *By using Algorithm 9.4.1, obtain for each plaintext $\underline{x}$ and for $\underline{x}_\nu = \underline{x} + \underline{u}_\nu$ $(1 \leq \nu \leq k)$ a GMV estimate $\underline{x}_\nu E$.*

Step 2 *Compute the row estimate $\underline{e}_\nu$ of the ν-th row of the secret matrix G as:*

$$\underline{e}_\nu = \underline{x}E + \underline{x}_\nu E, \qquad 1 \leq \nu \leq k.$$

Let $E = (\underline{e}_1^T, \underline{e}_2^T, \cdots, \underline{e}_k^T)^T$ be an estimate for the secret matrix G.

Step 3 *Let $\mathcal{Z}_l = \underline{x}E + \mathcal{Y}^{(\underline{x})_l}$.*

When $\mathcal{Z} + \underline{1} = \mathcal{Z}$, then the matrix E and the set $\mathcal{Z}$ together define an equivalent cryptosystem. Otherwise, compare the set $\mathcal{Y}_l^{(\underline{x}_\nu)} + \underline{e}_\nu$ with the set $\mathcal{Y}_l^{(\underline{x})}$. If the two sets are different, let $\underline{e}_\nu = \underline{e}_\nu + \underline{1}$. To obtain E, repeat this for all row estimates $\underline{e}_\nu$ $(1 \leq \nu \leq k)$. Furthermore, let

$$\mathcal{Z}_l = \{\underline{y} + \underline{x}E \mid \underline{y} \in \mathcal{Y}_l^{(\underline{x})}\}$$

be the set of corresponding error vectors. The matrix E and the set $\mathcal{Z}$ together define an equivalent cryptosystem.

The GMV algorithm obtains with overwhelming probability the following estimate

$$\underline{x}_\nu E = \begin{cases} \underline{x}_\nu G; \\ or\ \underline{x}_\nu G + \underline{1}, \quad 1 \leq \nu \leq k. \end{cases}$$

Therefore, the matrix $E = (\underline{e}_1^T, \underline{e}_2^T, \cdots, \underline{e}_k^T)^T$, as obtained in Step 2, equals with high probability the matrix $G + A$. So that depending on the structure of $\mathcal{Z}$, the matrix A can be removed by either defining an equivalent cryptosystem or by removing the all-one vectors in Step 3. It suffices to use l ($l \ll \mathsf{N}$) ciphertexts for Algorithm 9.5.1 (EMV).

9.6 Error Probability

This section proves an upper-bound on the error probability $\mathsf{Pe}_{\mathcal{A}}$. Then, this upper-bound is minimized by determining the number of coordinates in a LMV. Finally, the upper-bound on the overall error probability for Algorithm 9.5.1 (EMV) is given. An generalization of Theorem 9.2.2 is the following theorem with a similar proof:

Theorem 9.6.1 *Let $1 \leq s \leq n$, $\underline{v} \in GF(2)^s$ and $|\mathcal{A}| = s$. Let $\Pr\{\underline{z}_{\mathcal{A}}^{(j)} = \underline{v}\}$ denote the probability that the subvector $\underline{z}_{\mathcal{A}}$ of $\underline{z} \in \Upsilon_j^a$ equals the vector $\underline{v} \in GF(2)^s$. If $\underline{v} = \underline{z}_{\mathcal{A}}$, then let $N_{\mathcal{A}}^{(j)}(\underline{v})$ equal 1, otherwise, let it equal 0. For all $\underline{v} \in GF(2)^s$, define $N_{\mathcal{A}}(\underline{v})$ as the integer sum $\sum_{j=1}^{N} N_{\mathcal{A}}^{(j)}(\underline{v})$. For $\epsilon > 0$,*

$$\Pr\{|\frac{N_{\mathcal{A}}(\underline{v})}{N} - \sum_{j=1}^{N} \frac{VC_{\mathcal{A},\underline{v}}(\Upsilon_j^a)}{N|\Upsilon_j^a|}| \geq \epsilon\} \leq \frac{1}{4N\epsilon^2}. \tag{9.8}$$

The probability that the outcome of the random variable $\mathsf{N}_{\mathcal{A}}(\underline{v})/\mathsf{N}$ differs by more than $\epsilon > 0$ from its expectation is upper-bounded. This expectation depends on the structure of Υ_j^a. Without loss of generality, let the N different sets Υ_i^a be ordered, such that

$$|\Upsilon_1^a| \leq |\Upsilon_2^a| \leq \ldots \leq |\Upsilon_{\mathsf{N}}^a|.$$

Suppose among the N sets Υ_i^a there are m_1 with cardinality $|\Upsilon_1^a|$, m_2 with cardinality $|\Upsilon_{m_1+1}^a|$, and so forth. Let $m_j = \sum_{i=1}^{j} m_i$, then, for sufficiently large N, it follows:

$$\sum_{j=1}^{\mathsf{N}} \frac{\mathsf{VC}_{\mathcal{A},\underline{v}}(\Upsilon_j^a)}{\mathsf{N}|\Upsilon_j^a|} = \frac{1}{\mathsf{N}} \sum_{j=1}^{k} |\Upsilon_{m_j}^a|^{-1} \sum_{i=m_{j-1}+1}^{m_j} \mathsf{VC}_{\mathcal{A},\underline{v}}(\Upsilon_i^a) \simeq$$

$$\frac{1}{\mathsf{N}}\sum_{j=1}^{k}|\Upsilon^a_{m_j}|^{-1}\left[m_j\frac{\mathsf{VC}_{\mathcal{A},\underline{v}}(\mathcal{Z}^a)}{|\mathcal{Z}^a|}|\Upsilon^a_{m_j}|\right]=\frac{\mathsf{VC}_{\mathcal{A},\underline{v}}(\mathcal{Z}^a)}{|\mathcal{Z}^a|}.$$

Equality holds if $|\Upsilon^a_1| = |\Upsilon^a_2| = \ldots = |\Upsilon^a_{\mathsf{N}}|$. An interpretation of this results is: for large N, the outcome of $\mathsf{N}_{\mathcal{A}}(\underline{v})/\mathsf{N}$ is close to $\mathsf{VC}_{\mathcal{A},\underline{v}}(\mathcal{Z}^a)/|\mathcal{Z}^a|$ and depends solely on s (n and a are fixed).

Corollary 9.6.2 *Let $\mathcal{Z}^{n/2}$, so that, $|\Upsilon^{n/2}_1| = |\Upsilon^{n/2}_2| = \ldots = |\Upsilon^{n/2}_{N}|$. Then, for $\epsilon > 0$, $1 \leq s \leq n$, $|\mathcal{A}| = s$ and $\underline{v} \in GF(2)^s$, Equation (9.8) reduces to*

$$\Pr\{|\frac{N_{\mathcal{A}}(\underline{v})}{N} - \frac{1}{2^s}| \geq \epsilon\} \leq \frac{1}{4N\epsilon^2}.$$

From this corollary, it follows that all subvectors $\underline{v}$ are equally likely to appear in N realizations of $\boldsymbol{z}$. For this case, it may be concluded that an LMV algorithm will not succeed in obtaining the secret matrix G.

Due to the unequal weight distribution (9.3), the all-zero and all-one subvectors are expected to occur less frequently than any other subvector. Let $\mathsf{Pe}_{\mathcal{A}}$ denote the (error) probability that a majority vote, on the $\mathcal{A}$ coordinates of N realizations $\underline{\boldsymbol{y}}_{\mathcal{A}} = (\underline{x}G + \underline{z})_{\mathcal{A}}$, yields a vector other than the all-zero or all-one vector. Then,

$$\mathsf{Pe}_{\mathcal{A}} \leq \Pr\left\{\min(\mathsf{N}_{\mathcal{A}}(\underline{0}), \mathsf{N}_{\mathcal{A}}(\underline{1})) \geq \min_{\underline{v}\in\mathrm{GF}(2)^s/\{\underline{0},\underline{1}\}} \mathsf{N}_{\mathcal{A}}(\underline{v})\right\}.$$

Since the error probability $\mathsf{Pe}_{\mathcal{A}}$ is a non-decreasing function of a, two extreme cases are considered. Therefore, the following distinction is made: Either $\mathcal{Z}$ is randomly selected from $\mathcal{Z}^0$, where all vectors $\underline{v} \in \mathrm{GF}(s)^s$ satisfy (9.3), so that $\mathsf{Pe}_{\mathcal{A}} \to 0$ for sufficiently large N, or $\mathcal{Z}$ is randomly selected from $\mathcal{Z}^{n/2}$, where all vectors $\underline{v} \in \mathrm{GF}(2)^s$ have equal probability (Corollary 9.6.2). Consequently, $\mathsf{Pe}_{\mathcal{A}} \to 1 - 2^{s-1}$ ($s > 1$) for sufficiently large N.

