Application of Linear Block Codes in Cryptography

by

Mostafa Esmaeili
B.Sc., Isfahan University of Technology, Iran, 2009
M.Sc., Isfahan University of Technology, Iran, 2012

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

Application of Linear Block Codes in Cryptography

by

Mostafa Esmaeili
B.Sc., Isfahan University of Technology, Iran, 2009
M.Sc., Isfahan University of Technology, Iran, 2012

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Stephen W. Neville, Departmental Member
(Department of Electrical and Computer Engineering)

Dr. Bruce Kapron, Outside Member
(Department of Computer Science)

ABSTRACT

Recently, there has been a renewed interest in code based cryptosystems. Amongst the reasons for this interest is that they have shown to be resistant to quantum attacks, making them candidates for post-quantum cryptosystems. In fact, the National Institute of Standards and Technology is currently considering candidates for secure communication in the post-quantum era. Three of the proposals are code based cryptosystems. Other reasons for this renewed interest include efficient encryption and decryption. In this dissertation, new code based cryptosystems (symmetric key and public key) are presented that use high rate codes and have small key sizes. Hence they overcome the drawbacks of code based cryptosystems (low information rate and very large key size). The techniques used in designing these cryptosystems include random bit/block deletions, random bit insertions, random interleaving, and random bit flipping. An advantage of the proposed cryptosystems over other code based cryptosystems is that the code can be/is not secret. These cryptosystems are among the first with this advantage. Having a public code eliminates the need for permutation and scrambling matrices. The absence of permutation and scrambling matrices results in a significant reduction in the key size. In fact, it is shown that with simple random bit flipping and interleaving the key size is comparable to well known symmetric key cryptosystems in use today such as Advanced Encryption Standard (AES).

The security of the new cryptosystems are analysed. It is shown that they are immune against previously proposed attacks for code based cryptosystems. This is because scrambling or permutation matrices are not used and the random bit flipping is beyond the error correcting capability of the code. It is also shown that having a public code still provides a good level of security. This is proved in two ways, by finding the probability of an adversary being able to break the cryptosystem and showing that this probability is extremely small, and showing that the cryptosystem has indistinguishability against a chosen plaintext attack (i.e. is IND-CPA secure). IND-CPA security is among the primary necessities for a cryptosystem to be practical. This means that a ciphertext reveals no information about the corresponding plaintext other than its length. It is also shown that having a public code results in smaller key sizes.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank my family for all their support and help throughout this degree. Without their kindness and sacrifices I would not have been able to complete this degree. I would also like to thank my supervisor, Dr. Gulliver, who has helped me a tremendous amount. His help, support, and dedication to the success of his students not only in academia, but also outside of it is everything a graduate student would ask for. I would also like to thank Dr. Kapron for the time he spent on my ideas, reading and revising my papers and his help in improving my dissertation. His approaches and suggestions have made significant improvements in my research. Without his help and dedication, the last chapter (which I believe is 'the créme de la créme' of all chapters) of my dissertation would not even be close to what it is right now.

Chapter 1

Introduction

Reliability and security are two essential components of any communication system. Reliability is provided by using channel coding while security is provided by using encryption. In most communication systems, channel coding and encryption are done separately. In this dissertation, efficient encryption techniques based on channel codes and combining coding and encryption are presented.

In 1948, Shannon demonstrated that with an appropriate encoding scheme the number of errors induced by a communication channel can be reduced to any desired level as long as the information rate is less than the channel capacity [1]. Since then researchers have invested significant time and effort in finding efficient encoding and decoding techniques for controlling errors in noisy channels [2–6]. In this chapter, an introduction to encoding and decoding is provided. As previously mentioned, cryptography is used to provide a desired level of security in communication systems. Cryptography is the art of securing a message from anyone who is not supposed to access it. For many years cryptography was employed by governments and military. However, due to the widespread commercial use of computer networks and the internet and the significant amount of research in cryptography, it now has many commercial applications. Cryptography has become an essential tool used in many everyday tasks (e.g. online banking). In this chapter, various terms used in cryptography are defined. The algebraic structure of channel codes, particularly linear block codes, and how they can be used to construct cryptosystems are also provided.

1.1 Cryptography

In this section, the terminology used in cryptography is defined. Cryptography is the science of keeping a message secure. In cryptography, a message is known as a *plaintext*, denoted by $\boldsymbol{m}$. The algorithm to disguise a plaintext in order to hide its information is known as *encryption*. An encrypted plaintext is called a *ciphertext*. The algorithm used to recover the plaintext from a ciphertext is called *decryption*. The encryption (decryption) algorithm uses a *key* to find the ciphertext (plaintext) associated with a plaintext (ciphertext). In other words the key determines the changes that have to be made at each step of encryption (decryption) to a plaintext (ciphertext) to obtain the ciphertext (plaintext).

At this point a cryptosystem can be defined. A *cryptosystem* is a five-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{P}$ is the set of all plaintexts;
- $\mathcal{C}$ is the set of all ciphertexts;
- $\mathcal{K}$ is the set of all keys;
- for every $\mathcal{k} \in \mathcal{K}$, there exists an encryption algorithm $Enc_{\mathcal{k}} \in \mathcal{E}$ and a decryption algorithm $Dec_{\mathcal{k}} \in \mathcal{D}$ such that for every $\boldsymbol{m} \in \mathcal{P}$, $Dec_{\mathcal{k}}(Enc_{\mathcal{k}}(\boldsymbol{m})) = \boldsymbol{m}$.

Attempting to find the key used in an encryption algorithm is called *cryptanalysis*. If cryptanalysis is performed by an authorized user, it is usually done to measure the security of encryption and possibly improve it. On the other hand, if cryptanalysis is done by an unauthorized user (known as an *adversary*), it is to disrupt secure communication by decrypting ciphertexts. This is called an *attack*.

There are two types of cryptosystems: public key cryptosystems and symmetric key cryptosystems. In symmetric key cryptosystems, encryption and decryption are done using the same keys. Some well known symmetric key cryptosystems are the Advanced Encryption Standard (AES) [7] and Data Encryption Standard (DES) [8]. In public key cryptosystems, each user has two keys; a private key and a public key. Encryption is done using the public key while decryption is done using the private key. Well known public key cryptosystems include the Rivest-Shamir-Adleman (RSA), El-Gamal, and McEliece cryptosystems [8,9]. Later in this chapter a detailed introduction to the McEliece cryptosystem and its variants will be provided.

1.2 Linear block codes

In this section the terminology of channel codes is defined. A brief introduction to the algebraic structure of linear block codes is provided. A channel encoder is used to transform a data sequence (known as a *message*) $\boldsymbol{u} = (u_1, u_2, \ldots, u_k)$ into an encoded sequence (known as a *codeword*) $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$. In this dissertation, it is assumed that a message and its associated codeword are binary sequences. An encoder transforms a message into a codeword. Since there are 2^k distinct messages, there will be 2^k distinct codewords. The set of codewords of length n is called a $C(n, k)$ *block code*. The ratio $R = \frac{k}{n}$ is called the *rate* of a code. Note that $k \leq n$ or $R \leq 1$, so each codeword has $n - k$ more bits than the message associated with it. These extra bits are used to detect and correct errors introduced by a noisy channel.

Block codes can be divided into linear and non-linear block codes. Non-linear block codes are not used due to their complexity. Therefore, only linear block codes will be considered here. A $C(n, k)$ block code is *linear* if and only if its codewords form a k-dimensional subspace of the vector space of all n-tuples. From the definition of a linear code, it follows that there exist k linearly independent codewords $\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_k$ such that every codeword $\boldsymbol{c}$ is a linear combination of them. If $\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_k$ are arranged as the rows of a $k \times n$ matrix given by

$$G = \begin{bmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \vdots \\ \boldsymbol{g}_k \end{bmatrix},$$

then the codeword $\boldsymbol{c}$ associated with message $\boldsymbol{u}$ can be found via $\boldsymbol{c} = \boldsymbol{u}G$. The rows of G generate the codewords of a $C(n, k)$ block code. Thus G is called a *generator matrix*. Finding the codeword $\boldsymbol{c}$ associated with a message $\boldsymbol{u}$ is known as *encoding*.

For any k-dimensional subspace S of a vector space of all n-tuples, there exists an $(n - k)$-dimensional subspace S_d such that every vector in S_d is orthogonal to every vector in S. The subspace S_d is known as the *dual space* of S. As previously mentioned, codewords of a block code form a subspace of the vector space of all n-tuples. Hence, every block code has a dual space. Let G and H denote the generator matrices of a block code and its dual space, respectively. Based on the definitions of subspace and dual space, $GH^T = \mathbf{0}$ where T and $\mathbf{0}$ denote transposition and the all

zero matrix, respectively. The matrix H is called the *parity check matrix* of $C(n,k)$. A $C(n,k)$ block code can be fully described by either its generator matrix G or its parity check matrix H.

For a codeword $\boldsymbol{c}$, the Hamming weight (or just weight) of $\boldsymbol{c}$, denoted by $w(\boldsymbol{c})$, is defined as the number of non-zero bits in $\boldsymbol{c}$. The *distance* between two codewords $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$, $1 \leq i,j \leq 2^k$, denoted by $d(\boldsymbol{c}_i, \boldsymbol{c}_j)$, is defined as the number of positions in which $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ differ. The minimum distance of a $C(n,k)$ block code, denoted by $d_{min}(C)$, is defined as $\min_{1\leq i,j\leq 2^k, i\neq j} d(\boldsymbol{c}_i, \boldsymbol{c}_j)$. For linear block codes $d_{min}(C) = \min_{1\leq i\leq 2^k} w(\boldsymbol{c}_i)$ where $\boldsymbol{c}_i \neq \boldsymbol{0}$. The minimum distance of a $C(n,k)$ block code determines how many errors it can detect and correct. It is easy to show that a $C(n,k)$ block code with minimum distance $d_{min}(C)$ can detect $d_{min}(C)-1$ errors and correct $t = \lfloor \frac{d_{min}(C)-1}{2} \rfloor$ errors. The parameter $t = \lfloor \frac{d_{min}(C)-1}{2} \rfloor$ is called the *error correcting capability* of the code. In communication systems, correcting errors from a received word to find a codeword is known as *decoding*. In the next section it will be shown how linear block codes can be used to construct a public key cryptosystem.

1.3 The McEliece cryptosystem

In 1978 it was shown that decoding a linear code without knowledge of its algebraic structure is an NP-complete problem [10]. This suggested that it is possible to construct a cryptosystem using linear codes. Later that year, the first cryptosystem based on linear codes was introduced by McEliece [9], and thus is known as the McEliece cryptosystem. The McEliece cryptosystem scrambles bits of a plaintext, encodes, and then permutes and randomly flips some bits of the associated codeword. Scrambling, encoding and permuting are done by a $k \times k$ non-singular matrix S, the generator matrix G of the $C(n,k)$ block code and an $n \times n$ matrix P, respectively. Random flipping is done by adding a random error vector with a weight within the error correcting capability of the code. This is a public key cryptosystem. The private key is the three matrices S, G and P, and the public key is the product of them. As previously mentioned, in a public key cryptosystem encryption is done using the public key while decryption uses the private key. In the cryptography literature there are two characters who encrypt plaintexts and decrypt ciphertexts. Encryption is done by Alice and decryption by Bob. In the next section, encryption and decryption using the McEliece cryptosystem is explained.

1.3.1 Encryption and decryption algorithms

In the McEliece cryptosystem there is a code, represented by its generator matrix G, a scrambling matrix S, and a permutation matrix P. The public key is SGP and the private key is (S^{-1}, G, P^{-1}). The encryption algorithm of the McEliece cryptosystem is as follows.

1. For a plaintext $\boldsymbol{m}$, Alice encodes it using $\boldsymbol{c} = \boldsymbol{m}SGP$ using Bob's public key.

2. She then chooses a random error vector $\boldsymbol{e}$ of length n such that $w(\boldsymbol{e}) \leq t$, where t is the error correcting capability of the code Bob uses. The ciphertext $\boldsymbol{c}'$ associated with $\boldsymbol{m}$ is $\boldsymbol{c}' = \boldsymbol{c} + \boldsymbol{e} = \boldsymbol{m}SGP + \boldsymbol{e}$.