The following heuristics are used to obtain an upper-bound on $\mathsf{Pe}_{\mathcal{A}}$. For all $\underline{u}, \underline{v} \in \mathrm{GF}(2)^s$ with $0 < \mathsf{w}_{\mathsf{H}}(\underline{v}) < s$, let:

$$\begin{aligned}\lambda_{\mathcal{A}}(\underline{u},\underline{v}) &= \frac{\mathsf{P}_{\mathcal{Z}_{\mathcal{A}}}(\underline{u})}{\mathsf{P}_{\mathcal{Z}_{\mathcal{A}}}(\underline{u}) + \mathsf{P}_{\mathcal{Z}_{\mathcal{A}}}(\underline{v})}\\ &\simeq \frac{\mathsf{VC}_{\mathcal{A},\underline{u}}(\mathcal{Z}^a)}{\mathsf{VC}_{\mathcal{A},\underline{u}}(\mathcal{Z}^a) + \mathsf{VC}_{\mathcal{A},\underline{v}}(\mathcal{Z}^a)}\end{aligned}$$

and

$$\begin{aligned} l_{\mathcal{A}}(\underline{u},\underline{v}) &= [\mathsf{P}_{\mathcal{Z}_{\mathcal{A}}}(\underline{u}) + \mathsf{P}_{\mathcal{Z}_{\mathcal{A}}}(\underline{v})]l \\ &\simeq [\mathsf{VC}_{\mathcal{A},\underline{u}}(\mathcal{Z}^a) + \mathsf{VC}_{\mathcal{A},\underline{v}}(\mathcal{Z}^a)]\frac{l}{|\mathcal{Z}^a|}. \end{aligned}$$

The average error probability $\mathsf{Pe}_{\mathcal{A}}(\underline{v})$ that subvector $\underline{v}$ appears less frequently than $\underline{0}$ or $\underline{1}$ is given by:

$$\begin{aligned} \mathsf{Pe}_{\mathcal{A}}(\underline{v}) &= \mathsf{Pe}(\lambda_{\mathcal{A}}(\underline{0},\underline{v}), l_{\mathcal{A}}(\underline{0},\underline{v})) \\ &= \mathsf{Pe}(\lambda_{\mathcal{A}}(\underline{1},\underline{v}), l_{\mathcal{A}}(\underline{1},\underline{v})) \quad \text{with} \quad 0 < \mathsf{w_H}(\underline{v}) < s. \end{aligned} \tag{9.9}$$

The overall average error probability $\mathsf{Pe}_{\mathcal{A}}$ can be upper-bounded as:

$$\mathsf{Pe}_{\mathcal{A}} < \sum_{0<\mathsf{w_H}(\underline{v})<s} \mathsf{Pe}_{\mathcal{A}}{}^2(\underline{v}). \tag{9.10}$$

Let $a = 0$, and define $\lambda_i = \lambda_{\mathcal{A}}(\underline{0},\underline{v})$ and $l_i = l_{\mathcal{A}}(\underline{0},\underline{v})$, where $0 < i = \mathsf{w_H}(\underline{v}) < s$. Next, substitute (8.2) and (9.9) into (9.10) to obtain:

$$\mathsf{Pe}_{\mathcal{A}} < \sum_{i=1}^{s-1} \binom{s}{i} \mathsf{Pe}^2(\lambda_i, l_i) \leq \sum_{i=1}^{s-1} \binom{s}{i} [4\lambda_i(1-\lambda_i)]^{l_i}. \tag{9.11}$$

As a result, the error probability that the vector counter $\mathsf{VC}_{\mathcal{A},\underline{v}}(\mathcal{Z})$ is not minimal for some vector $\underline{v}$ ($0 < \mathsf{w_H}(\underline{v}) < s$) is upper-bounded by (9.11). Due to the fixed underlying code parameters n and k, a cryptanalyst is only free to choose s. If $\mathsf{Pe}_{\mathcal{A}}$ is minimized according to the upper-bound (9.11), then for all interesting cases the optimal value $s = 4$ is found. The error probability Pe for Algorithm 9.5.1 (EMV) is upper-bounded by

$$\mathsf{Pe} < 1 - (1 - \mathsf{Pe}_{\mathcal{A}})^{r(k+1)} < r(k+1)\mathsf{Pe}_{\mathcal{A}}.$$

9.7 Improvements

Depending on the structure of the set of error vectors, the performance of the Algorithm 9.5.1 (EMV) can be improved. Instead of focussing on the value of the estimate $(\underline{x}E)_{\mathcal{A}_i}$, search for a subvector where the vector count is different from all other estimates, so that

$$\forall_{\underline{v}\in \mathrm{GF}(2)^s\setminus(\underline{x}E)_{\mathcal{A}_i}} \quad \mathsf{VC}_{\mathcal{A}_i,(\underline{x}E)_{\mathcal{A}_i}}(\mathcal{Y}^{(\underline{x})}) \neq \mathsf{VC}_{\mathcal{A}_i,\underline{v}}(\mathcal{Y}^{(\underline{x})}).$$

Next, find a corresponding estimate in $\mathcal{Y}^{(\underline{x}_\nu)}$, for example $(\underline{x}_\nu E)_{\mathcal{A}_i}$, for which

$$\mathsf{VC}_{\mathcal{A}_i,(\underline{x}_\nu E)_{\mathcal{A}_i}}(\mathcal{Y}^{(\underline{x}_\nu)}) = \mathsf{VC}_{\mathcal{A}_i,(\underline{x}E)_{\mathcal{A}_i}}(\mathcal{Y}^{(\underline{x})}).$$

For the secret matrix G the s coordinates $\mathcal{A}_i$ in the ν-th row follow from

$$(\underline{e}_\nu)_{\mathcal{A}_i} = (\underline{x}E)_{\mathcal{A}_i} + (\underline{x}E)_{\mathcal{A}_i}.$$

In this way, obtain the subvectors $(\underline{e}_\nu)_{\mathcal{A}_i}$ for $1 \leq \nu \leq k$. Repeat this for all $\mathcal{A}_i$ $(1 \leq i \leq r = n/s)$ to find $\underline{e}_\nu$ $(1 \leq \nu \leq k)$. Because this procedure yields a correct matrix G, overlapping coordinates are not required.

Appendix A

Swap Computations

A.1 Number of Swaps

Lemma A.1.1 *Let* $a = \min\{t, k\}$, *and let* N_l *be as defined in 5.9. Then*

$$N_l > N_{l+1} > 0, \quad 1 \leq l \leq a - 1.$$

Proof: From (5.5)–(5.7) it follows that

$$\mathsf{N}_a = \frac{k(n-k)}{a(n-k-t+a)} = \frac{k(n-k)}{an-kt} > 0.$$

For $a > 1$, it is proved by induction on l that $\mathsf{N}_l > \mathsf{N}_{l+1}$ holds for $1 \leq l \leq a-1$. Using the induction assumption $\mathsf{N}_l > \mathsf{N}_{l+1}$ in (5.9) yields

$$\mathsf{N}_l < \frac{k(n-k) + (k-l)(t-l)\mathsf{N}_l}{l(n-k-t+l)} \Rightarrow (ln - kt)\mathsf{N}_l < k(n-k).$$

For N_{l-1} it follows that

$$\begin{aligned}
\mathsf{N}_{l-1} &= \frac{k(n-k) + (k-l+1)(t-l+1)\mathsf{N}_l}{(l-1)(n-k-t+l-1)} \\
&> \frac{(ln-kt) + (k-l+1)(t-l+1)}{(l-1)(n-k-t+l-1)}\mathsf{N}_l \\
&= \frac{ln + (l-1)(l-1-k-t)}{(l-1)n + (l-1)(l-1-k-t)}\mathsf{N}_l > \mathsf{N}_l,
\end{aligned}$$

This appendix is based on joint work with Raymond Doyen [DT94b].

which proves the lemma. □

Lemma A.1.2 *For sufficiently large n:*

$$N_1 = O\left(n\sqrt{n}\right) q^{n\left(\widehat{H}_q(\frac{t}{n}) - (1-R)\widehat{H}_q\left(\frac{t}{n(1-R)}\right)\right)}. \tag{A.1}$$

Proof: To obtain an expression for N_1 in N_a, with $a = \min(k,t)$, rewrite (5.9) as

$$\mathsf{N}_l = u_{l+1} + v_{l+1}\mathsf{N}_{l+1}, \quad 1 \leq l \leq a-1, \tag{A.2}$$

with

$$u_{l+1} = \frac{1}{\Pr(l-1 \mid l)} = \frac{k(n-k)}{l(n-k-t+l)} > 0,$$

and

$$v_{l+1} = \frac{\Pr(l+1 \mid l)}{\Pr(l-1 \mid l)} = \frac{(k-l)(t-l)}{l(n-k-t+l)} < v_l.$$

Apply (A.2) recursively, from 1 to a, to obtain

$$\mathsf{N}_1 = \mathsf{N}_a \prod_{i=1}^{a-1} v_{i+1} + \sum_{j=1}^{a-1}\left(\prod_{i=1}^{j-1} v_{i+1}\right) u_{j+1}. \tag{A.3}$$

The value j, $1 \leq j \leq a-1$, that maximizes

$$\prod_{i=1}^{j-1} v_{i+1}$$

equals $j = \lfloor kt/n \rfloor < \min(t,k)$ and therefore exists. Define $h := \lfloor kt/n \rfloor$, and note that $v_{h+1} \geq 1$, $v_{h+2} < 1$ and $u_{j+1} = O(n)$, for every $0 < j < a$. Therefore, (A.3) becomes

$$\begin{aligned}
\mathsf{N}_1 &= O(n)\left(\prod_{i=1}^{h} v_{i+1}\right) + \sum_{j=1}^{a-1}\left(\prod_{i=1}^{h} v_{i+1}\right) O(n) && \text{(A.4)}\\
&= \left(\prod_{i=1}^{h} v_{i+1}\right) O(n^2). && \text{(A.5)}
\end{aligned}$$

To show that the complexity of the SPD algorithm is asymptotically equal to N_1, proceed as follows:

$$\begin{aligned}\prod_{i=1}^{h} v_{i+1} &= \prod_{i=1}^{h} \frac{(k-i)(t-i)}{i(n-k-t+i)} \\ &= \frac{(k-1)!\,(t-1)!\,(n-k-t)!}{(k-h-1)!\,(t-h-1)!\,h!\,(n-k-t+h)!}.\end{aligned}$$

Since $k = \mathrm{R}n$ and $t = \lambda n$, for $n \to \infty$ substitute Stirling's formula [Fel57] (i.e., $m! = e^{-m} m^m \sqrt{2\pi m}(1+o(1))$ when $m \to \infty$) to yield

$$\prod_{i=1}^{h} v_{i+1} = \frac{k^k t^t (n-k-t)^{n-k-t} \times n^{-\frac{1}{2}} O(1)(1+o(1))}{(k-h)^{k-h}(t-h)^{t-h} h^h (n-k-t+h)^{n-k-t+h}}$$

$$= \frac{(\frac{k}{n})^k (\frac{t}{n})^t (1-\frac{k}{n}-\frac{t}{n})^{n-k-t} \times O(n^{-\frac{1}{2}})}{(\frac{k}{n}-\frac{h}{n})^{k-h}(\frac{t}{n}-\frac{h}{n})^{t-h}(\frac{h}{n})^h(1-\frac{k}{n}-\frac{t}{n}+\frac{h}{n})^{n-k-t+h}}. \qquad \text{(A.6)}$$