To decrypt a ciphertext Bob does the following

1. For a ciphertext $\boldsymbol{c}'$, Bob finds $P^{-1} = P^T$. He then multiplies $\boldsymbol{c}'$ by P^{-1} to obtain

$$\boldsymbol{c}'P^{-1} = (\boldsymbol{m}SGP + \boldsymbol{e})P^{-1} = \boldsymbol{m}SG + \boldsymbol{e}P^{-1}.$$

2. Since P is a permutation matrix, $P^{-1} = P^T$ will also be a permutation matrix. This implies that $\boldsymbol{e}P^{-1}$ is a vector with weight less than or equal to t. Hence $\boldsymbol{c}'P^{-1}$ can be decoded to obtain $\boldsymbol{m}S$.

3. By multiplying $\boldsymbol{m}S$ by S^{-1}, the plaintext $\boldsymbol{m}$ is found.

In the next section, a security analysis of the McEliece cryptosystem is provided.

1.3.2 Security analysis

There are three main attacks against the McEliece cryptosystem. The first is where an adversary attempts to find G from the public key. This attack is known as a *structural attack*. Depending on the chosen code, there are numerous possibilities for S, G, and P [9]. Hence, the probability of finding G from a structural attack is very small. The second attack is to directly find the plaintext from a ciphertext without having the private key. The second attack is more promising. This attack is done by finding k random bits of a ciphertext that have not been flipped and then the corresponding plaintext. This is known as an *information set decoding* attack. It was shown that the probability of finding k error free bits in a ciphertext is $(1 - \frac{t}{n})^k$ and

the amount of work required is $k^3(1-\frac{t}{n})^{-k}$ [9]. McEliece suggested using a Goppa code with parameters $n = 1024$, $k = 524$ and $t = 50$. For these parameters, the work required for each choice of $k = 524$ bits is $2^{65} \approx 10^{19}$.

The third type of attack is a *decoding attack*. Decoding attacks are designed to solve the decoding problem, that is finding the random error vector used in encrypting a plaintext. This is accomplished by finding the codeword with minimum weight in the code given by the generator matrix

$$G' = \begin{pmatrix} G \\ \boldsymbol{c}' \end{pmatrix},$$

where $\boldsymbol{c}'$ is the ciphertext for which the adversary wants to find the plaintext [11]. These attacks usually have a smaller work factor than structural attacks and are more effective [12]. It was shown in [11] that the probability of finding the random error vector is very small. To date, no polynomial time attack has been proposed for the McEliece cryptosystem.

1.4 Drawbacks of the McEliece and improved code based cryptosystems

Although the McEliece cryptosystem has not been broken yet, it has not been used in a real application. The reason is that it has two main drawbacks, namely a large key size and a low information rate. With the suggested code in [9], the private key size is almost 227 KB and the information rate is slightly more than 0.5. However, code based cryptosystems have been the subject of recent research. The factors contributing to this are its resistance to quantum attacks and its efficient encryption and decryption [12–14]. Various modifications have been proposed to increase the information rate of these cryptosystems [15–24] and decrease the key size [25–36].

Codes have also been used to construct symmetric key cryptosystems. In [37], a symmetric key code based cryptosystem was introduced where the error correcting capability of the code is used to remove errors and provide an acceptable level of security. In [38], this system was modified to use simpler codes in order to reduce the key size and increase the information rate. This system is vulnerable to some chosen plaintext attacks, but it has been improved by using non-linear codes and modifying the set of allowable error patterns [39]. The cryptosystems proposed in [30] and [31]

have been successfully broken in [40] and [41], respectively. The attacks proposed in [41] use an algebraic approach based on a system of bi-homogeneous polynomial equations to recover the code. Some private key code based cryptosystems are based on inserting random bits at random positions in a codeword [42]. It was shown in [42] that obtaining the codeword corresponding to a given plaintext which is a punctured version of the received ciphertext, is an NP-complete problem. Random puncturing has also been used to construct symmetric key code based cryptosystems [17]. In this system, turbo codes are punctured according to the channel noise. If the channel is very noisy, only a few bits are punctured, otherwise more are punctured. Unfortunately, this puncturing can significantly increase the probability of decoding error at the receiver.

Other proposals for code based cryptosystems use quasi-cyclic (QC) [30], quasi-dyadic (QD) [31], and quasi-cyclic low density parity check (QC-LDPC) codes [32]. The simple structure of their parity check matrix result in small key sizes. The cryptosystem proposed in [28] has a public key consisting of a matrix $H' = T \times H$ where H is the parity check matrix of the code and T is a high density matrix, i.e. with a large number of ones. Hence H' is such that if an adversary uses the public key to decode a received word, decoding will be very inefficient and likely fail. The information rate is better than the McEliece cryptosystem (≈ 0.67), but the key size is still large (2.5 kB). In [27], the security and efficiency of this cryptosystem was improved by modifying the allowable set of error patterns which do not necessarily have small weight. The resulting key size is reduced (≈ 2.5 kb) compared to similar cryptosystems while the error performance of the code is unchanged. However, the information rate is smaller (≈ 0.5).

1.5 Contributions and outline

In this dissertation different techniques other than permuting and scrambling are used to construct symmetric and public key cryptosystems. These techniques include random bit/block deletion, random bit/block insertion, random interleaving, and bit flipping beyond the error correcting capability of the code. It will be shown that these simple techniques can provide high levels of security, eliminating the need for permutation and scrambling. Furthermore, the code can be public (i.e. not secret) which allows for the use of high rate codes. These cryptosystems are among the first to have public codes. The absence of permutation and scrambling matrices results in

a reduction in the key size compared to previous code based cryptosystems. In fact, it will be shown that the key size is comparable to symmetric key cryptosystems in use today.

The rest of the dissertation is as follows. In Chapter 2, two symmetric key code based cryptosystems are presented. The first cryptosystem is a symmetric key cryptosystem based on random deletion of bits in a codeword [43]. It is shown that simple random bit deletion can result in a good level of security. In this cryptosystem the code is secret. However, it is shown that if the code is revealed the cryptosystem will remain secure. The second cryptosystem is an extension of the first one. In the second cryptosystem in addition to deleting random bits of a codeword, random bits are also inserted into it. It is shown that adding this simple technique can make a significant improvement in the security.

In Chapter 3, two symmetric key cryptosystems are proposed. The first is based on random bit deletion, random bit insertion and random bit flipping beyond the error correcting capability of the code. This cryptosystem has a public code. It is shown that even though the code is public (i.e. the structure of the code is known), decrypting a ciphertext is a very hard task and can be successfully accomplished with a very low probability. The second cryptosystem employs random bit flipping and random block deletions to encrypt a plaintext. It is shown that for a given security level, deleting blocks instead of bits can result in a smaller key size compared to bit deletion. The key size is also comparable to symmetric key cryptosystems that are used today (e.g. AES).

In Chapter 4, random interleaving is used to construct a symmetric key code based cryptosystem. The interleaving process can be viewed as inserting blocks of bits into an erroneous codeword. These blocks are from the codeword of another plaintext. Hence this cryptosystem encrypts two plaintexts at a time. It is shown that this simple technique provides a high level of security while having a very small key size. In fact, the key size is smaller than that of the cryptosystems presented in Chapter 3.

In Chapter 5, the random bit flipping technique used in Chapters 3 and 4 and random bit padding is used to construct a public key code based cryptosystem. Similar to the cryptosystems in Chapters 3 and 4, this cryptosystem also has a public code. It is shown that the combination of randomly padding a plaintext and randomly flipping more bits of a codeword than a code can correct results in indistinguishability against chosen plaintext attack (IND-CPA) security. This implies that a ciphertext

reveals no information about the corresponding plaintext other than its length. It is also shown that this cryptosystem has a much smaller key size than that of the McEliece cryptosystem. If the same code is used in both cryptosystems, the proposed cryptosystem has a key size 75% less than that of the McEliece cryptosystem. Finally, in Chapter 6, a summary is given and suggestions for future work are provided.

Chapter 2

Code Based Cryptosystems with Random Deletions and Insertions

In this chapter, two new symmetric key code based cryptosystems are introduced. The first cryptosystem is based on randomly deleting bits of a codeword. It is shown that with this simple approach, higher rate codes can be used and the key size can be reduced compared to other symmetric key code based cryptosystems. It is also shown that in the case that the code is revealed, the cryptosystem will still maintain a high level of security. The second cryptosystem is a modified version of the first. This modification is achieved by not only randomly deleting bits from a codeword, but also inserting random bits in random positions. As with the first cryptosystem, this cryptosystem is shown to have good security if the code is revealed.

2.1 Secure channel coding with random deletions

In this section, a new symmetric key code based cryptosystem is introduced. This cryptosystem is based on randomly puncturing bits of a codeword associated to a plaintext. This system consists of two parts, a $C(n, k)$ block code characterized by its parity check matrix and a pseudo-random number generator (PRNG). The code is constructed using an extended difference family (EDF) as described in [44]. The definition of an EDF is given below.

Definition 1. Let $\mathcal{F} = \{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \ldots, \mathcal{B}_\tau\}$ be a set of sets of ω integers (i.e. $\mathcal{B}_i, 1 \leq i \leq \tau$, is a set of integers). Let $\mathcal{D}_i, 1 \leq i \leq \tau$, be the set of differences of all two elements in $\mathcal{B}_i$. If all sets $\mathcal{D}_i$ are distinct, $\mathcal{F}$ is a (ω, τ)-EDF.

A method for finding a (ω, τ)-EDF is given in [44]. In this cryptosystem a $C(n, k)$ block, where $n = m\tau$, $k = m(\tau - 1)$, and m is an integer, is used along with a linear feedback shift register (LFSR) for the PRNG. Note that an LFSR is used only for illustrative purposes. In the case of implementing any of these cryptosystems a more secure PRNG should be used. The key consists of ω, the parity check matrix of the $C(n, k)$ block code, and the initial state of the LFSR.

2.1.1 Encryption and decryption algorithms

Alice and Bob construct a $C(n, k)$ block code with a generator matrix G and decide on how many bits should be punctured from the codeword corresponding to a plaintext. The encryption algorithm is as follows:

1. For a plaintext $\boldsymbol{m}$, Alice finds its associated codeword via $\boldsymbol{c} = \boldsymbol{m}G$.
2. She then punctures the bits of $\boldsymbol{c}$ at indexes determined by the PRNG. The remaining bits are the ciphertext corresponding to $\boldsymbol{m}$.

The decryption algorithm is given below.

1. To decrypt a ciphertext Bob has to recover the punctured bits. Since he has the same PRNG as Alice, he knows the indexes of the codeword bits that have been punctured. These punctured bits can be considered as erasures, and can be recovered using erasure decoding.
2. Once the codeword has been obtained, Bob finds the corresponding plaintext.

Code based cryptosystems have also been used in joint channel coding and cryptography (also known as secure channel coding) schemes [37], [27]. The main purpose of this technique is to provide both encryption and reliable data transmission. The advantages of this approach are increased speed, efficient implementation and a trade-off between security and reliable communication so that one may be preferred over the other. To use the proposed cryptosystem for joint channel coding and cryptography, Alice and Bob must agree on a $C(n, k)$ block code constructed via a (m, ω, τ)-EDF and how many bits of each codeword are to be punctured. The block diagram in Figure 2.1 illustrates how the proposed joint channel coding and cryptography system works. For a plaintext $\boldsymbol{m}$, Alice finds it corresponding ciphertext $\overline{\boldsymbol{c}}$ using the encryption algorithm. This ciphertext is transmitted over the channel to Bob. Suppose the

received word is $\overline{\boldsymbol{r}}$. To obtain $\boldsymbol{m}$, Bob employs two decoding steps. Since Bob has the parity check matrix for the punctured code, he first decodes $\overline{\boldsymbol{r}}$ to obtain $\overline{\boldsymbol{c}}$. He then proceeds to find the plaintext $\boldsymbol{m}$ using the decryption algorithm.

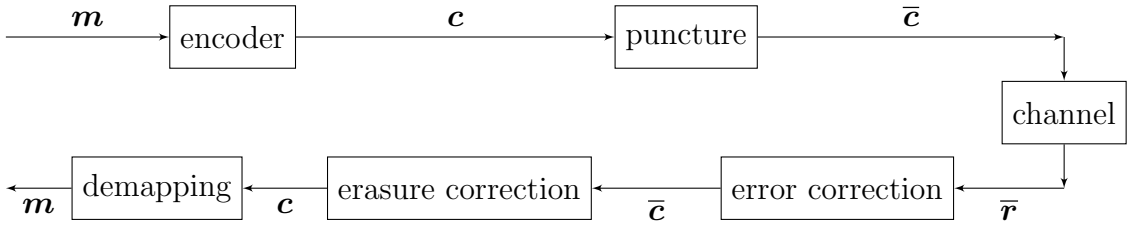

Figure 2.1: Block diagram of the joint channel coding-cryptography scheme based on random deletions.

In the next section the key size of this joint channel coding scheme will be analysed.

2.1.2 Key size

As previously mentioned, in this cryptosystem the key consists of ω, the parity check matrix of the $C(n,k)$ block code, and the initial state of the LFSR. How to use the LFSR to generate pseudo-random numbers for puncturing is determined by Alice and Bob. It was suggested that the decimal equivalent of the state of the LFSR will represent the positions to be punctured. In this case, suppose that Alice and Bob have agreed to puncture β bits. Therefore, each codeword is divided into $\alpha = \left\lfloor \frac{n}{\beta} \right\rfloor$ sub-blocks, where $n = m\tau$. If all β bits are to be punctured in one clock pulse, an LFSR of length $l = \frac{n}{\alpha} \times \lfloor \log_2 \alpha \rfloor$ is required. In general, if an $C(m\tau, m(\tau-1))$ block code and an LFSR of length l are used, the key size will be $m\tau + l + \lceil \log_2 \omega \rceil$. For example, for a $C(2048, 1536)$ block code obtained from a (16,4)-EDF and a 192 bit LFSR, the key size will be 2191 bits ($\approx$ 2.19 kbits). This key size is smaller than previously symmetric key code based cryptosystems, as shown in Table 2.1.

In the next section the error performance of the suggested code to be used in the scheme is analysed.

2.1.3 Error performance

It is well known that puncturing a code degrades its error performance. Hence to show that the suggested code can be used in the secure channel coding scheme and still maintain a desirable level of error performance, the bit error rate (BER) of the

Table 2.1: Key Size For Some Private Key Code Based Cryptosystems

Scheme	Code	Key size
Rao [37]	C(1024,524,101)	2 Mbits
RN [38]	C(72,64,3)	18 kbits
Struik-Tilberg [39]	C(72,64,3)	18 kbits
Sun-Shieh [45]	C(49,36)	42 kbits
Barbero-Ytrehus [46]	C(30,20) over F_{2^8}	4.9 kbits
proposed	C(2048,1536) with 32 bits punctured	2.191 kbits

punctured code is compared to a random code with the same length and dimension on an AWGN channel. Figure 2.2 shows that the suggested code outperforms a random code. Thus randomly puncturing a code provides better performance in the secure channel coding scheme. In the next section the security of the cryptosystem will be

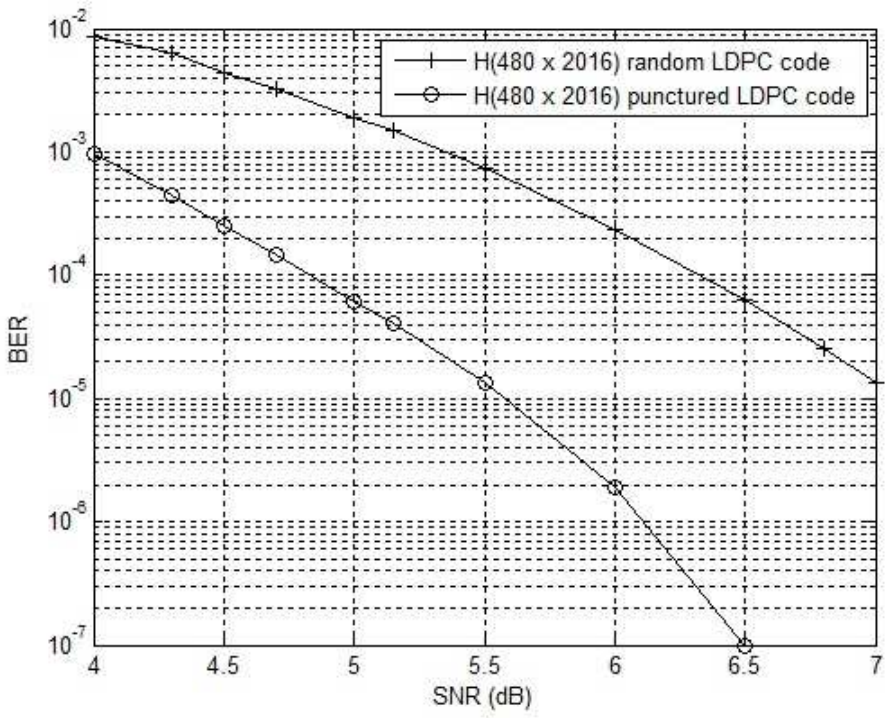

Figure 2.2: BER performance of a punctured C(2016,1536) LDPC code and a random C(2016,1536) LDPC code on an AWGN channel.

analysed.

2.1.4 Security analysis

Although the number of punctured bits is small compared to the code length, it will be shown that this provides a high level of security. In the proposed cryptosystem, bits of a codeword corresponding to the plaintext are randomly omitted. Conversely, in other cryptosystems based on error correcting codes the bits of a codeword corresponding to the plaintext are scrambled, permuted and randomly changed. This difference in structure results in the proposed system being immune against attacks on algebraic coded cryptosystems (e.g. the Stern [11], Struik [39], and RN [38] attacks).

As explained in Section 1.3.2, information set decoding attack is another main threat to code based cryptosystems. In this attack, an adversary randomly chooses k bits of an n bit ciphertext. This is repeated until a valid message $\boldsymbol{m}$ is recovered (i.e., the k bits are error free). A systematic procedure to determine whether $\boldsymbol{m}$ is actually the message $\boldsymbol{m}'$ sent by Alice is provided in [47]. In this approach, denote $\overline{\boldsymbol{c}}_k$ as the k random bits of the ciphertext $\overline{\boldsymbol{c}}$. Let G_k be the $k \times k$ matrix obtained from the corresponding columns of the generator matrix of the code. If $\boldsymbol{m} = \overline{\boldsymbol{c}}_k G_k^{-1}$ is not $\boldsymbol{m}'$, then $\boldsymbol{m}'G + \boldsymbol{m}G$ must have weight at least equal to $2t$ since the minimum distance of the code is greater than $2t$. Otherwise, the adversary can claim that $\boldsymbol{m} = \boldsymbol{m}'$. However, this attack fails in the first step when applied to the proposed cryptosystem because there is no known method to determine the columns of G corresponding to the chosen k bits unless the puncturing is known.

To analyse the security of the proposed cryptosystem, two cases are considered. In the first case, the adversary does not have any knowledge of the code employed. The main advantage of using an EDF to construct a code is that a large number of equivalent codes can be constructed from one extended difference family. For example, the number of codes of rate $R = 0.75$ and length $n = 2048$ bits with parity check matrix column weight 4 that can be constructed from a (70,4)-EDF is greater than 2^{79}. Thus in this case the cryptographic system is robust to brute force attacks.

In the second case, the code used in the cryptosystem is known by the adversary. Since the code is known, all that remains to break the system is to find the initial state of the LFSR, but as will be shown, this task is very difficult. To find the initial LFSR state, the adversary has to find the current state by guessing the correct positions of the punctured bits of a ciphertext. If the current state is obtained, the initial state can easily be determined via the relation $\mathbf{s}_{t_0} = C^{-t_c}\mathbf{s}_{t_c}$, where $\mathbf{s}_{t_0}$ and $\mathbf{s}_{t_c}$ are the state vectors at times t_0 (initial state) and t_c (current state), respectively. C^{-t_c} is the t_c-th

inverse of the matrix

$$C = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ c_0 & c_1 & c_2 & \cdots & c_{l-1} \end{pmatrix},$$

where $c_0, c_1, \ldots, c_{l-1}$ are the feedback coefficients of the LFSR. If an adversary knows the current time t_c, then C^{-t_c} is known and finding $\mathbf{s}_{t_0}$ from $\mathbf{s}_{t_0} = C^{-t_c}\mathbf{s}_{t_c}$ has complexity $O(l)$, where l is the length of the LFSR. Therefore the security of the system lies in the positions of the punctured bits. Although the length of the LFSR is not large (here $l = 192, 224$), it will be shown that the probability of guessing the current state is very small. How to guess the current state is given in the following attack.

Step 1: An adversary chooses a plaintext $\boldsymbol{m}$ corresponding to a codeword $\boldsymbol{c}$ which is more likely to break the cryptosystem. Codewords which are more likely to do so will be discussed shortly.

Step 2: By giving $\boldsymbol{m}$ to the cryptosystem and obtaining the ciphertext $\overline{\boldsymbol{c}}$, an adversary can compare $\boldsymbol{c}$ and $\overline{\boldsymbol{c}}$ and guess which indexes have been punctured.

Step 3: Having guessed the indexes of the punctured bits, the adversary has guessed the current state of the LFSR.

Suppose that the code employed has length $n = m\tau$ and β bits are punctured. Hence the parent code is divided into sub-blocks of length $\alpha = \lfloor \frac{m\tau}{\beta} \rfloor$ bits. The probability of an adversary correctly guessing which bit is punctured in a sub-block is $\frac{1}{\alpha}$. Thus the probability of guessing all the punctured bits correctly is $\left(\frac{1}{\alpha}\right)^{\beta}$. For a $C(2048, 1536)$ code with $\beta = 32$ bits, this probability is 2^{-192}.

The worst case is when an adversary has the ciphertext corresponding to the codewords $1010101010\cdots$ or $0101010101\cdots$. Then by comparing the ciphertext and codeword, the current state of the LFSR can easily be determined as there will be two consecutive ones or zeros. It should be noted that the probability of such codewords occurring is very low, $O(2^{-k})$. However, the problem can be eliminated by changing the ciphertext corresponding to these codewords. Without loss of generality, suppose the codeword corresponding to the plaintext is $1010101010\ldots$ In this case, if the LFSR has determined that a 1 is to be deleted, also puncture the 0 to its right. If a 0 is to be deleted, also puncture the 1 to its left. In either case, a pair 10 will be punctured from each sub-block of the codeword. If the ciphertext associated with a codeword is $0101010101\ldots$, the same procedure can be used, except that in either case a pair 01

will be punctured from each sub-block of the codeword. This solution significantly decreases the probability of determining the correct LFSR state. In the general case, suppose a code of length $n = m\tau$ is used in the proposed cryptosystem. If β bits are randomly punctured, there are $\frac{\alpha}{2}$ possible positions where the 10 (or 01) pair can be punctured with $\alpha = \left\lfloor \frac{n}{\beta} \right\rfloor$. The probability of guessing which bit of the 10 (or 01) pair has been punctured is $(\frac{1}{2})^{\beta}$. Therefore, in this case, the probability of guessing the correct state is $2^{-\beta} \times \left(\frac{\alpha}{2}\right)^{\beta}$. For a $C(2048, 1536)$ code with 32 bits randomly punctured, the procedure described will be successful with probability 2^{-192}.