The following expression for N_1 follows from substituting (A.6) into (A.4):

$$\frac{\mathrm{R}^{n\mathrm{R}} \lambda^{n\lambda} (1-\mathrm{R}-\lambda)^{n(1-\mathrm{R}-\lambda)} \times O(n\sqrt{n})}{(\mathrm{R}-\mathrm{R}\lambda)^{n(\mathrm{R}-\mathrm{R}\lambda)}(\lambda-\mathrm{R}\lambda)^{n(\lambda-\mathrm{R}\lambda)}(\mathrm{R}\lambda)^{n\mathrm{R}\lambda}(1-\mathrm{R}-\lambda+\mathrm{R}\lambda)^{n(1-\mathrm{R}-\lambda+\mathrm{R}\lambda)}},$$

where $\mathrm{R} = k/n$, $\lambda = t/n$ and $\mathrm{R}\lambda = h/n + o(1)$. Using the entropy function notation, this equals

$$\mathsf{N}_1 = O\left(n\sqrt{n}\right) q^{n\left(\widehat{\mathrm{H}}_q(\lambda)-(1-\mathrm{R})\widehat{\mathrm{H}}_q\left(\frac{\lambda}{1-\mathrm{R}}\right)\right)},$$

and proves the lemma. □

Lemma A.1.3 *For all integers n, k and t satisfying $1 \le k \le n$ and $1 \le t \le n-k$, it holds that*

$$\sum_{j=1}^{\min(t,k)} \frac{\binom{k}{j}\binom{n-k}{t-j}}{\binom{n}{t}} \to 1, \qquad (n \to \infty).$$

Proof: From elementary probability theory [Fel57], it follows that

$$\sum_{j=0}^{\min(t,k)} \frac{\binom{k}{j}\binom{n-k}{t-j}}{\binom{n}{t}} = \frac{\binom{n-k}{t}}{\binom{n}{t}} + \sum_{j=1}^{\min(t,k)} \frac{\binom{k}{j}\binom{n-k}{t-j}}{\binom{n}{t}} = 1.$$

Note that

$$\begin{aligned}\frac{\binom{n-k}{t}}{\binom{n}{t}} &= \frac{2^{n(1-\mathrm{R})\mathrm{H}_2\left(\frac{t}{n(1-\mathrm{R})}\right)+o(n)}}{2^{n\mathrm{H}_2\left(\frac{t}{n}\right)+o(n)}} \\ &= 2^{n\left((1-\mathrm{R})\mathrm{H}_2\left(\frac{t}{n(1-\mathrm{R})}\right)-\mathrm{H}_2\left(\frac{t}{n}\right)\right)+o(n)} \to 0, \qquad (n \to \infty),\end{aligned}$$

which follows from the $\cap$-convexity property of the entropy function, and proves the lemma. □

A.2 Rank Transition Probabilities

Lemma A.2.1 *Let $u > 0$, and let $G = (K \mid R)$ be a random $k \times n$ matrix in $(k-j)$-KR-form where $0 \leq j \leq \min\{k-1, u-1\}$. For arbitrary, but fixed k, the rank-transition probabilities satisfy:*

- Decrement ($0 \leq j \leq \min\{k-2, u-1\}$):

$$q_{k-j}^{k-j-1} = \frac{k-j}{k}\left(\frac{1}{2}\right)^{2j+1}, \tag{A.7}$$

- No change ($0 \leq j \leq \min\{k-1, u-1\}$):

$$q_{k-j}^{k-j} = \frac{k-j}{k}\left(\left(\frac{1}{2}\right)^{j-1} - 3\left(\frac{1}{2}\right)^{2j+1}\right) + \frac{j}{k}\left(\frac{1}{2}\right)^{j}, \tag{A.8}$$

- Increment ($1 \leq j \leq \min\{k-1, u-1\}$):

$$q_{k-j}^{k-j+1} = \frac{k-j}{k}\left(1 - \left(\frac{1}{2}\right)^{j}\right)^{2} + \frac{j}{k}\left(1 - \left(\frac{1}{2}\right)^{j}\right). \tag{A.9}$$

Proof: Suppose,

$$G = (K \mid R) = \left(\begin{array}{c|c} \begin{array}{c|c} I_{k-j} & A \\ \hline O_{j,k-j} & O_{j,j} \end{array} & R \end{array} \right).$$

After one swap the rank of the K matrix either decreases, remains the same (unchanged) or increases. In the continuation of this proof, each situation is considered separately.

Decreasing Rank Without loss of generality, suppose that column $\underline{e}_{k-j}^T$ from the first $k-j$ columns of K is swapped for column $(\underline{r}_1 \mid \underline{0})^T$ from the R-part, where $\underline{r}_1$ is a fixed $(k-j-1)$-vector and $\underline{e}_{k-j}$ is the $(k-j)$-th unit vector. Then the matrix GP is

$$GP = \left(\begin{array}{c|c|c} \begin{array}{c|c|c} I_{k-j-1} & A^- & \underline{r}_1^T \\ \hline O_{j+1,k-j-1} & \multicolumn{2}{c}{O_{j+1,j+1}} \end{array} & \underline{e}_{k-j}^T & R^- \end{array} \right),$$

where A^- is obtained from A by omitting an all-zero row and R^- follows from R by omitting the column $(\underline{r}_1 \mid \underline{0})^T$. The first k columns of GP form a submatrix of rank $k-j-1$. The $O_{j,j}$-part of G is extended to the $O_{j+1,j+1}$-part in GP. This means that there are $2j+1$ extra zero-elements, where j zeros are obtained from A and $j+1$ from R. As a result, the probability of a decreasing rank is equal to (A.7).

Unchanged Rank Two possibilities exist:

1. As in the situation of decreasing rank, suppose that column $\underline{e}_{k-j}^T$ from the first $k-j$ columns of the K-part is swapped for column $(\underline{r}_1 \mid \underline{r}_2)^T$ from the R-part, where $\underline{r}_2 \in \mathrm{GF}(2)^{j+1}$ is *not* the all-zero vector. Then GP satisfies

$$GP = \left(\begin{array}{c|c|c} \begin{array}{c|c|c} I_{k-j-1} & A^- & \underline{r}_1^T \\ \hline O_{j+1,k-j-1} & \begin{array}{c} \underline{a} \\ \hline O_{j,j} \end{array} & \underline{r}_2^T \end{array} & \underline{e}_{k-j}^T & R^- \end{array} \right),$$

where A^- and $\underline{a}$ together form A. Consider the matrix

$$D := \left(\begin{array}{c|c} \begin{array}{c} \underline{a} \\ \hline O_{j,j} \end{array} & \underline{r}_2^T \end{array} \right).$$

Note that D corresponds to the matrix $O_{j+1,j+1}$ in the situation of decreasing rank. If the rank of D is one, then the rank of GP remains $k-j$. For random vectors $\underline{a} \in \mathrm{GF}(2)^j$ and $\underline{r}_2 \in \mathrm{GF}(2)^{j+1}$:

$$\Pr\{rank(D) = 1\} = \frac{2(2^{j+1}-1)-1}{2^{2j+1}} = \left(\frac{1}{2}\right)^{j-1} - 3\left(\frac{1}{2}\right)^{2j+1}.$$

2. Suppose that the permutation P_b swaps a column from the last j columns of K for a column $(\underline{s}_1 \mid \underline{s}_2)^T$ from R, where $\underline{s}_2 \in \mathrm{GF}(2)^j$. If $\underline{s}_2$ is the all-zero vector, then the first k columns of GP form a submatrix of rank $k-j$. The corresponding probability is $(\frac{1}{2})^j$.

Combining the two results proves (A.8).

Increasing Rank Two possibilities exist:

1. Let GP and D be the same as in the first situation of unchanged rank. If the rank of D is two, then the rank of GP is $k-j+1$. Therefore,

$$\Pr\{rank(D) = 2\} = \left(\frac{2^j-1}{2^j}\right)^2.$$

2. Consider the second situation of unchanged rank. The first k columns of GP form a submatrix of rank $k-j+1$, which has probability $1-(\frac{1}{2})^j$ provided $\underline{s}_2$ is *not* the all-zero vector.

Combining the two results yields (A.9). □

Example A.2.2 *For $k = 634$ and $0 \le j \le 5$, the values for q_{k-j}^{k-j+s} $(s \in \{-1,0,1\})$ are given in Table A.1. For $j > 3$, the probability that the rank of the K-matrix decreases, compared to the probability of an increased or an unchanged rank, is relatively small. The values for $p_{k-j}^{(u)}$ with $0 \le u \le 10$, $0 \le j \le 5$ and $k = 634$ are given in Table A.2.*

A.3 Rank Equilibrium State

Lemma A.3.1 *Let $\pi_r = \lim_{u\to\infty} p_r^{(u)}$ $(1 \le r \le k)$ with $p_r^{(u)}$ being the same as in Definition 5.3.2. For $0 \le j \le k-1$, it holds that*

$$\pi_{k-j} = \left\{ 1 + \sum_{m=1}^{j} \left(\prod_{l=m}^{j} \frac{q_{k-l}^{k-l+1}}{q_{k-l+1}^{k-l}} \right) + \sum_{m=j+1}^{k-1} \left(\prod_{l=j+1}^{m} \frac{q_{k-l+1}^{k-l}}{q_{k-l}^{k-l+1}} \right) \right\}^{-1}. \tag{A.10}$$

Proof: In the equilibrium distribution, $\underline{\pi}$ satisfies $\underline{\pi} = \underline{\pi}Q$, where Q is the rank-transition matrix (5.16). This matrix equation can be expressed as:

$$\begin{aligned}
\pi_1 &= \pi_2 q_2^1 + \pi_1 q_1^1 \\
\pi_2 &= \pi_3 q_3^2 + \pi_2 q_2^2 + \pi_1 q_1^2 \\
&\vdots \\
\pi_{k-2} &= \pi_{k-1} q_{k-1}^{k-2} + \pi_{k-2} q_{k-2}^{k-2} + \pi_{k-3} q_{k-3}^{k-2} \\
\pi_{k-1} &= \pi_k q_k^{k-1} + \pi_{k-1} q_{k-1}^{k-1} + \pi_{k-2} q_{k-2}^{k-1} \\
\pi_k &= \pi_k q_k^k + \pi_{k-1} q_{k-1}^k .
\end{aligned} \tag{A.11}$$

Define the sequence $\{s_{k-j}\}_{j=0,1,\ldots,k-1}$ by

$$s_k := q_k^k \tag{A.12}$$

$$s_{k-j} := \frac{q_{k-j}^{k-j+1} q_{k-j+1}^{k-j}}{1 - s_{k-j+1}} + q_{k-j}^{k-j}, \quad 1 \le j \le k-1. \tag{A.13}$$

From the recurrence relations (A.11), it now follows that:

$$\pi_{k-j+1} = \pi_{k-j} \frac{q_{k-j}^{k-j+1}}{1 - s_{k-j+1}}, \quad 1 \le j \le k-1. \tag{A.14}$$

Moreover, (A.11) implies that $s_1 = 1$. Relation (A.13) can be rewritten as

$$\begin{aligned}
s_{k-j+1} &= 1 - \frac{q_{k-j}^{k-j+1} q_{k-j+1}^{k-j}}{s_{k-j} - q_{k-j}^{k-j}}, \\
&= 1 - q_{k-j+1}^{k-j}, \quad 1 \le j \le k-1 && \text{(A.15)} \\
s_1 &= 1. && \text{(A.16)}
\end{aligned}$$

Substituting (A.15) and (A.16) into (A.14) yields

$$\pi_{k-j+1} = \pi_{k-j}\left(\frac{q_{k-j}^{k-j+1}}{q_{k-j+1}^{k-j}}\right) = \pi_1 \prod_{l=j}^{k-1} \frac{q_{k-l}^{k-l+1}}{q_{k-l+1}^{k-l}}, \quad 1 \leq j \leq k-1. \tag{A.17}$$

The sequence $\{\pi_{k-j+1}\}_{j=1,\dots,k}$ defines a probability distribution, therefore

$$\sum_{j=1}^{k} \pi_{k-j+1} = 1. \tag{A.18}$$

Next, substituting (A.17) into (A.18) gives the following expression for π_1:

$$\pi_1 = \left\{\sum_{m=1}^{k}\left(\prod_{l=m}^{k-1} \frac{q_{k-l}^{k-l+1}}{q_{k-l+1}^{k-l}}\right)\right\}^{-1}. \tag{A.19}$$

Substituting (A.19) into (A.17) yields

$$\pi_{k-j+1} = \frac{\prod_{l=j}^{k-1} \frac{q_{k-l}^{k-l+1}}{q_{k-l+1}^{k-l}}}{\sum_{m=1}^{k}\left(\prod_{l=m}^{k-1} \frac{q_{k-l}^{k-l+1}}{q_{k-l+1}^{k-l}}\right)}, \quad 1 \leq j \leq k. \tag{A.20}$$

After elementary calculations, the result (A.10) is obtained. □

Lemma A.3.2 *Let $\pi_k = \lim_{u\to\infty} p_k^{(u)}$ with $p_k^{(u)}$ as in Definition 5.3.2, then*

$$\pi_k = \left\{1 + \sum_{m=1}^{k-1}\left(\prod_{l=1}^{m} \frac{2(k-l+1)}{k(2^l-1)^2 + l(2^l-1)}\right)\right\}^{-1} \geq \frac{2}{7}. \tag{A.21}$$

Proof: Let $j = 0$ in (A.10), then the following expression for π_k is obtained:

$$\pi_k = \left\{1 + \sum_{m=1}^{k-1}\left(\prod_{l=1}^{m} \frac{q_{k-l+1}^{k-l}}{q_{k-l}^{k-l+1}}\right)\right\}^{-1}. \tag{A.22}$$

Applying Lemma A.2.1 to (A.22) leads to

$$\pi_k = \left\{1 + \sum_{m=1}^{k-1}\left(\prod_{l=1}^{m} \frac{2(k-l+1)}{k(2^l-1)^2 + l(2^l-1)}\right)\right\}^{-1}. \tag{A.23}$$

After elementary computations, it follows that

$$\begin{aligned}
&\sum_{m=1}^{k-1}\left(\prod_{l=1}^{m}\frac{2(k-l+1)}{k(2^l-1)^2+l(2^l-1)}\right)\le\sum_{m=1}^{k-1}2^m\left(\prod_{l=1}^{m}\frac{k-l+1}{k(2^l-1)^2}\right)\\
&\le\sum_{m=1}^{k-1}\left(\frac{2^m}{\prod_{l=1}^{m}(2^l-1)^2}\right)=\sum_{m=1}^{k-1}\frac{2^{-m^2}}{\left(\prod_{l=1}^{m}(1-2^{-l})\right)^2}\\
&\le 2+\frac{2^2}{3^2}+\frac{2^3}{3^2 7^2}+\sum_{m=4}^{\infty}\frac{2^{-m^2}}{\left(\prod_{l=1}^{m}1-\left(\frac{1}{2}\right)^l\right)^2}\le 2\frac{1}{2}.
\end{aligned}\qquad\text{(A.24)}$$

For the last inequality, note that for $m \geq 1$,

$$\begin{aligned}
\ln\prod_{l=1}^{m}\left(1-\left(\frac{1}{2}\right)^l\right) &= \sum_{l=1}^{m}\ln(2^l-1)-\sum_{l=1}^{m}\ln 2^l\\
&\ge -\sum_{l=1}^{m}\frac{1}{2^l-1}\ge -1-\sum_{l=1}^{m-1}\left(\frac{1}{2}\right)^l\ge -2,
\end{aligned}$$

which is an application of the *Mean Value Theorem.* Substituting (A.24) into (A.23) leads to the desired lower bound:

$$\pi_k\ge\frac{1}{1+\frac{5}{2}}=\frac{2}{7},$$

which proves the lemma. □

Lemma A.3.3 *Let* $\pi_{k-1} = \lim_{u\to\infty} p_{k-1}^{(u)}$ *with* $p_{k-1}^{(u)}$ *as in Definition 5.3.2, then*

$$\pi_{k-1}=2\left(1-\frac{1}{k+1}\right)\pi_k. \qquad\text{(A.25)}$$

Proof: Let $j = 1$ in (A.10), then the following expression for π_{k-1} is obtained:

$$\pi_{k-1}=\left\{1+\frac{q_{k-1}^{k}}{q_{k}^{k-1}}+\sum_{m=2}^{k-1}\left(\prod_{l=2}^{m}\frac{q_{k-l+1}^{k-l}}{q_{k-l}^{k-l+1}}\right)\right\}^{-1}. \qquad\text{(A.26)}$$

From Lemma A.2.1 and (A.26), it follows that

$$\begin{aligned}\pi_{k-1} &= \left\{1+\frac{k+1}{2k}+\sum_{m=2}^{k-1}\left(\prod_{l=2}^{m}\frac{2(k-l+1)}{k(2^l-1)^2+l(2^l-1)}\right)\right\}^{-1}\\ &= \frac{2k}{k+1}\left\{1+\sum_{m=1}^{k-1}\left(\prod_{l=1}^{m}\frac{2(k-l+1)}{k(2^l-1)^2+l(2^l-1)}\right)\right\}^{-1}\\ &= 2\left(1-\frac{1}{k+1}\right)\pi_k,\end{aligned}$$

which proves the lemma. □

Lemma A.3.4 *For $k \to \infty$ (i.e., $n \to \infty$), then*

$$\pi_{k-1} = 2\pi_k(1-o(1)). \tag{A.27}$$

Proof: The asymptotical expression follows from (A.25) since

$$\pi_{k-1} = 2\left(1-\frac{1}{k+1}\right)\pi_k = 2\pi_k(1+o(1)), \quad (k \to \infty),$$

which proves the lemma. □

Upper bounds are $\pi_k \le (3-\frac{2}{k+1})^{-1} \le \frac{1}{2}$ and $\pi_{k-1} \le 2(3+\frac{1}{k})^{-1} \le \frac{2}{3}$.

j	q_{k-j}^{k-j-1}	q_{k-j}^{k-j}	q_{k-j}^{k-j+1}
0	0.5	0.5	0
1	0.1248	0.6248	0.2504
2	0.0312	0.4058	0.5631
3	0.0078	0.2261	0.7661
4	0.0019	0.1188	0.8793
5	0.0005	0.0608	0.9387

Table A.1: Rank-transition probabilities for $k = 634$.

$j \backslash u$	0	1	2	3	4	5	6	7	8	9	10
0	1	0.5	0.3751	0.3284	0.3079	0.2983	0.2937	0.2914	0.2903	0.2898	0.2895
1	0	0.5	0.5624	0.5741	0.5767	0.5773	0.5774	0.5775	0.5775	0.5775	0.5775
2	0	0	0.0624	0.0955	0.1119	0.1200	0.1240	0.1260	0.1270	0.1275	0.1278
3	0	0	0	0.0019	0.0034	0.0043	0.0047	0.0050	0.0051	0.0051	0.0052
4	0	0	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table A.2: Rank probabilities $p_{k-j}^{(u)}$ for $k = 634$.

Appendix B

Independent Subsets

Coffey and Goodman [CG90b] proved that the fraction of randomly selected k-subsets out of n-tuples is almost an information set if $n \to \infty$ as shown in the following theorem:

Theorem B.0.5 *For sufficiently large n, virtually all linear codes in $C(n, q, R)$ contain no $\lfloor nR \rfloor$-tuple with fewer than $\lfloor nR(1-\alpha) \rfloor$ independent symbols* $(0 < \alpha < 1)$.

Although this theorem is correct, an imperfection appears in the proof. Coffey and Goodman assumed that the probability of obtaining a full rank $k \times n$ generator matrix tends to 1 as $n \to \infty$ with $k = n\mathrm{R}$ (i.e., they considered all $k \times n$ matrices over GF(q) instead of all $k \times n$ generator matrices).