With the above modification to the system, the best situation for an adversary is that the sub-blocks of a codeword have the form $110011001100\ldots$ or its complement. In this case, either a 0 or a 1 is punctured from a 00 or 11 pair, respectively. Hence the probability that an adversary will be able to guess which one of the bits from each pair was deleted is equal to $\frac{1}{2}$. For a length n code with β bits randomly punctured, this procedure of determining which bits were punctured in each sub-block of the codeword will succeed with probability $2^{-\beta}$. For a $C(2048, 1536)$ code with 32 bits punctured and a 192 bit LFSR, the probability of finding the correct initial state is $\left(\frac{1}{2}\right)^{32}$. Although this attack has a low probability of succeeding, modifying how the LFSR output is employed can reduce the probability of guessing the correct LFSR state, as shown below.

As before, suppose each sub-block in a codeword has length α. An LFSR of length $l = \frac{n}{\alpha}\left(\lfloor \log_2 \alpha \rfloor + 1\right)$ is chosen where n is the code length. The bits to be punctured from each sub-block are determined in the following way. Divide the LFSR state into $\frac{n}{\alpha}$ parts each consisting of $\lfloor \log_2 \alpha \rfloor + 1$ bits. If the first bit of a part is equal to 1, the next $\lfloor \log_2 \alpha \rfloor$ bits determine the bit to be punctured, otherwise the next $\lfloor \log_2 \alpha \rfloor$ bits are ignored. Thus each part that begins with a 0 is ignored.

It is obvious that in this approach not all the bits to be punctured may be determined in one clock cycle. Therefore, with this approach the rate of encryption is decreased, but it will be shown that the probability of guessing the correct LFSR state is very small. In the general case, suppose that a code of length $n = m\tau$ and an LFSR of length l are employed in the cryptosystem. If β bits are to be punctured and the adversary knows that in the first clock cycle $\gamma < \beta$ bits are punctured, it can be assumed that the first bit of γ parts of the LFSR are equal to 1 and the other $\beta - \gamma$ bits are 0. The probability that an adversary guesses which bit from each sub-block is punctured is $(\frac{1}{2})^{\gamma}$. However, he has no knowledge of the remaining $l - \beta - \gamma \log_2 \alpha$ bits of the LFSR. Hence all possible values must be tested. Therefore, the probability

of success of this attack is at most $2^{-(\gamma+l-\beta-\gamma\log_2\alpha)}$. Note that in the LFSR, the binary representation of the bit to be punctured in the first sub-block of the codeword should come before the binary representation of the position to be punctured in the second sub-block, and so on. This means that the $\lfloor\log_2\alpha\rfloor+1$ bits determining the bit that should be punctured in the first sub-block of the codeword should be on the left of the $\lfloor\log_2\alpha\rfloor+1$ bits determining which bit of the second sub-block should be punctured, and so on. Clearly there is more than one way for this to occur. This decreases the probability of determining the correct LFSR state and thus makes the attack more difficult.

For a $C(2048, 1536)$ code with 32 bits punctured and an LFSR of length 224 bits, assuming that the adversary has knowledge that 2 bits were punctured in the first clock cycle, the probability of finding the correct LFSR state will be 2^{-183}. If the adversary does not know how many bits are punctured in one clock pulse, all possible values have to be tested. This will result in a probability of success equal to 2^{-218}, which is very low. Hence in the unlikely event of the used code being revealed the cryptosystem still has a high level of security.

2.2 Encryption and decryption using random insertions and deletions

In this section, a new symmetric key coding based cryptosystem is presented which randomly inserts and deletes bits in the codeword corresponding to a plaintext. This is an improvement of the approach in Section 2.1 where only deletions were employed. The number of insertions and deletions depends on a pseudo-random number generator. Hence the length of a ciphertext will not necessarily be the same for two plaintexts. A ciphertext will have the smallest length if only deletions take place in a codeword. Conversely, if only insertions occur, the resulting ciphertext will have the longest length. This variation in length increases the security of the system (or equivalently decreases the probability of an adversary obtaining the key). It will be shown that the decryption complexity of this cryptosystem is identical to that in Section 2.1, but the security is significantly improved. This cryptosystem uses an LFSR as a pseudo-random number generator.

2.2.1 Encryption and decryption algorithms

Alice and Bob choose a block code of length n and the number of changes (insertions and deletions) to be made. Similar to the cryptosystem in Section 2.1 the codes used in this cryptosystem are constructed using the method in [44]. The number of changes should be chosen so that if there are only deletions, the number of different punctured codes is sufficiently large (e.g. 10^{35}), to ensure that an adversary cannot determine the code via an exhaustive search. For β changes, an LFSR of length $\beta\left(\log_2 \frac{n}{\beta} + 1\right)$ is used as will be described in the following encryption algorithm.

1. For a plaintext $\boldsymbol{m}$, Alice finds the corresponding codeword via $\boldsymbol{c} = \boldsymbol{m}G$, where G is the generator matrix of the $C(n, k)$ block code.

2. For β changes, Alice divides the codeword into β equal length sub-blocks. The output of an LFSR of length $\beta\left(\log_2 \frac{n}{\beta} + 1\right)$ is divided into groups of $\log_2 \frac{n}{\beta} + 1$ bits. Each group will determine the position for an insertion or deletion in the corresponding sub-block.

3. For each sub-block of $\boldsymbol{c}$, the position to insert or delete a bit is determined as follows. If the first bit in the corresponding LFSR group is zero, Alice deletes the bit in the position determined by the next $\log_2 \frac{n}{\beta}$ bits, otherwise, she insert a bit in this position. The inserted bit is determined based on the bits adjacent to the chosen position to better conceal the insertion position. If the bits to the right and left of the chosen position differ, Alice finds the lengths of the strings of identical bits to the right and left of the chosen position. She sets the value of the inserted bit to the value corresponding to the longest string. In the case that the strings to the left and right have the same length, she sets the value of the inserted bit to the modulo two sum of the string of ones. The obtained word $\boldsymbol{c}'$ is the ciphertext corresponding to $\boldsymbol{m}$.

It is shown that this method of determining the value of the inserted bit reduces the probability of revealing the position compared to random bit insertion. The decryption algorithm is as follows:

1. Bob has the same LFSR as Alice, he knows where the changes have been made to the codeword. Therefore, to decrypt a ciphertext $\boldsymbol{c}'$, he first removes the inserted bits as they carry no information.

2. Bob then attempts to find the deleted bits. This can be easily done via erasure correction as he knows the deletion positions.

3. Once Bob has found the deleted bits, he has the codeword $\boldsymbol{c}$, so he finds $\boldsymbol{m}$.

Similar to the cryptosystem in Section 2.1, this cryptosystem can also be used in a joint channel coding-cryptography scheme. A block diagram of this secure channel coding scheme is shown in Figure 2.2. In this scheme Alice finds the ciphertext corresponding to a plaintext $\boldsymbol{m}$ using the encryption algorithm and sends it to Bob. Upon receiving a word $\boldsymbol{r}$, Bob first removes the inserted bits as they carry no information. Error correction is then performed on the remaining bits as they represent a codeword $\overline{\boldsymbol{c}}$ of a punctured code from the used $C(n, k)$ block code. Once the errors have been corrected, the deleted bits can be found via erasure correction as Bob knows the deletion positions. After the deleted bits have been recovered, Bob has the codeword $\boldsymbol{c}$, so $\boldsymbol{m}$ can easily be found. By comparison to the secure channel coding scheme presented in Section 2.1, the decryption algorithm consists of only error and erasure correction. Hence the decoding complexity is the same. In the next section the parameters of the proposed cryptosystem will be analysed.

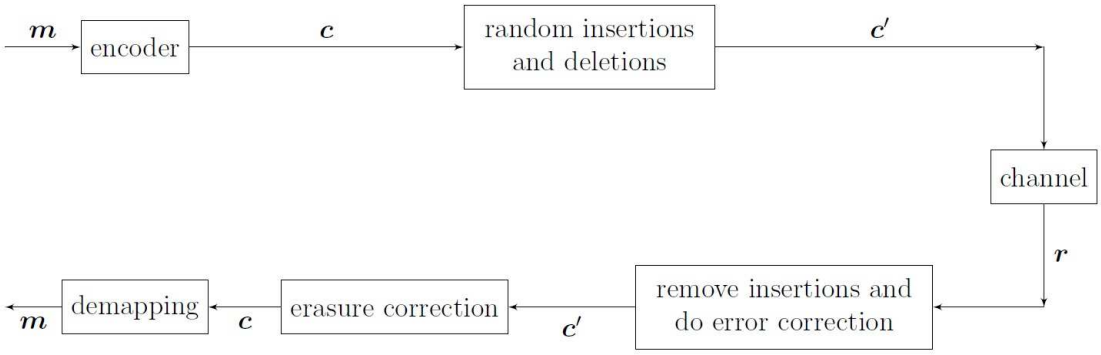

Figure 2.3: Block diagram of a secure channel coding scheme with random insertions and deletions.

2.2.2 Key size

As previously stated, the key in this cryptosystem consists of the parity check matrix of the code and the initial state of the LFSR. The LFSR structure and number of changes are not part of the key. Therefore, the size of the key in the proposed cryptosystem will be $n + \beta \left(\log_2 \frac{n}{\beta} + 1 \right)$ bits. For a $C(2048, 1536)$ block code with

$\beta = 32$ changes, the key size will be 2272 bits (≈ 2.27 kbits). Although the key size for the proposed scheme is slightly larger than that in Section 2.1.2, it will be shown that the security is much higher. Thus, the slight increase in key size is more than offset by the improved security of the proposed approach.

2.2.3 Error performance

The error performance of the code is now examined. Here the focus is on security rather than error control, hence the number of deletions and insertions in a codeword is determined based on security issues rather than error performance. In the proposed scheme, a random number of insertions and deletions are made. Insertions will not effect the error performance as the inserted bits are discarded at the receiver. Hence if only insertions are done the error performance will remain unchanged. Therefore, the worst case error performance occurs when only deletions are made. Figure 2.4 presents the error performance of a $C(2048, 1536)$ block code with 32 deletions and a random code with the same length and rate on an AWGN channel. This shows that even in the worst case, the used code outperforms a random code.

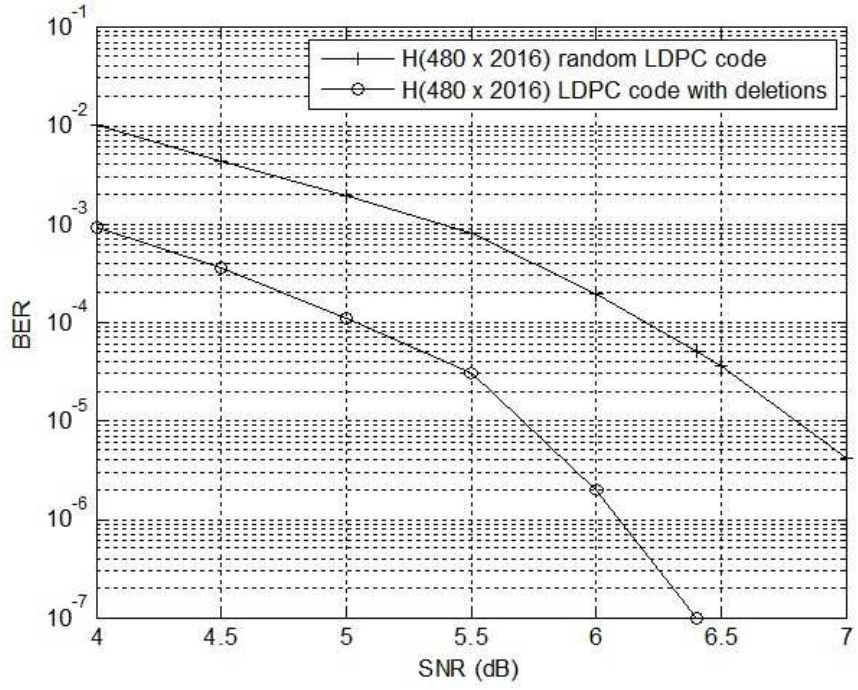

Figure 2.4: BER performance of a C(2048,1536) LDPC code with 32 deletions and a random C(2016,1536) LDPC code on an AWGN channel.