A brief review of the *Kolmogorov complexity* is helpful before a correct proof can be given. (For details about Kolmogorov complexity, the reader is referred to [LV93].) Consider a string of q-ary symbols of length n. The aim is to remove the redundancy of this string, that is, to reduce the length of a string while guaranteeing unicity. This means that a q-ary string of length n must be identified with a unique program. In [Cha74], this length is referred to as the Kolmogorov complexity. Let $0 < m < n$, then the number of q-ary strings of length n that can be identified by a program of length less than or equal to

This appendix is based on joint work with Raymond Doyen [Doy92, DT94b].

$n - m$ is

$$\sum_{i=0}^{n-m} q^i = \frac{q^{n-m+1} - 1}{q - 1}.$$

Therefore, the fraction of such q-ary strings of length n is less than q^{-m+1}. This justifies the following proof based on [CG90b].

Proof: Let $\mathcal{C}$ be an arbitrary linear code in $\mathcal{C}(n, q, \mathrm{R})$ with generator matrix G. Consider all possible $k \times n$ generator matrices G belonging to linear codes of length n and dimension k. With a generator matrix G, associate the string $\underline{s}(G)$ of length nk (i.e., describe G row by row). Then, each generator matrix corresponds exactly to one string and vice versa.

A k-subset $\mathcal{I}$ of $\{1, 2, \ldots, n\}$ is said to be *h-independent* ($1 \leq h \leq k$), if the corresponding columns of the $k \times n$ matrix G have rank h. For $h = k$, the k-subset is an *Information Set.* A *seriously α-defective k-tuple* is defined as a k-tuple which is less than $\lfloor k(1-\alpha) \rfloor$-independent.

Suppose a generator matrix of a code contains k columns which correspond to a seriously α-defective k-tuple. This generator matrix (together with the defect) can be uniquely specified as follows:

- Specify the defective k-tuple (taking $\log_q \binom{n}{k}$ symbols).
- List the other $n - k$ columns in full (taking $k(n-k)$ symbols).
- List the $k(1-\alpha)$-independent columns belonging to the defective k-tuple (taking $k^2(1-\alpha)$ symbols).
- Specify each of the remaining $k\alpha$ columns, belonging to the defective k-tuple, by the linear combination of independent columns which span the same space (taking $k\alpha \times k(1-\alpha)$ symbols).

The total length of this program is

$$\beta + \log_q \binom{n}{k} + k(n-k) + k^2(1-\alpha) + k^2\alpha(1-\alpha) = n^2\mathrm{R} - n^2\mathrm{R}^2\alpha^2 + o(n^2),$$

where β is a constant. Consequently, the strings $\underline{s}(G)$ of length $n \times k = n^2\mathrm{R}$, which belong to a seriously α-defective k-tuple, can be identified by a program of length $n^2\mathrm{R} - n^2\mathrm{R}^2\alpha^2 + o(n^2)$. The number of strings of length $n^2\mathrm{R}$ that can be identified by a program of length, which is less than or equal to $n^2\mathrm{R} - n^2\mathrm{R}^2\alpha^2 + o(n^2)$, is

$$\frac{q^{n^2\mathrm{R}-n^2\mathrm{R}^2\alpha^2+o(n^2)}}{q-1} = q^{n^2\mathrm{R}-n^2\mathrm{R}^2\alpha^2+o(n^2)}. \tag{B.1}$$

This means that the number of seriously α-defective k-tuples belonging to a $k \times n$ generator matrix is less than or equal to (B.1). Consequently, the number of $k \times n$ generator matrices containing k columns, which correspond to a seriously α-defective k-tuple, is also less than or equal to (B.1).

A lower bound on the number of $k \times n$ generator matrices is

$$\prod_{i=0}^{k-1}(q^n - q^i) \geq q^{nk}(1-q^{k-n})^k \geq q^{n^2\mathrm{R}}\left(1 - n\mathrm{R}q^{-n(1-\mathrm{R})}\right). \tag{B.2}$$

Dividing (B.1) by (B.2) proves that the fraction of generator matrices containing k columns, which correspond to a seriously α-defective k-tuple, is less than or equal to

$$\begin{aligned}
&\frac{q^{n^2\mathrm{R}-n^2\mathrm{R}^2\alpha^2+o(n^2)}}{q^{n^2\mathrm{R}}\left(1 - n\mathrm{R}q^{-n(1-\mathrm{R})}\right)} \\
&= q^{-n^2\mathrm{R}^2\alpha^2+o(n^2)}\left(1 + O\left(n\mathrm{R}q^{-n(1-\mathrm{R})}\right)\right) \\
&= q^{-n^2\mathrm{R}^2\alpha^2+o(n^2)}.
\end{aligned}$$

Since this fraction tends to 0 for $n \to \infty$, then the fraction of linear codes of length n and rate R over GF(q), which contain an nR-tuple that is less than $\lfloor n\mathrm{R}(1-\alpha)\rfloor$-independent, also tends to 0 for $n \to \infty$. □

Bibliography

[AHU74] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *The design and analysis of computer algorithms*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1974.

[AK92] I.A. Al-Kadi, *Origins of cryptology: The Arab contributions*, Cryptologia **XVI** (1992), no. 2, 97–126.

[ALCdS93] F.M.R. Alencar, A.M.P. Léo, and R.M. Campello de Souza, *Private-key burst correcting code encryption*, Proceedings of the 1993 IEEE-ISIT, San Antonio, Texas USA (1993), 227.

[AM89] C. Adams and H. Meijer, *Security-related comments regarding McEliece's public-key cryptosystem*, IEEE Transactions on Information Theory **35** (1989), no. 2, 454–455.

[AW92a] M. Alabbadi and S.B. Wicker, *Cryptanalysis of the Harn and Wang modification of the Xinmei digital signature scheme*, Electron. Lett. **28** (1992), no. 18, 1756–1758.

[AW92b] M. Alabbadi and S.B. Wicker, *Security of Xinmei digital signature scheme*, Electron. Lett. **28** (1992), no. 9, 890–891.

[AW93] M. Alabbadi and S.B. Wicker, *Digital signature schemes based on error-correcting codes*, Proceedings of the 1993 IEEE-ISIT, San Antonio, Texas USA (1993), 199.

[Ber84] E.R. Berlekamp, *Algebraic coding theory*, revised 1984 Edition ed., Aegean Park Press, Laguna Hills, CA, 1984.

[Bla83] R.E. Blahut, *Theory and practice of error control codes*, Addison-Wesley Publishing Company, 1983.

[Bli87] V.M. Blinovskii, *Lower asymptotic bound on the number of linear codewords in a sphere of given radius in $GF(q)^n$*, Problemy Peredachi Informatsii **23** (1987), no. 2, 50–53.

[BMT78] E.R. Berlekamp, R.J. McEliece, and H.C.A. van Tilborg, *On the inherent intractability of certain coding problems*, IEEE Transactions on Information Theory **24** (1978), no. 3, 384–386.

[BN90] J. Bruck and M. Naor, *The hardness of decoding linear codes with preprocessing*, IEEE Transactions on Information Theory **36** (1990), no. 2, 381–385.

[BP82] H. Beker and F. Piper, *Cipher systems*, Wiley, New York, NY, 1982.

[BPV94] R. Govaerts B. Preneel and J. Vandewalle, *Computer security and industrial cryptography — state of the art and evolution*, Lecture Notes in Computer Science, vol. 741, Springer-Verlag, 1994, ESAT Course, Leuven, Belgium, May 1991.

[Bra88] G. Brassard, *Modern cryptography - a tutorial*, Lecture Notes in Computer Science, vol. 325, Springer-Verlag, Berlin-Heidelberg-New York, 1988.

[CdSCdS94] R.M. Campello de Souza and J. Campello de Souza, *Array codes for private-key encryption*, Electron. Lett. **30** (1994), no. 17, 1394–1396.

[CG81] A.H. Chan and R.A. Games, *(n, k, t)-covering systems and error-trapping decoding*, IEEE Transactions on Information Theory **27** (1981), no. 5, 643–646.

[CG90a] J.T. Coffey and R.M. Goodman, *Any code of which we cannot think is good*, IEEE Transactions on Information Theory **36** (1990), no. 6, 1453–1461.

[CG90b] J.T. Coffey and R.M. Goodman, *The complexity of information set decoding*, IEEE Transactions on Information Theory **36** (1990), no. 5, 1031–1037.

[CGF91] J.T. Coffey, R.M. Goodman, and P.G. Farrell, *New approaches to reduced-complexity decoding*, Discrete Applied Mathematics **33** (1991), 43–60.

[Cha72] D. Chase, *A class of algorithms for decoding block codes with channel measurement information*, IEEE Transactions on Information Theory **18** (1972), no. 4, 170–182.

[Cha74] G.J. Chaitin, *Information-theoretic computational complexity*, IEEE Transactions on Information Theory **20** (1974), no. 10, 10–15.

[Cha94] F. Chabaud, *On the security of some cryptosystems based on error-correcting codes*, Lecture Notes in Computer Science, Springer-Verlag, 1994, To appear.

[Coo71] S.A. Cook, *The complexity of theorem-proving procedures*, Proceedings of the 3rd Anual ACM Symposium on Theory of Computing, 1971, pp. 151–158.

[Den82] D. Denning, *Cryptography and data security*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1982.

[Den88] W.F. Denny, *Encryptions using linear and non-linear codes: Implementation and security considerations*, Ph.D. thesis, The Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, Spring 1988.

[DH76] W. Diffie and M. E. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory **22** (1976), no. 6, 644–654.

[Doy92] R. Doyen, *On the security of the McEliece public-key cryptosystem with respect to information set decoding*, Tech. report, Technische Universiteit Eindhoven, 1992, M.Sc. Thesis.

[DP84] D. W. Davies and W. L. Price, *Security for computer networks*, (New York), John Wiley & Sons, 1984.

[DT94a] R. Doyen and J. van Tilburg, *The complexity of Chase's bounded soft-decision decoding algorithm*, Publication PU-94-175, PTT Research, P.O. Box 421, 2260 AK, Leidschendam, the Netherlands, February 1994.

[DT94b] R. Doyen and J. van Tilburg, *The complexity of the bit swapping algorithm with respect to information set decoding*, Publication PU-94-28, PTT Research, P.O. Box 421, 2260 AK, Leidschendam, the Netherlands, January 1994.

[Dum89] I.I. Dumer, *Two decoding algorithms for linear codes*, Problemy Peredachi Informatsii **25** (1989), no. 1, 24–32.

[Dum91] I.I. Dumer, *On minimum distance decoding of linear codes*, Fifth Joint Soviet-Swedish International Workshop on Information Theory, January 1991, pp. 50–52.

[Dum93] I.I. Dumer, *Suboptimal decoding of linear codes*, EUROCODE'92, CISM Courses and Lectures, vol. 339, 1993, pp. 369–382.

[Dum94] I.I. Dumer, September 1994, Private communication.

[EH63] P. Erdös and Hanani. H., *On a limit theorem in combinatorial analysis*, Publ. Math. Debrecen **10** (1963), 10–13.

[ES74] P. Erdös and J. Spencer, *Probabilistic methods in combinatorics*, Academic Press, New York, NY, 1974, A series of Monographs and Textbooks.

[Evs83] G.S. Evseev, *Complexity of decoding for linear codes*, Problemy Peredachi Informatsii **19** (1983), no. 1, 3–8.

[Fel57] W. Feller, *An introduction to probability theory and its applications*, second ed., vol. 1, John Wiley and Sons, Inc., New York, 1957.

[FR85] P. Frankl and V. Rödl, *Near perfect covering in graphs and hypergraphs*, Europ. J. Combinatorics **6** (1985), 317–326.

[Gib91a] J.K. Gibson, *Equivalent Goppa codes and the McEliece's public key cryptosystem*, Tech. report, Department of Computer Science, Birkbeck College, London, England, April 1991.

[Gib91b] J.K. Gibson, *Equivalent Goppa codes and trapdoors to McEliece's public key cryptosystem*, Advances in Cryptology – Proceedings of Eurocrypt'91 (D. W. Davies, ed.), Lecture Notes in Computer Science, vol. 547, Springer-Verlag, 1991, pp. 517–521.

[Gib93] J.K. Gibson, *Severely denting the Gabidulin version of the McEliece public key cryptosystem*, Tech. report, Department of Computer Science, Birkbeck College, London, England, April 1993.

[Gil52] E.N. Gilbert, *A comparison of signalling alphabets*, Bell System Tech. J. **31** (1952), 504–522.

[GJ79] M.R. Garey and D.S. Johnson, *Computers and intractability – A guide to the theory of NP-completeness*, W.H. Freeman and Company, San Fransisco, 1979.

[GPT91] E.M. Gabidulin, A.V. Paramonov, and O.V. Tretjakov, *Ideals over a non-commutative ring and their application in cryptology*, Advances in Cryptology – Proceedings of Eurocrypt'91 (D. W. Davies, ed.), Lecture Notes in Computer Science, vol. 547, Springer-Verlag, 1991, pp. 482–489.

[Har89] S. Harari, *A new authentication algorithm*, Coding Theory and Applications (G. Cohen and J. Wolfmann, eds.), Lecture Notes in Computer Science, vol. 388, Springer-Verlag, 1989, pp. 91–105.

[Hei87] R. Heiman, *On the security of cryptosystems based on linear error-correcting codes*, Tech. report, Feinburg Graduate School, Weizmann Institute of Science, Rehovot, Israel, August 1987, M.Sc. Thesis.

[Hin86a] P.J.M. Hin, December 1986, Private communication.

[Hin86b] P.J.M. Hin, *Channel-error-correcting privacy cryptosystems*, Tech. report, Delft University of Technology, 1986, M.Sc. Thesis (in Dutch).

[HR90a] T. Hwang and T.R.N. Rao, *Private-key algebraic-code cryptosystems with high information rates*, Advances in Cryptology – Proceedings of Eurocrypt'89 (J.-J. Quisquater and J. Vandewalle, eds.), Lecture Notes in Computer Science, vol. 434, Springer-Verlag, 1990, pp. 657–661.

[HR90b] T. Hwang and T.R.N. Rao, *Secret error-correcting codes (SECC)*, Advances in Cryptology – Proceedings of Crypto'88 (S. Goldwasser, ed.), Lecture Notes in Computer Science, vol. 403, Springer-Verlag, 1990, pp. 540–563.

[HW92] L. Harn and D.-C. Wang, *Cryptanalysis and modification of digital signature scheme based on error-correcting code*, Electron. Lett. **28** (1992), no. 2, 157–159.

[Jor83] J.P. Jordan, *A variant of a public-key cryptosystem based on Goppa codes*, Sigact news **15** (1983), no. 1, 61–66.

[Jor86] F. Jorissen, *A security evaluation of the public-key cipher system proposed by R.J. McEliece, used as a combined scheme*, Tech. report, Katholieke Universiteit Leuven, Departement Elektrotechniek, Afdeling ESAT, Januari 1986.

[Kah67] D. Kahn, *The codebreakers: The story of secret writing*, Macmillan, New York, 1967.

[Kar72] R.M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Computations (New York) (R.E. Miller and J.W. Tatcher, eds.), Plenum Press, 1972, pp. 85–104.

[Kob88] N. Koblitz, *A course in number theory and cryptography*, second ed., Springer-Verlag, New York, 1988.

[Kra86] E Kranakis, *Primality and cryptography*, John Wiley and Sons, Chichester, great Britain, 1986.

[Kru89] E.A. Kruk, *Decoding complexity bound for linear block codes*, Problemy Peredachi Informatsii **25** (1989), no. 3, 103–107, (Translation from the Russian original.).

[KT91] V.I. Korzhik and A.I. Turkin, *Cryptanalysis of McEliece's public-key cryptosystem*, Advances in Cryptology – Proceedings of Eurocrypt'91 (D. W. Davies, ed.), Lecture Notes in Computer Science, vol. 547, Springer-Verlag, 1991, pp. 68–70.

[LB88] P.J. Lee and E.F. Brickell, *An observation on the security of McEliece's public-key cryptosystem*, Advances in Cryptology – Proceedings of Eurocrypt'88 (C.G. Günther, ed.), Lecture Notes in Computer Science, vol. 330, Springer-Verlag, 1988, pp. 153–157.

[LCF90] M.-C. Lin, T.-C. Chang, and H.-L. Fu, *Information rate of McEliece's public-key cryptosystem*, Electron. Lett. **26** (1990), no. 1, 16–17.

[LDW94] Y.X. Li, R.H. Deng, and X.M. Wang, *On the equivalence of McEliece's and Niederreiter's public-key cryptosystem*, IEEE Transactions on Information Theory **40** (1994), no. 1, 271–273.

[Leo88] J.S. Leon, *A probabilistic algorithm for computing minimum weights of large error-correcting codes*, IEEE Transactions on Information Theory **34** (1988), no. 5, 1354–1359.

[LH85] L.B. Levitin and C.R.P. Hartmann, *A new approach to the general minimum distance decoding problem — the zero-neighbours algorithm*, IEEE Transactions on Information Theory **31** (1985), no. 3, 378–384.

[Lin82] J.H. van Lint, *Introduction to coding theory*, Graduate Texts in Mathematics, vol. 86, Springer-Verlag, New York, 1982.

[LV93] M. Li and P.M.B. Vitanyi, *An introduction to Kolmogorov complexity and its applications*, Springer-Verlag, Berlin-Heidelberg-New York, 1993, Texts and Monographs in Computer Science Series.

[LW91] Y.X. Li and X.M. Wang, *A joint authentication and encryption scheme based on algebraic coding theory*, Applied algebra, Algebraic algorithms and Error Correcting Codes 9 (H.F. Mattson, T. Mora, and T.R.N. Rao, eds.), Lecture Notes in Computer Science, vol. 539, Springer-Verlag, 1991, pp. 241–245.

[Mau90] U. Maurer, *Provable security in cryptography*, Ph.D. thesis, Swiss Federal Institute of Technology, Zürich, 1990, Diss. ETH No. 9260.

[McE78] R.J. McEliece, *A public-key cryptosystem based on algebraic coding theory*, DSN Progress Report 42–44, JPL, Pasadena, January and February 1978.

[Mei90] J. Meijers, *Algebraic-coded cryptosystems*, Tech. report, Technische Universiteit Eindhoven, 1990, M.Sc. Thesis.

[Men93] A. Menezes, *Elliptic curve public-key cryptosystems*, Kluwer Academic Publishers, Dordrecht, 1993.

[Mer78] R.C. Merkle, *Secure communication over insecure channels*, Comm. ACM **21** (1978), no. 4, 294–299.

[Mil79] W.H. Mills, *Covering designs I: coverings by a small number of subsets*, Ars Combinatoria **8** (1979), 199–315.

[MM82] C. Meyer and S. Matyas, *Cryptography: A new dimension in computer security*, John Wiley and Sons, New York, 1982.

[MS77] F.J. MacWilliams and N.J.A. Sloane, *The theory of error-correcting codes*, Elsevier Science Publishers, Amsterdam, North-Holland, 1977.

[MT91a] J. Meijers and J. van Tilburg, *An improved ST-attack on the Rao-Nam private-key cryptosystem*, International Conference on Finite Fields, Coding Theory, and Advances in Communications and Computing (1991), Las Vegas, Nevada, USA.

[MT91b] J. Meijers and J. van Tilburg, *On the Rao-Nam private-key cryptosystem using linear codes*, IEEE 1991 International Symposium on Information Theory (1991), 126, Budapest, Hungarije.

[MT93] J. Meijers and J. van Tilburg, *Extended majority voting and private-key algebraic-code encryptions*, Advances in Cryptology - Proceedings of Asiacrypt'91 (R. L. Rivest H Imai and T. Matsumoto, eds.), Lecture Notes in Computer Science, vol. 739, Springer-Verlag, 1993, pp. 288–298.