2.2.4 Security analysis

To evaluate the security of the proposed system, the approach introduced in Section 2.1.4 will be employed. As the same code construction method in [44] is used, it is obvious that finding the code via a brute force attack is hopeless. To put the cryptosystem at risk the code will be made public and finding the remainder of the key is examined. It will be shown that the proposed system has good security even with this assumption, which can be considered worst case.

In general, suppose that the block code has length n. Let β_i and β_d denote the number of insertions and deletions, respectively. Hence a codeword is divided into $\beta = \beta_i + \beta_d$ sub-blocks of length $\alpha = \frac{n}{\beta}$. The probability of an adversary guessing the correct LFSR state is $\frac{1}{\binom{\beta}{\beta_i}} \frac{1}{\alpha^{\beta_i}} \frac{1}{\alpha^{\beta_d}}$. For a $C(2048, 1536)$ block code, with $\beta_i = \beta_d = 16$, the probability of guessing the correct LFSR state is 2^{-221}.

Similar to Section 2.1.4 the best case for an adversary is if the codeword is a string of alternating 0's and 1's, i.e. $1010101010\ldots$ or its complement. In this case the probability of correctly determining the LFSR state increases. One method to prevent this problem is to eliminate codes for which these codewords exist. Conversely, if one or both of these codewords exist, the modification used in Section 2.1.4 can be used. If an insertion is to be done between a 0 and a 1 in a sub-block, insert the pair of bits 10. Similarly, if an insertion is to be done between a 1 and a 0 in a sub-block, insert the pair 01. If a zero is to be deleted in a sub-block, also delete the 1 to its right, and if a 1 is to be deleted, also delete the 0 to its left. With this modification the probability of guessing the correct insertion position in a sub-block is $\frac{1}{\alpha}$, where $\alpha = \frac{n}{\beta}$. The probability of guessing the correct deletion position in a sub-block is also $\frac{1}{\alpha}$. Further, the probability of an insertion in a sub-block is $\frac{1}{\binom{\beta}{\beta_i}}$, where β_i is the number of insertions, and the probability of a deletion in a sub-block is $\frac{1}{\binom{\beta}{\beta_d}}$, where β_d is the number of deletions. Therefore, the probability of guessing the correct insertion and deletion positions is

$$\frac{1}{\alpha^{\beta_i}} \times \frac{1}{\alpha^{\beta_d}} \times \frac{1}{\binom{\beta}{\beta_i}}.$$

For a $C(2048, 1536)$ block code, with $\beta_i = \beta_d = 16$, the probability of guessing the correct insertion and deletion positions is

$$\frac{1}{2^{96}} \times \frac{1}{2^{96}} \times \frac{1}{\binom{32}{16}} \approx 2^{-221}.$$

If the number of insertions and deletions are not known, all possible values for β_d and β_i must be considered, which lowers this probability

With the modification above, the best situation for an adversary is when the codeword is similar to a string consisting of altering pairs of 11 and 00, that is $11001100110011001100\ldots$, or its complement. However, this attack will have a very low probability of success. In general, the probability of guessing a correct insertion and deletion position is $\frac{1}{3}$ and $\frac{1}{2}$, respectively. Hence if β_i insertions and β_d deletions have occurred, the probability of guessing the correct insertion and deletion positions is $\left(\frac{1}{2}\right)^{\beta_d} \times \left(\frac{1}{3}\right)^{\beta_i}$. Thus, the highest probability for an adversary to guess the LFSR state correctly is $\left(\frac{1}{2}\right)^{\beta_d}$, which occurs when deletions are made in every sub-block. For a $C(2048, 1536)$ block code with $\beta_d = 32$, this probability is 2^{-32}. It is obvious that if only insertions take place, the adversary has the lowest probability of guessing the correct insertion positions. For a $C(2048, 1536)$ block code with $\beta_i = 32$, this probability is 3^{-32}. If there is no codeword of this form in the code, the probability of guessing the correct insertion and deletion positions will be much smaller.

Although the probabilities obtained above are small, a slight change in the LFSR can make them even smaller, as shown below. To reduce the probability of guessing the correct insertion and deletion positions, the LFSR can be used to determine the length of each sub-block. Suppose that β changes are to be made to each codeword. An LFSR of length $l = (2\beta - 1)\log_2 \frac{n}{\beta} + \beta$ bits is used, and its output is divided into $\beta - 1$ groups each consisting of $2\log_2 \frac{n}{\beta} + 1$ bits and a group of $\log_2 \frac{n}{\beta} + 1$ bits at the end. In the $\beta - 1$ groups, the first $\log_2 \frac{n}{\beta}$ bits and the second $\log_2 \frac{n}{\beta} + 1$ bits represent the length of the corresponding sub-block in the codeword and the position of the insertion or deletion, respectively. The last $\log_2 \frac{n}{\beta} + 1$ bits represent the insertion or deletion position in the remaining bits that make up the last sub-block in the codeword.

It is obvious that with this method, the probability of guessing the correct LFSR state will be much lower than with fixed length sub-blocks. Considering all possible situations and determining the probability of obtaining the correct LFSR state is very complex. Hence a simplified case is analysed. Suppose that the changes in a codeword $\boldsymbol{c}$ corresponding to a plaintext $\boldsymbol{m}$ are equally distant from each other, i.e. the number of bits between each change (regardless of whether it is an insertion or deletion), is the same. Then the probability of guessing the correct LFSR state becomes a case of guessing the length of the corresponding sub-block length and the position in which a change is made. Let i denote the number of bits between consecutive changes. As the

maximum sub-block length is $\alpha = \frac{n}{\beta}$, it must be that $i \leq \alpha$. Therefore, a sub-block will have length between $2i - \alpha$ and α. Each length has a probability of $\frac{1}{2(\alpha - i)}$ of occurring. Of the possible lengths, each can be generated by $\log_2 \frac{n}{\beta}$ bits in the LFSR with a probability of $\frac{1}{\alpha} \times 2(\alpha - i)$. Hence the probability of guessing the correct state of the $2 \log_2 \frac{n}{\beta}$ bits corresponding to a sub-block is $\frac{1}{\alpha}$. Since there are β sub-blocks, the probability of guessing the correct LFSR state is $\left(\frac{1}{\alpha}\right)^{\beta}$. For a $C(2048, 1536)$ block code with $\beta = 32$, this probability is $\left(\frac{1}{64}\right)^{32} = 2^{-192}$. As previously stated, this is a simplified case, as in general the changes in each sub-block will not be equally spaced. This will result in a significantly lower probability of guessing the correct sub-block length and the insertion/deletion positions. This simplified case has a success rate equal to the strongest case for the secure channel coding scheme presented in Section 2.1 and thus including bit insertions provides a substantial improvement in security.

2.3 Conclusion

In this chapter, two proposed improvements to symmetric key code based cryptosystems were explained. The encryption and decryption algorithms, key sizes and their security analysis were provided. It was shown how simple insertions and deletions can significantly increase the security. Throughout the security analysis it was shown that having a public code will not necessarily compromise the security of a code based cryptosystem. This gives the idea that code need not be kept secret, hence resulting in possible use of high rate codes and significant reduction in key size. In the next chapter two symmetric key cryptosystems will be proposed that have public codes.

Chapter 3

Symmetric Key Code Based Cryptosystems with Public Codes

In this chapter, two symmetric key cryptosystems with public codes are proposed. The first is based on randomly flipping an arbitrary number of bits in the codeword corresponding to a plaintext and randomly inserting and deleting bits from it. The second is based on random bit flipping and block deletions. The security of these cryptosystems is analysed. It is shown that the probability of an adversary breaking them is negligible.

3.1 Code based encryption via random bit flipping, bit deletion, and insertion

In this section, a new symmetric key cryptosystem is presented which employs a public code. This cryptosystem uses random flipping of codeword bits and random insertions and deletions similar to the one in Section 2.2. Two random number generators are used to determine which bits should be flipped and the insertion/deletion positions. It is shown that this cryptosystem is more secure than similar code based cryptosystems, while having a smaller key size. In fact, the key size is comparable to that of many well known symmetric key cryptosystems, which is a significant improvement over other code based cryptosystems. Note that Alice and Bob are not limited to a specific class of codes and can use any code that they desire. This will be the case for the rest of the dissertation. Being able to choose any code is an advantage over previously proposed code based cryptosystems that use codes with specific structures to decrease the key

size. The encryption, decryption algorithms along with an analysis of the key size and security is provided.

3.1.1 Encryption algorithm

As mentioned previously, this cryptosystem is based on randomly flipping, inserting and deleting bits of a codeword. Two random number generators are used to determine which bits should be flipped and the insertion and deletion positions. The key consists of the states of these two random number generators. Linear feedback shift registers (LFSRs) are used as random number generators as with most code based cryptosystems in the literature, but other random number generators can be employed. The states of the LFSRs are used to generate the random numbers as described below.

The encryption algorithm is as follows.

1. Alice and Bob choose a $C(n, k)$ block code and decide how many sub-blocks a codeword is divided into, denoted by β. The desired levels of security and reliability provide upper and lower bounds for β, as will be discussed shortly. Note that β is not secret.

2. Let $\boldsymbol{m}$ and $\boldsymbol{c}$ denote a plaintext and the corresponding codeword, respectively. The codeword is obtained from the plaintext as $\boldsymbol{c} = \boldsymbol{m}G$ where G is a generator matrix for the code. Alice finds $\boldsymbol{c}_e = \boldsymbol{c} + \boldsymbol{e}$ where $\boldsymbol{e}$ is a random error vector obtained from $\boldsymbol{e} = \boldsymbol{s} \times H^{-1}$. The state of an $n-k$ bit LFSR determines $\boldsymbol{s}$, and H^{-1} is a right inverse matrix of the parity check matrix of the $C(n, k)$ block code. If $\boldsymbol{e}$ has weight less than the error correcting capability of the code, Alice discards it and chooses another $\boldsymbol{s}$.

3. After obtaining $\boldsymbol{c}_e$, it is divided into β equal length sub-blocks. An LFSR of length $\beta(\lfloor \log_2 \frac{n}{\beta} \rfloor + 1)$ bits determines in which sub-blocks of $\boldsymbol{c}_e$ a random bit will be inserted and from which a bit will be deleted. How to determine the insertion or deletion position and the value of the inserted bit is identical to the third step of the encryption algorithm of given in Section 2.2.1. The result after inserting and deleting bits, denoted by $\boldsymbol{c}'$, is the ciphertext corresponding to $\boldsymbol{m}$, and this is sent to Bob.

If the value of β is chosen such that $n - k + \beta \geq 110$, then the probability of randomly obtaining the insertion/deletion positions along with the random error

vector is at most 2^{-110}. This lower bound ensures that an adversary will not be able to find the key in a reasonable amount of time. In addition, if the code is also to provide reliable communications over a noisy channel, then β should be chosen such that $\beta \leq d_{min} - 2t$, where d_{min} denotes the minimum distance of the code. This upper bound ensures that the code will be able to correct up to t errors introduced by the channel.

As explained in the encryption algorithm, the state of an $n-k$ bit LFSR determines a binary vector $\boldsymbol{s}$. This vector can be treated as a random syndrome which is then used to find an n bit random error vector $\boldsymbol{e} = \boldsymbol{s} \times H^{-1}$ [46]. The main advantage of this approach is that, with high probability, the generated random error vectors have weights exceeding the error correcting capability of the code. Hence they cannot be removed by any error correction techniques. As mentioned previously, H^{-1} is a right inverse of the parity check matrix of C, i.e. HH^{-1} is the $(n-k) \times (n-k)$ identity matrix. However, H^{-1} is not unique, so many error vectors can be obtained from a random syndrome. Consider the following example.

Example 1. The parity check matrix of a $(7, 4)$ code is given by

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Two right inverses for H are

$$H_1^{-1} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \quad H_2^{-1} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

There are only 8 possible choices for $\boldsymbol{s}$ and hence 8 random error vectors that can be used. From $\boldsymbol{s}(H_1^{-1})^T$ these vectors are

$$0000000, 1011101, 1010000, 0111111, 0001101, 1100010, 1101111, 0110010.$$

Note that the vectors other than the all zero vector have a weight greater than one which is the error correcting capability of the code.

To ensure Alice and Bob use the same matrix, a publicly known algorithm to find H^{-1} can be used. With this approach, Bob and Alice will have the same right inverse matrix of H and thus will obtain the same random error vectors.

3.1.2 Decryption algorithm

The decryption algorithm is as follows.

1. To decrypt a ciphertext, Bob discards the inserted bits as they carry no information. Bob can find these bits because the LFSR state which determined the insertion positions is part of the key. Let $\boldsymbol{r}'$ denote the resulting vector.

2. It is obvious that $\boldsymbol{r}'$ is $\boldsymbol{c}_e$ with bits punctured at the positions selected by the LFSR. Before finding the punctured bits, Bob must remove the random errors introduced by Alice. To remove these errors, Bob next finds the random error vector $\boldsymbol{e}$. This is obtained from the LFSR used to determine $\boldsymbol{s}$ and the right inverse of the parity check matrix. Since bits of $\boldsymbol{c}_e$ were deleted by the sender, the same bits should be deleted from $\boldsymbol{e}$. Hence by deleting the bits of $\boldsymbol{e}$ at the same positions that bits of $\boldsymbol{c}_e$ were deleted, Bob obtains $\boldsymbol{e}'$. He then constructs $\boldsymbol{r} = \boldsymbol{r}' + \boldsymbol{e}'$.

3. The vector $\boldsymbol{r}$ is a codeword in a punctured code from C, with the punctured bits being those deleted at the positions chosen by the LFSR. As Bob knows the positions of these bits, erasure correction can be used to find them and obtain $\boldsymbol{c}$. Having $\boldsymbol{c}$, he finds the plaintext $\boldsymbol{m}$.

The block diagram in Figure 3.1 illustrates a secure channel coding scheme using the proposed cryptosystem. To send a plaintext $\boldsymbol{m}$ to Bob, Alice encrypts it using the encryption algorithm and sends the resulting ciphertext $\boldsymbol{c}'$ to Bob. Upon receiving a word from the channel, Bob decrypts it using the corresponding decryption algorithm. However, as errors from the channel may be present, they must be corrected before executing step 3 of the algorithm. This can easily be done as the result of step 2 is a codeword in a punctured code with errors, so error correction can be employed to remove these errors. Bob can then proceed with step 3 of the decryption algorithm to obtain the plaintext $\boldsymbol{m}$.

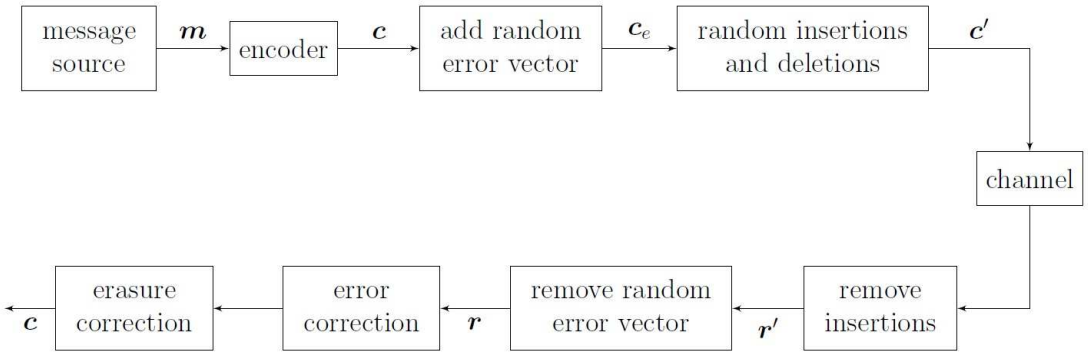

Figure 3.1: Block diagram of a secure channel coding scheme based on random insertions, deletions and random errors.

3.1.3 Security analysis

In this section, the security of the proposed cryptosystem is analyzed. It will be shown that even though the code is public, the system has excellent security. Many code based cryptosystems employ various bit flipping techniques to conceal the code structure. Some attacks attempt to recover the flipped bits (e.g. the Stern [11], RN [38], Struik-Tilberg [39], and Barbero-Ytrehus [46] attacks). However these attacks are not a threat to the proposed cryptosystem since the error vectors used here have an average weight of $n/2$ where n is the code length. For the same reason information set decoding attacks will not be effective on this cryptosystem. Note that the total number of random error vectors in the proposed scheme is 2^{n-k}. Hence if $n-k$ is a small value the number of random error vectors is small. This makes the cryptosystem vulnerable to a brute force attack. Hence to avoid this situation it is recommended that $n-k$ be chosen such that the total number of error vectors is large.

To analyze the security of the proposed cryptosystem, the success rate of finding the insertion and deletion positions and the random error vector by observing a plaintext and its corresponding ciphertext is considered. Based on the encryption algorithm, similar to Sections 2.1.4 and 2.2.4 it is easy to see that the best situation for an adversary is when $\boldsymbol{c}_e$ is an alternating string of zeros and ones. In this case, if any bit is deleted in a sub-block, a pair of identical adjacent bits will appear in $\boldsymbol{c}'$. Hence by comparing $\boldsymbol{c}_e$ and $\boldsymbol{c}'$, it is easy to determine where the deletion positions are. On the other hand, if a bit is inserted in a string of $0101010101\ldots$ or $1010101010\ldots$, a pair of ones will appear in the ciphertext. Hence an adversary can determine where the insertion is located.

To avoid these situations, the selected bit in the string can be deleted along with a bit adjacent to it, so then a 10 or 01 pair will always be deleted. This will result in a string of alternating zeros and ones similar to but shorter than $\boldsymbol{c}_e$. In the case of an insertion, a pair of bits should be inserted in the selected position based on the adjacent bits. If the bit on the left of the selected position is a zero, insert 10, otherwise insert 01. In either case, the resulting ciphertext will be a string of alternating zeros and ones longer than $\boldsymbol{c}_e$. From an adversary perspective, the insertion/deletion position could be anywhere in $\boldsymbol{c}_e$. Hence the probability of an adversary correctly determining the insertion/deletion position is $(1/\lfloor \frac{n}{\beta} \rfloor)^{\beta}$. Note that there can exist only two random error vectors which can change a codeword $\boldsymbol{c}$ to a string of alternating zeros and ones. The probability of one of these random error vectors appearing is at most $2^{-(n-k)}$. Hence the probability of this situation occurring and an adversary successfully obtaining the key is $2^{-(n-k-1)} \times (1/\lfloor \frac{n}{\beta} \rfloor)^{\beta}$. For a $C(2048, 1826)$ code with $\beta = 32$, this probability is approximately 2^{-413}.

With the above modification, the best case for an adversary is if $\boldsymbol{c}_e$ is an alternating string of 11 and 00 pairs. In this case, if a deletion occurs in a sub-block, a single 0 or 1 will appear in $\boldsymbol{c}'$. Using a similar approach to that above, it can be concluded that the highest probability of guessing the correct positions is if only deletions occur, which is $2^{-\beta}$. For a $C(2048, 1826)$ code with $\beta = 32$, an adversary would at best be successful with a probability of 2^{-32}. The probability of $\boldsymbol{c}_e = 1100110011001100\ldots$ occurring is at most $2^{-(n-k)}$, as only one random error vector exists which can change a codeword $\boldsymbol{c}$ to $\boldsymbol{c}_e$. Thus, with a $C(2048, 1826)$ code and $\beta = 32$, an adversary will be successful in obtaining the key with probability at most $2^{-222} \times 2^{-32} = 2^{-254}$. This probability is significantly smaller than the corresponding value of 2^{-32} in Sections 2.1.4 and 2.2.4. Therefore the cryptosystem has a higher level of security than those in Sections 2.1 and 2.2.

Although the proposed cryptosystem with the parameters considered in this section has excellent security, the success rate of an adversary obtaining the key can be further decreased by using variable length sub-blocks in a codeword at the cost of a slight increase in the key size. This method was employed in Section 2.2.4 and provided a significant improvement in security. With this approach, an LFSR of length $(2\beta - 1)\lfloor \log_2 \frac{n}{\beta} \rfloor + \beta$ can be used to determine the sub-block lengths and insertion/deletion positions. For this purpose, the LFSR output is divided into $\beta - 1$ groups of length $2\lfloor \log_2 \frac{n}{\beta} \rfloor + 1$ bits and a final group of $\lfloor \log_2 \frac{n}{\beta} \rfloor$ bits. In the $\beta - 1$ groups, the first $\lfloor \log_2 \frac{n}{\beta} \rfloor$ bits determine the sub-block length, and the following $\lfloor \log_2 \frac{n}{\beta} \rfloor + 1$ bits

determine the insertion/deletion position. The last $\lfloor \log_2 \frac{n}{\beta} \rfloor + 1$ bits determine the insertion/deletion position in the final sub-block. It was shown in Section 2.2.4 that this approach decreases the probability of an adversary determining the insertion/deletion positions.

In a simple case where consecutive insertion/deletion positions are equidistant, the probability of finding the insertion/deletion positions is $(1/\lfloor \frac{n}{\beta} \rfloor)^{\beta}$. To break the proposed cryptosystem, the random error vector must also be obtained. As all random error vectors are equally likely, the probability of finding the correct one is $2^{-(n-k)}$. Therefore, the probability of obtaining the key and breaking the cryptosystem is $2^{-(n-k)} \times (1/\lfloor \frac{n}{\beta} \rfloor)^{\beta}$. For a $C(2048, 1826)$ code with $\beta = 32$, this probability is approximately 2^{-414}, which is smaller than the corresponding value in Section 2.2. However, in general the insertion/deletions positions will not be equidistant, so the success rate for an adversary will be much smaller than the values given in this section.

3.1.4 Key size

The key size of the proposed cryptosystem is found and compared with that of other code based cryptosystems. As mentioned previously, the key of the proposed cryptosystem consists of the states of the two random number generators used to determine the random syndrome $\boldsymbol{s}$ and the insertion/deletion positions. In this cryptosystem, LFSRs of lengths $n-k$ and $\beta(\lfloor \log_2 \frac{n}{\beta} \rfloor + 1)$ are used as the random number generators for illustration purposes, where n, k and β denote the length and dimension of the code, and the number of changes, respectively. Thus as with other cryptosystems that employ LFSRs, the key only consists of their initial states. Hence the key size is $n-k+\beta(\lfloor \log_2 \frac{n}{\beta} \rfloor + 1)$ bits. For a 254 bit security level (i.e. breaking the cryptosystem requires 2^{254} operations) the key size of this cryptosystem is 446 bits.

In the next section another symmetric key code based cryptosystem is presented. Similar to the cryptosystem presented in this section, it has a public code.

3.2 Code based encryption via random bit flipping and block deletion

In this section a new symmetric key cryptosystem is introduced. This cryptosystem is a variant of the one presented in Section 3.1. In this cryptosystem bits of a codeword

are randomly flipped and blocks of the resulting vector are deleted. Similar to previously proposed encryption schemes in this dissertation, LFSRs are used to generate random error vectors and determine which blocks in an erroneous codeword should be deleted.

3.2.1 Encryption algorithm

The encryption algorithm is as follows.

1. Alice and Bob agree on an error correcting code $C(n, k, d)$ where n, k and d denote the length, dimension and minimum Hamming distance, respectively. They also agree on how many groups a codeword should be divided into, denoted by β, such that $\frac{2t}{\beta} \geq 2$, where $t = \lfloor (d-1)/2 \rfloor$ is the error correcting capability of C. Note that β is not secret.