[Nie86] H. Niederreiter, *Knapsack-type cryptosystems and algebraic coding theory*, Problems of Control and Information Theory **15** (1986), no. 2, 159–166.

[Par89] C.S. Park, *Improving code rate of McEliece's public-key cryptosystem*, Electron. Lett. **25** (1989), no. 21, 1466–1467.

[PBGV92] B. Preneel, A. Bosselaers, R. Govaerts, and J. Vandewalle, *A software implementation of the McEliece publick-key cryptosystem*, Proceedings of the Thirdteenth Symposium on Information Theory in the Benelux (1992), 119–126.

[Pie67] J.N. Pierce, *Limit distribution of the minimum distance of random linear codes*, IEEE Transactions on Information Theory **13** (1967), no. 4, 595–599.

[PS82] C.H. Papadimitrou and K. Steiglitz, *Combinatorial optimization: Algorithms and complexity*, Prentice-Hall, 1982.

[PW72] W.W. Peterson and E.J. Weldon, *Error-correcting codes*, MIT Press, Cambridge, 1972.

[QD91] J.-J. Quisquater and Y.G. Desmedt, *Chinese lotto as an exhaustive code-breaking machine*, Computer **24** (1991), no. 11, 14–22.

[Rab80] M.O. Rabin, *Probabilistic algorithms in finite fields*, SIAM Journal on Computing **9** (1980), 273–280.

[Rao84] T.R.N. Rao, *Cryptosystems using algebraic codes*, Int. Conf. on Computer Systems & Signal Processing, December 1984, Bangalore, India.

[Rao88] T.R.N. Rao, *On Struik-Tilburg cryptanalysis of Rao-Nam scheme*, Advances in Cryptology – Proceedings of Crypto'87 (C. Pomerance, ed.), Lecture Notes in Computer Science, vol. 293, Springer-Verlag, 1988, pp. 458–460.

[RN87] T.R.N. Rao and K.H. Nam, *Private-key algebraic-coded cryptosystems*, Advances in Cryptology – Proceedings of Crypto'86 (A. M. Odlyzko, ed.), Lecture Notes in Computer Science, vol. 263, Springer-Verlag, 1987, pp. 35–48.

[RN89] T.R.N. Rao and K.H. Nam, *Private-key algebraic-code encryptions*, IEEE Transactions on Information Theory **35** (1989), no. 4, 829–833.

[Roc92] V.C. da Rocha, *Secret-key cryptosystems based on algebraic codes*, Proceedings of the 1992 IEEE-IT workshop, Brazil (1992), 92–93.

[Röd85] V. Rödl, *On a packing and covering problem*, Europ. J. Combinatorics **6** (1985), 69–78.

[RT94] P.L.A. Roelse and J. van Tilburg, *A probabilistic approach to syndrome decoding with applications in cryptography*, Publication PU-94-417, PTT Research, P.O. Box 421, 2260 AK, Leidschendam, the Netherlands, February 1994.

[Rue86] R Rueppel, *Design and analysis of stream ciphers*, Springer-Verlag, Berlin-Heidelberg-New York, 1986.

[Sac58] G.E. Sacks, *Multiple error correcting by means of parity checks*, IEEE Transactions on Information Theory **4** (1958), 145–147.

[Sch74] J. Schönheim, *Covering problems*, Pacific J. Math. **14** (1974), 1401–1411.

[Sch94] B. Schneier, *Applied cryptography*, John Wiley and Sons, New York, 1994.

[Sha49] C.E. Shannon, *Communication theory of secrecy systems*, Bell System Tech. J. **28** (1949), 656–713.

[Sim92] G.J. Simmons (ed.), *Contemporary cryptology. The science of information integrity*, IEEE Press, 1992.

[SS92] V.M. Sidelnikov and S.O. Shestakov, *On cryptosystems based on generalized Reed-Solomon codes*, Diskretnaya Math **4** (1992), 57–63, In Russian.

[ST88] R. Struik and J. van Tilburg, *The Rao-Nam scheme is insecure against a chosen-plaintext attack*, Advances in Cryptology – Proceedings of Crypto'87 (C. Pomerance, ed.), Lecture Notes in Computer Science, vol. 293, Springer-Verlag, 1988, pp. 445–457.

[Ste89] J. Stern, *A method for finding codewords of small weight*, Coding Theory and Applications (G. Cohen and J. Wolfmann, eds.), Lecture Notes in Computer Science, vol. 388, Springer-Verlag, 1989, pp. 106–113.

[Ste90] J. Stern, *An alternative to the Fiat-Shamir protocol*, Advances in Cryptology – Proceedings of Eurocrypt'89 (J.-J. Quisquater and J. Vandewalle, eds.), Lecture Notes in Computer Science, vol. 434, Springer-Verlag, 1990, pp. 173–180.

[Ste94] J. Stern, *A new identification scheme based on syndrome decoding*, Advances in Cryptology – Proceedings of Crypto'93 (D.R. Stinson, ed.), Lecture Notes in Computer Science, vol. 773, Springer-Verlag, 1994, pp. 13–21.

[Sti95] D.R. Stinson, *Cryptography: Theory & practice*, CRC Press, Inc, CRC Press, Inc., publication date: February 1995, ISBN: 0-8493-8521-0.

[Str87] R. Struik, *Algebraic-coded cryptosystems*, Stageverslag (in Dutch), 1987.

[Str91] R. Struik, *On the Rao-Nam private-key cryptosystem using non-linear codes*, IEEE 1991 International Symposium on Information Theory (1991), 174, Budapest, Hungarije.

[Str93] R. Struik, *An efficiency enhancement of the McEliece cryptosystem*, Private communication, 1993.

[Til88] H.C.A. van Tilborg, *An introduction to cryptology*, Kluwer Academic Publishers, Dordrecht, 1988.

[Til90] J. van Tilburg, *On the McEliece public-key cryptosystem*, Advances in Cryptology – Proceedings of Crypto'88 (S. Goldwasser, ed.), Lecture Notes in Computer Science, vol. 403, Springer-Verlag, 1990, pp. 119–131.

[Til92] J. van Tilburg, *Cryptanalysis of the Xinmei digital signature scheme*, Electron. Lett. **28** (1992), no. 20, 1935–1936.

[Til93a] J. van Tilburg, *Cryptanalysis of the Alabbadi-Wicker digital signature scheme*, Proceedings of the Fourteenth Symposium on Information Theory in the Benelux (1993), 114–119.

[Til93b] J. van Tilburg, *On the nonexistence of digital signature schemes based on maximum likelihood decoding*, International Conference on Finite Fields: Theory Applications and Algorithms (1993), Las Vegas, Nevada, USA.

[Til93c] J. van Tilburg, *Two chosen-plaintext attacks on the Li-Wang joint authentication and encryption scheme*, Applied algebra, Algebraic algorithms and Error Correcting Codes 10 (G. Chen, T. Mora, and O. Moreno, eds.), Lecture Notes in Computer Science, vol. 673, Springer-Verlag, 1993, pp. 332-343.

[Til94] J. van Tilburg, *A code-equivalence problem*, Finite Fields: Theory, Applications & Algorithms, Contemporary Mathematics, vol. 168, American Mathematical Society, 1994.

[TQ93] J. van Tilburg and J.-J. Quisquater, *Security-related comments regarding the McEliece cryptosystem and an ID-based cryptosystem*, Rump session Crypto'93 (unpublished), Santa Barbara CA, USA, August 1993.

[Var57] R.R. Varshamov, *Estimate of the number of signals in error correcting codes*, Dokl. Akad. Nauk SSSR **117** (1957), 739–741.

[Ver26] G.S. Vernam, *Cipher printing telegraph systems for secret wire and radio telegraphic communications*, J. Amer. Inst. Elec. Eng. **55** (1926), 109–115.

[Wol78] J.K. Wolf, *Efficient maximum likelihood decoding of linear block codes using a trellis*, IEEE Transactions on Information Theory **24** (1978), no. 1, 76–80.

[Xin90] W. Xinmei, *Digital signature scheme based on error-correcting codes*, Electron. Lett. **26** (1990), no. 13, 898–899.

Index

Samenvatting

Het toepassen van cryptologie, de leer van het geheimschrift, is de laatste jaren zeer sterk toegenomen. Tegenwoordig wordt bijna iedereen, al dan niet bewust, geconfronteerd met de vele facetten van de cryptologie. Belangrijke oorzaak is de toename van diensten op telecommunicatie- en informatiegebied, waarbij steeds meer informatie, ook van persoonlijke aard, voor derden bereikbaar is. Daarnaast worden bedrijfsvoeringsprocessen steeds meer afhankelijk van computer- en telecommunicatiesystemen. Beveiliging van informatie en informatiestromen is daarom een essentieel aspect van de datacommunicatie geworden.

Een belangrijk middel om een informatieproces te beveiligen is het vercijferen of versluieren van informatie. Dit vindt zijn oorsprong in een deelgebied van de cryptologie: de cryptografie. De cryptografie houdt zich bezig met het ontwerpen van cryptosystemen waarmee informatie vercijferd kan worden voor geheimhouding (privacy en anonimiteit) en/of authenticiteitsgarantie (identificatie en integriteit). Een tweede deelgebied van de cryptologie, de crypto-analyse, houdt zich bezig met het analyseren van geheimschriften. Het mag duidelijk zijn dat men de cryptografie niet goed kan beoefenen zonder voldoende kennis van de crypto-analyse. Belangrijk hierbij is ook de stand van de techniek. Naast de kwaliteit van het vercijferalgoritme zijn ook de manier waarop de vercijfering wordt toegepast, de identificatiemethode, het bijbehorend communicatieprotocol, het (sleutel-)beheersysteem en het onderliggende informatieproces van belang.