2. For a plaintext $\boldsymbol{m}$, Alice finds the corresponding codeword $\boldsymbol{c} = \boldsymbol{m}G$, where G is the generator matrix for C. She then uses the state of an $n-k$ bit LFSR to determine a random syndrome $\boldsymbol{s}$. This syndrome is used to obtain a random error vector $\boldsymbol{e} = \boldsymbol{s} \times (H^{-1})^T$, where H^{-1} is a right inverse of the parity check matrix of C. If $\boldsymbol{e}$ has weight less than the error correcting capability of the code, Alice discards it and chooses another $\boldsymbol{s}$.

3. Alice next calculates $\boldsymbol{c}_e = \boldsymbol{c} + \boldsymbol{e}$. She then divides $\boldsymbol{c}_e$ into β groups and using an LFSR of length $\beta \lfloor \log_2 \frac{n}{\beta} \rfloor$ finds the deletion positions in each group. This is done by dividing the LFSR into β groups of $\lfloor \log_2 \frac{n}{\beta} \rfloor$ bits. The decimal equivalent of each group of $\lfloor \log_2 \frac{n}{\beta} \rfloor$ bits represents the position in the associated group in $\boldsymbol{c}_e$ from which bits are to be deleted. Alice deletes $\frac{2t}{\beta}$ consecutive bits on the right of the selected position in each group in $\boldsymbol{c}_e$. However, if some of these bits belong to the adjacent group, Alice deletes $\frac{2t}{\beta}$ consecutive bits to the left of the chosen position. This approach prevents the block deletions from overlapping, thus providing a higher level of security. The resulting punctured codeword $\boldsymbol{c}'$ is the ciphertext corresponding to $\boldsymbol{m}$.

The following example illustrates how block deletions are done in groups of a codeword with errors.

Example 2. Suppose Alice and Bob have agreed to use a code $C(32, 12, 9)$ and $\beta = 4$. Hence for every plaintext, blocks of two bits from each group in the codeword

containing errors are deleted. Let $\boldsymbol{c}_e$ for a plaintext $\boldsymbol{m}$ be

$$\boldsymbol{c}_e = 1001001101111100001101001010101001.$$

Alice divides $\boldsymbol{c}_e$ into four groups, namely 10010011, 01111100, 00110100, and 10101001. The first position in each group is defined to be on the left of the first bit. Now suppose the LFSR has chosen position 7 in the first, position 8 in the second, position 2 in the third, and position 3 in the fourth group. Hence, based on the encryption algorithm, bits five and six from the first, bits two and three from the third, and bits three and four from the last group should be deleted, but for the second group, bits four and five are deleted to avoid deleting a bit from the third group. Therefore, the ciphertext corresponding to $\boldsymbol{m}$ is $\boldsymbol{c}' = 1001000111100101001010001$.

Note that if the code must also provide reliable communication over a noisy channel as well as security, β should be chosen such that $\beta \leq d-2e$ where e is the number of channel errors to be corrected.

3.2.2 Decryption algorithm

The decryption algorithm is as follows.

1. To decrypt a ciphertext $\boldsymbol{c}'$, Bob must first remove the random error vector that Alice introduced. This can easily be done since he has the same LFSR that Alice used to generate $\boldsymbol{s}$. Hence he can find the same $\boldsymbol{e}$ that Alice used. Next Bob has to delete blocks of $\boldsymbol{e}$ at the same deletion positions employed by Alice. This can also be done easily since he has the same LFSR that chose the deletion positions. Let $\boldsymbol{e}'$ denote the vector obtained by deleting the chosen blocks of $\boldsymbol{e}$.

2. Bob finds $\boldsymbol{c}' + \boldsymbol{e}'$ which is a punctured codeword in C with the punctured positions determined by the LFSR. Hence the deleted bits can be retrieved by erasure correction.

3. Once the deleted bits have been recovered, Bob has the codeword $\boldsymbol{c}$ corresponding to the plaintext $\boldsymbol{m}$, so he finds $\boldsymbol{m}$.

Note that the decryption algorithm of the proposed cryptosystem only consists of flipping bits and erasure correction as in Section 3.1.Hence the decryption complexity is similar.

The proposed cryptosystem is now considered in a secure channel coding scheme. A block diagram of this system using the proposed cryptosystem is illustrated in Figure 1. To send a message $\boldsymbol{m}$ to Bob, Alice uses the encryption algorithm to find the corresponding ciphertext $\boldsymbol{c}'$. She then sends $\boldsymbol{c}'$ to Bob. Once Bob receives a word $\boldsymbol{r}$, the first step of the decryption algorithm is used to remove the random error vector introduced by Alice to obtain $\boldsymbol{r}'$. Before proceeding to the second step, Bob removes the errors introduced by the channel. It is obvious that $\boldsymbol{r}'$ is a codeword in a punctured code obtained from C that contains errors. Hence error correction can be used to remove these errors. Having removed the channel errors, Bob can proceed to steps 2 and 3 of the decryption algorithm to find the plaintext $\boldsymbol{m}$ that Alice sent. Note that the random error vector $\boldsymbol{e}$ is removed at the receiver, hence it has no effect on the BER performance of the code.

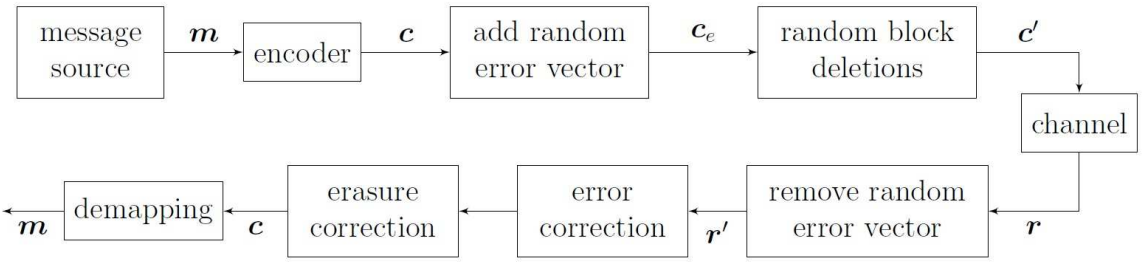

Figure 3.2: Block diagram of a secure channel coding scheme using random bit flipping and block deletions.

In this section the security and key size of the proposed cryptosystem are analysed.

3.2.3 Security analysis

In this section the security of the proposed cryptosystem is examined. Code based cryptosystems have been subjected to many attacks. The goal of attacks such as the RN attack [38] and Struik-Tilberg attack [39] is to find the permutation and scrambling matrices. Since these matrices are not used in the proposed cryptosystem, these attacks are irrelevant. There are also attacks that try to find the random error vector such as the Stern attack [11]. However, these attacks only find random error vectors with weights within the error correcting capability of the code. In the proposed cryptosystem, random error vectors with average weight $\frac{n}{2}$ are used. Hence this attack can be considered ineffective against the proposed cryptosystem. Similar to Section 3.1.3, note that the total number of random error vectors in the proposed scheme is

2^{n-k}. Hence if $n-k$ is a small value the number of random error vectors is small. This makes the cryptosystem vulnerable to a brute force attack. Hence to avoid this situation it is recommended that $n-k$ be chosen such that the total number of error vectors is large.

Another type of attack is information set decoding [9], [42]. They are based on the fact that out of n bits of a ciphertext, only k error free bits are required to reconstruct the message. A systematic approach to finding k error free bits in a codeword with errors was provided in [42]. However, with the proposed cryptosystem the ciphertext will differ significantly from the corresponding codeword as random error vectors are used which have an average weight of $\frac{n}{2}$. Hence there is no means of finding k error free bits in the ciphertext, so the proposed cryptosystem is immune to this type of attack.

To analyse the security of the cryptosystem, the success rate of a possible attack will be derived and shown to be very small. Suppose that an adversary possesses a plaintext-ciphertext pair $(\boldsymbol{m}, \boldsymbol{c}')$. The codeword $\boldsymbol{c}$ corresponding to $\boldsymbol{m}$ can easily be found as the code is public. Hence for a general pair $(\boldsymbol{m}, \boldsymbol{c}')$ (or equivalently $(\boldsymbol{c}, \boldsymbol{c}')$) an adversary can guess the deletion positions in each group of $\boldsymbol{c}$. The probability of finding the correct deletion positions in a group is $\frac{1}{n/\beta}$. Since there are β groups in $\boldsymbol{c}$, the probability of finding the correct deletion positions in a single guess is $\left(\frac{1}{n/\beta}\right)^{-\beta}$. Having guessed the deletion positions, the random error vector has to be found. It is obvious that each bit in the deleted blocks in the random error vector can either be zero or one. Since $2t$ bits of a codeword (and consequently a random error vector) are deleted, there are 2^{2t} possible random error vectors that could have been used by Alice. Hence the probability of guessing the correct one is 2^{-2t}. The probability of finding the correct deletion positions and random error vector is then $\left(\frac{1}{n/\beta}\right)^{-\beta} \times 2^{-2t}$. For a code $C(4096, 3900, 41)$ and $\beta = 38$, this probability is 2^{-296} which is extremely small. Thus it can be concluded that the cryptosystem is secure against this attack.

The best situation for an adversary is if the random error vector $\boldsymbol{e}$ has weight $2t$ and the deletion positions coincide with non-zero bits in $\boldsymbol{e}$. In this case, the deletion positions are easily found by comparing $\boldsymbol{c}$ and $\boldsymbol{c}'$, and simple erasure correction will determine the deleted bits. The probability of such an event is at most $\left(\frac{n}{\beta}\right)^{-\beta}$. For the code $C(4096, 3900, 41)$ and $\beta = 38$, this probability is $\approx 2^{-256}$, which is very small. Hence this case is very unlikely to occur. Therefore, there is no need to discard random error vectors with low weight. Although the success rate of the proposed attack is

very low, modifications can be made to further decrease the probability of breaking the system. For instance, the groups in a codeword can have variable lengths. In addition, the encryption algorithm can be modified so that the deleted blocks are not of equal size. This improvement in security comes at a cost of an increase in key size as the pseudo-random number generator used for determining the deletion positions must also provide the lengths of the groups and/or the deletion block lengths.

3.2.4 Key size

In this section, the key size of the proposed cryptosystem is analysed. As previously mentioned, two pseudo-random number generators are used to determine a random syndrome and the deletion positions. Thus in this paper, the key consists of the initial states of the LFSRs used as pseudo-random number generators, so the key size is $n - k + \beta\lfloor \log_2 \frac{n}{\beta} \rfloor$ bits. For example, if a code $C(4096, 3900, 41)$ and $\beta = 38$ are chosen, the key size will be 434 bits. This key size is smaller than that of similar code based cryptosystems, i.e. based on random bit flipping and deletions. Table 3.1 gives the key sizes of code based cryptosystems similar to that proposed here. For a fair comparison, the key sizes are such that the computational complexity of the best known attack on each cryptosystem is $O(2^{254})$.

Table 3.1: Comparison of Key Sizes For Code Based Cryptosystems

Scheme	Key size
Section 2.1 random deletion with private code	1254 bits
Section 2.2 random deletion and insertion with private code	1300 bits
Section 3.1 random bit flipping and deletion and insertion with public code	446 bits
Section 3.2 random bit flipping and block deletion with public code	434 bits

3.3 Conclusion

In this chapter, two new symmetric key code based cryptosystems were introduced. The first was based on randomly flipping, deleting, and inserting bits into a codeword.

This cryptosystem had a public code. The encryption and decryption algorithms were given. The application of the encryption scheme in a secure channel coding scheme was explained. It was shown that the using simple bit flipping, deletion and insertion results in a high level of security. The key size was analysed and shown to have a smaller size than those in Chapter 2. Specifically, it was shown how having a public code can significantly decrease the key size.

The second cryptosystem was based on random bit flipping and block deletion of a codeword corresponding to a plaintext. The encryption and decryption algorithms were presented. The use of the proposed cryptosystem in a secure channel coding scheme was also explained. The security was derived by showing the success rate of an adversary being able to find the random error vector and the deletion positions. This probability is smaller than that for previous code based cryptosystems. The key size of the proposed cryptosystem was derived. It was shown that for an attack with computational complexity $O(2^{254})$, the proposed cryptosystem has a smaller key size than cryptosystems presented thus far while having the same encryption/decryption complexity. This key size is comparable to many well-known symmetric-key cryptosystems.