Random-eigenschappen spelen een belangrijke rol in de cryptologie. Niet alleen bij het vercijferen van een boodschap, maar onder andere ook bij het genereren van willekeurige getallen (sleutels), priemgetallen, het factoriseren van getallen, het nemen van discrete logaritmen

en in anonimiteitsprotocollen. Daarnaast zijn random-eigenschappen van belang bij het vercijferen van een boodschap. Een belangrijke vercijfermethode is gebaseerd op het volgende *kruis-of-munt* principe: de uitkomst (één of nul) van een worp met een zuivere munt opgeteld (modulo 2) bij de uitkomst van een worp met een valse munt heeft toch een willekeurige (niet voorspelbare) uitkomst als resultaat (d.w.z. random + niet random = random)!

Vernam [Ver26] gebruikte dit principe door bij een boodschaprij een random sleutelrij op te tellen, waardoor een random cryptogramrij wordt verkregen. Shannon [Sha49] bewees dat in Vernam's cryptosysteem de boodschap perfect verborgen blijft, mits de random sleutelrij tenminste net zo lang is als de boodschaprij en slechts éénmaal wordt gebruikt (one-time pad). In de praktijk wordt veelal een lange sleutelrij afgeleid uit een vele malen kleinere random sleutelrij (kortweg sleutel genoemd). Deze sleutelrij is echter deterministisch en wordt vaak pseudo-random rij genoemd. In de cryptologie moet een pseudo-random rij niet alleen goede statistische eigenschappen bezitten, maar moet ook cryptografisch sterk zijn. Het komt erop neer dat het ondoenlijk moet zijn een cryptografisch sterke pseudo-random rij van een echte random rij te onderscheiden.

In Vernam's vercijfermethode wordt gebruik gemaakt van het globaal randomiseren van een boodschap. De volgende vercijfermethode is gebaseerd op het lokaal randomiseren van de boodschap. Een boodschap wordt hiervoor opgedeeld in blokken van een bepaalde lengte. Ieder boodschapblok wordt met een lineaire foutverbeterende code eerst omgezet in een codewoord. Vervolgens wordt bij het codewoord een foutvector (willekeurig gekozen uit de verzameling van toegestane foutvectoren) opgeteld. Op deze manier wordt een vercijferde boodschap verkregen. Met andere woorden, de boodschaprij wordt lokaal gerandomiseerd. Een legale gebruiker kan met kennis van een bijbehorend decodeeralgoritme de foutvector corrigeren en de oorspronkelijke boodschap bepalen. Een niet-geautoriseerde gebruiker kent het decodeeralgoritme niet en moet een veel complexer decodeerprobleem oplossen. De veiligheid van dit soort cryptosystemen hangt af van de complexiteit van dit laatste decodeerprobleem.

De klasse van cryptosystemen, bestudeerd in dit proefschrift, wordt gekarakteriseerd door het gebruik van lineaire foutverbeterende codes.

Belangrijkste voordelen van dit soort cryptosystemen zijn de hoge snelheden die kunnen worden bereikt met een lage systeemcomplexiteit [Sim92]. Voor een veiligheidsanalyse van dit soort cryptosystemen, beschrijft dit proefschrift een nieuw en universeel probabilistisch decodeeralgoritme. Voor dit decodeeralgoritme zijn de rekentijd, het geheugengebruik en de asymptotische complexiteit volledig uitgewerkt. De complexiteit van het nieuwe decodeeralgoritme komt overeen met informatie set decoderen. Het nieuwe decodeeralgoritme bezit op het moment, vergeleken met bekende universele decodeeralgoritmen, de laagste rekentijd en een minimaal geheugengebruik voor het genoemde decodeerprobleem.

Met behulp van de in dit proefschrift verkregen resultaten is een gebied van waarden voor de codeparameters bepaald dat leidt tot onveilige cryptosystemen. Verder zijn de optimale codeparameters berekend om de complexiteit van het decodeerprobleem voor de niet-geautoriseerde gebruiker te maximaliseren. Drie onveilige digitale handtekeningenschema's gebaseerd op lineaire foutverbeterende codes zijn geïdentificeerd. Verder is voor een aantal verwante geheime sleutel cryptosystemen aangetoond dat ze niet de geclaimde cryptografische sterkte bezitten.

About the Author

Johan van Tilburg (Hans) completed the theoretical requirements towards a technical diploma in Electronics from the Anthony Fokker School of Aviation and Electronics in 1977. His course work included aiding in the design and implementation of a Digital Phase-Meter and a Transistor Curve-Tracer. The same year, he began a study in Electrical Engineering at the College of Advanced Technology, The Hague. In 1980, at the Rand Afrikaan University (RAU), Johannesburg, he worked on the Electric Vehicle Project which met the practical requirements for both studies. Under the supervision of Ir. B.W.A. van Dorp at PTT Research, he met his final requirement with the implementation of a Mini-Floppy-Disk System, and then received the equivalent of a B.Sc. degree in Electrical Engineering with emphasis in Telecommunications in 1981. In 1985, he received his M.Sc. with honors degree in Electrical Engineering from Delft University of Technology. His theses, *Decisions and Selections Based on the Average Error Probability with Respect to Shannon Information, Certainty and f-Divergence* and *An Information-Theoretic Approach to Cryptology*, where under the supe~vision of Prof. dr. ir. D.E. Boekee.

In 1986, he joined PTT Research were his activities included helping to start the Crypto Research Group, securing the Dutch Mobile Telephone System and providing design work for TIRO crypto products. In 1988, after a year's work at the Department of Defense. he rejoined PTT Research where his research included fast software implementations for protocols based on the discrete-logarithm problem, key-management and cryptanalysis.

Johan van Tilburg has lectured to the former Eindhoven International Institute (EII), to the Katholieke Universiteit Leuven for the ESAT course on *Computer Security and Industrial Cryptography*, and to the Euler Institute of Discrete Mathematics and its Applications (EIDMA). He has served on the *Eurocrypt'87* organizing committee and the *Eurocrypt'89* program committee. He is a member of the *Institute of Electrical and Electronics Engineers* (IEEE) and the *International Association for Cryptologic Research* (IACR), and is the newsletter editor for the *Information and Communication Theory Society* (WIC) in the Benelux. His interests include statistical information theory, source coding, pattern recognition, cryptology, and data security for telecommunication systems. In these areas, he has (co-)authored more than 30 articles for publications and international conference proceedings.

Stellingen

behorende bij het proefschrift

Security-Analysis of a Class of Cryptosystems Based on Linear Error-Correcting Codes

door

Johan van Tilburg

geboren te Voorburg (Z-H)

1. In general, the key equivocation is a poor measure of theoretical security.

   J. van Tilburg and D.E. Boekee, *Divergence bounds on key equivocation and error probability in cryptanalysis*, Advances in Cryptology – Proceedings of Crypto'85 (H. C. Williams, ed.), Lecture Notes in Computer Science, vol. 218, Springer-Verlag, 1986, pp. 489–513.

2. The Unicity Distance is usually defined or interpreted incorrectly in literature.

   J. van Tilburg and D.E. Boekee, *The Pe-security distance as a measure of cryptographic performance*, Traitement de L'Information **4** (1987), no. 6, 479–483.

3. The matrices $(W|H^T)$ and $(J|T)$ in the Xinmei digital signature scheme [1] always have full rank as shown in [2]. Therefore, the omitted examples in [3] do not exist.

   [1] W. Xinmei, *Digital signature scheme based on error-correcting codes*, Electron. Lett. **26** (1990), no. 13, 898–899.

   [2] J. van Tilburg, *Cryptanalysis of the Xinmei digital signature scheme*, Electron. Lett. **28** (1992), no. 20, 1935–1936.

   [3] Y.-X. Li, *An attack on Xinmei's digital signature scheme*, Proceedings of the 1993 IEEE-ISIT, San Antonio, Texas USA (1993), 236.

4. A digital signature scheme whose security relies only on the Bounded Hard-Decision Decoding (BHDD) problem does not exist.

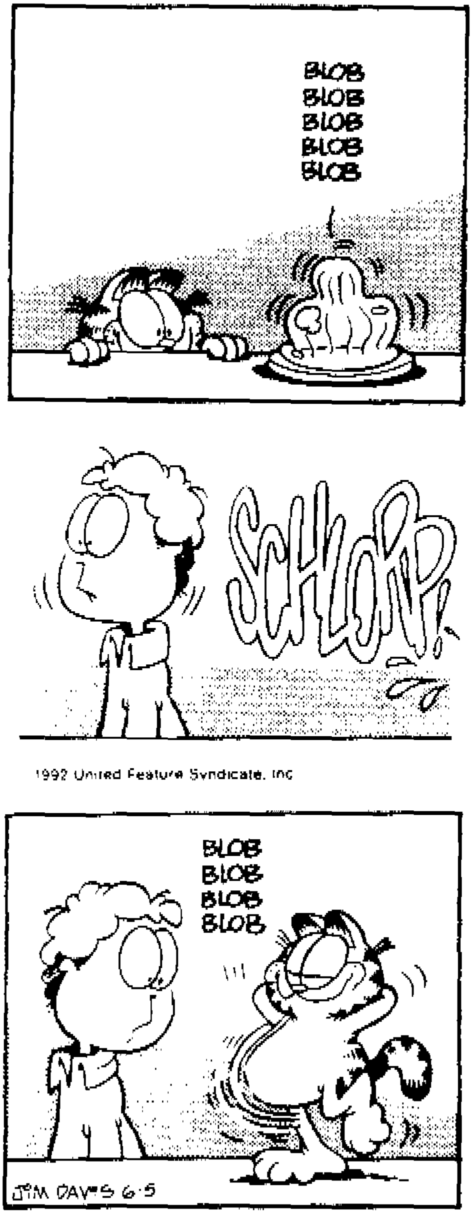
5. If you are going to put all your eggs in one basket, make sure that they are scrambled.

6. A Personal Identification Number is a PIN that will not stick.

7. An asymmetric cryptosystem is not necessarily a public-key cryptosystem.

8. Security is a management problem.

9. Good customer service does not treat people like numbers, however to be served in order, we are usually given a number.



1992 United Feature Syndicate, Inc

In: Jim Davis, *Garfield pocket 21.*

10. The figure above is a generalized visualization of the Blob given in [1].

[1] E.J.L.J. van Heijst, Stellingen behorende bij het proefschrift *Special Signature Schemes.*