In the next chapter a new technique will be used to construct a symmetric key code based cryptosystem. This cryptosystem will be based on randomly interleaving two erroneous codewords to obtain a plaintext. It will be shown that this technique further decreases the key size.

Chapter 4

Code Based Cryptography with Random Interleaving

In this chapter, a new symmetric key code based cryptosystem with a public code is presented. This cryptosystem is based on randomly flipping the bits of two codewords and randomly interleaving them to obtain a ciphertext. It is shown that this simple technique can result in smaller key sizes than previously obtained in Chapters 2 and 3. It is also shown that the system has a high level of security. Note that similar to the encryption schemes in Chapter 3, Alice and Bob can choose any code they desire.

4.1 Encryption algorithm

In this section, the encryption algorithm of the encryption scheme is presented. It employs an error correcting code denoted by $C(n, k)$, where n and k are the length and dimension of the code, respectively. For convenience, an LFSR is used as a pseudo-random number generator, but other types of random number generators can be employed. The LFSR determines where the sub-blocks from one codeword are inserted into the sub-blocks of the other codeword. The key consists of the LFSR initial state.

The steps of the encryption algorithm are as follows.

1. Alice and Bob agree on $C(n, k)$ and how many sub-blocks a codeword is to be divided into, denoted by β. Note that β is not secret.

2. Let $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ denote two plaintexts. Alice constructs the codewords $\boldsymbol{c}_1$ and

$\boldsymbol{c}_2$ corresponding to $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$, respectively, according to

$$\boldsymbol{c} = \boldsymbol{m}G,$$

where G is the generator matrix of C.

3. Alice randomly chooses two non-zero $n-k$ bit syndromes $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$. Two random n bit error vectors $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are obtained using

$$\boldsymbol{e}_i = \boldsymbol{s}_i \times (H^{-1})^T, i = 1, 2$$

where H^{-1} is a right inverse matrix of the parity check matrix, H, of C. If $\boldsymbol{e}$ has weight less than the error correcting capability of the code, Alice discards it and chooses another $\boldsymbol{s}$.

4. Alice then constructs the two codewords containing errors by adding the error vectors to the codewords $\boldsymbol{c}_{1,e} = \boldsymbol{c}_1 + \boldsymbol{e}_1$ and $\boldsymbol{c}_{2,e} = \boldsymbol{c}_2 + \boldsymbol{e}_2$. Next, $\boldsymbol{c}_{1,e}$ and $\boldsymbol{c}_{2,e}$ are each divided into β sub-blocks. Using an LFSR of length $\beta\lfloor \log_2 \frac{n}{\beta} \rfloor$, the decimal equivalent of each group of $\lfloor \log_2 \frac{n}{\beta} \rfloor$ bits represents the insertion position of each sub-block of $\boldsymbol{c}_{2,e}$ into the corresponding sub-block of $\boldsymbol{c}_{1,e}$. The resulting string of length $2n$, denoted by $\boldsymbol{c}'$, is the ciphertext corresponding to $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$.

Example 3. Suppose Alice and Bob have agreed on $\beta = 4$, and $\boldsymbol{c}_{1,e}$ and $\boldsymbol{c}_{2,e}$ corresponding to the plaintexts $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ are

$$\boldsymbol{c}_{1,e} = 1011000011011100 \text{ and } \boldsymbol{c}_{2,e} = 1111011010110011,$$

respectively. Each erroneous codeword is divided into four sub-blocks of length four bits. Suppose the LFSR state for determining the insertion positions is 10011100. Assuming 00 denotes the position left of the first bit in a sub-block, the first, second, third, and fourth sub-blocks of $c_{2,e}$ are inserted between the second and third bits of the first, the first and second bits of the second, the third and fourth bits of the third, and before the first bit of the fourth sub-block in $\boldsymbol{c}_{1,e}$, respectively. Table 1 illustrates the interleaving process along with the sub-blocks of $\boldsymbol{c}_{1,e}$ and $\boldsymbol{c}_{2,e}$, and the corresponding LFSR state. Therefore, the ciphertext corresponding to $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ is

$$\boldsymbol{c}' = 10111111001100001101011100111100.$$

Table 4.1: Interleaving Process with Four Sub-blocks of Length Four Bits

$\boldsymbol{c}_{1,e}$	1011	0000	1101	1100
$\boldsymbol{c}_{2,e}$	1111	0110	1011	0011
LFSR state	10	01	11	00
interleaved sub-blocks	10111111	00110000	11010111	00111100

4.2 Decryption algorithm

The decryption algorithm is as follows.

1. As Bob is aware of the insertion positions, he can deinterleave a ciphertext $\boldsymbol{c}'$ into two codewords with random errors which are the same as $\boldsymbol{c}_{1,e}$ and $\boldsymbol{c}_{2,e}$ that Alice has.

2. Bob can find the random syndromes $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ that Alice used to obtain the random error vectors using

$$\boldsymbol{c}_{i,e}H = (\boldsymbol{c}_i + \boldsymbol{e}_i)H = \boldsymbol{e}_i H = \boldsymbol{s}_i,\ i = 1, 2.$$

   Having found $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$, Bob proceeds to find $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ using the same approach as Alice.

3. The codewords corresponding to the plaintexts $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ are then $\boldsymbol{c}_1 = \boldsymbol{c}_{1,e} + \boldsymbol{e}_1$ and $\boldsymbol{c}_2 = \boldsymbol{c}_{2,e} + \boldsymbol{e}_2$.

4. Having $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$, Bob finds the plaintexts $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$.

The proposed cryptosystem can be used in a secure channel coding scheme. However a slight change has to be made. As in any communication system, errors will be introduced by the channel. These errors will add to the intentional errors introduced by Alice, making the recovery of $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ by Bob more difficult. To account for this, two additional random numbers are needed to encrypt the two plaintexts, which can be obtained from the LFSR or an additional LFSR (resulting in a larger key size). With this adjustment, Bob will have the same random syndromes and consequently the same error vectors that Alice used.

The block diagram in Figure 1 illustrates how the proposed cryptosystem can be employed in a joint encryption and channel coding scheme. Alice and Bob agree on a code and how many sub-blocks the codewords will be divided into, denoted by β. The message source can either generate a single message which is divided into two halves, or two separate messages. For the plaintexts $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$, Alice finds the corresponding codewords, denoted by $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$, respectively. Using two outputs of the LFSR, the syndromes $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ are determined, and then the corresponding error vectors $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are constructed. The error vectors are added to the corresponding codewords, and the resulting words are divided into β sub-blocks. The sub-blocks from one word are inserted into the corresponding sub-blocks from the other word. The insertion positions are determined by a third output of the LFSR. The resulting vector is the ciphertext corresponding to the plaintexts $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$. This ciphertext is transmitted through the channel to Bob. Since Bob has the same random number generator as Alice, upon receiving a word $\boldsymbol{r}$, he can deinterleave it into two words $\boldsymbol{r}_{1,e}$ and $\boldsymbol{r}_{2,e}$, each of which consist of a codeword, a random error vector and an error vector introduced by the channel. Bob proceeds to remove the random error vectors by finding $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. This can easily be done since he has the same random number generator Alice used to generate $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$. At this point, Bob has two words, denoted by $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, which are codewords with channel errors. Error correction is used to remove these errors to recover the codewords $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$, and then $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ can be obtained.

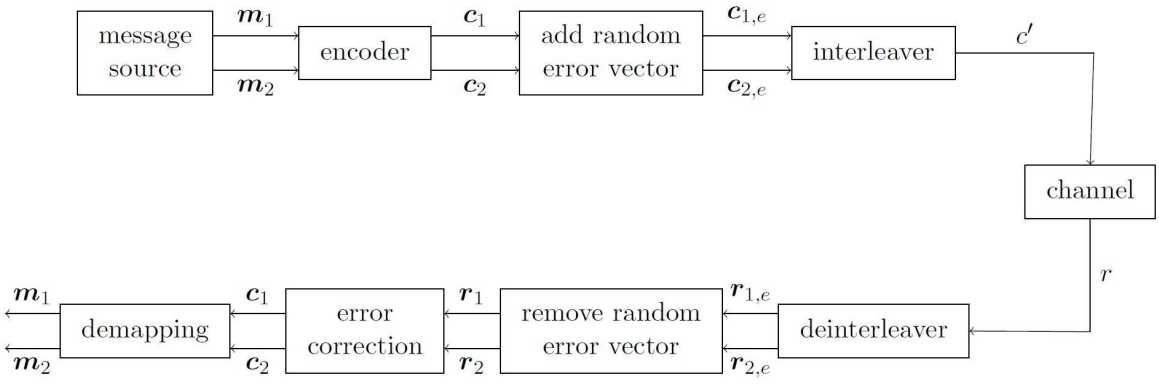

Figure 4.1: Block diagram of a secure channel coding scheme with random bit flipping and interleaving.

4.3 Key size

In this section, the key size of the proposed cryptosystem is analysed and compared with other code based cryptosystems. As mentioned previously, the key of the proposed cryptosystem consists of the initial state of the LFSR used for interleaving which has length $\beta\lfloor\log_2 \frac{n}{\beta}\rfloor$. In the proposed cryptosystem the code employed is public. It will be shown in the next section that this system still provides excellent security. Hence, the key length will be $\beta\lfloor\log_2 \frac{n}{\beta}\rfloor$ bits. For example, if the code is $C(1024, 896)$ and $\beta = 64$, the key size will be $32\lfloor\log_2 64\rfloor = 256$ bits. Note that the proposed cryptosystem encrypts two plaintexts simultaneously, whereas in previous cryptosystems one plaintext is encrypted at a time. Hence for a fair comparison with other cryptosystems, the *effective key size* for encrypting a single plaintext is $256/2 = 128$ bits. This is much smaller than the keys of previous code based cryptosystems proposed in this dissertation. Table 2 compares the key size of the proposed cryptosystem with those in previous chapters. In this table, all cryptosystem parameters are set such that the computational complexity of the best known attack is $O(2^{254})$.

Table 4.2: Comparison of Key Sizes for Code Based Cryptosystems

Scheme	Key size
Section 2.1 random deletion with private code	1254 bits
Section 2.2 random deletion and insertion with private code	1300 bits
Section 3.1 random bit flipping and deletion and insertion with public code	446 bits
Section 3.2 random bit flipping and block deletion with public code	256 bits
Section 4.1 random interleaving with public code	128 bits

4.4 Security analysis

The security of the proposed cryptosystem is now analysed. Code based cryptosystems have been subjected to many attacks (e.g. Stern, RN, Struik-Tilberg, and

Barbero-Ytrehus attacks). Unlike previous code based cryptosystems that employ scrambling matrices and sparse error vectors to conceal the code structure, the proposed cryptosystem is based on random bit flipping and interleaving. Hence any attack designed to recover scrambling matrices and error vectors within the error correcting capability of the code are not effective on the proposed cryptosystem. Another family of attacks on code based cryptosystems are information set decoding attacks which were introduced in [9] and improved in [42]. These attacks are based on the fact that only k information bits (out of n bits of a ciphertext), are required to find the corresponding plaintext. Hence if an adversary can find k error free bits from the ciphertext, the corresponding plaintext can be obtained. A systematic approach for this attack is provided in [42]. However this attack will not be effective on the proposed cryptosystem. The reason is that the random error vectors can have weights up to the code length, which results in many of the ciphertext bits differing from the codeword bits. Therefore, efforts to find k error free bits will be unsuccessful. Further, an adversary would first have to correctly deinterleave the ciphertext into two words before employing an information set decoding attack. Similar to sections 3.1.3 and 3.2.3, note that the total number of random error vectors in the proposed scheme is 2^{n-k}. Hence if $n-k$ is a small value the number of random error vectors is small. This makes the cryptosystem vulnerable to a brute force attack. Hence to avoid this situation it is recommended that $n-k$ be chosen such that the total number of error vectors is large.

It was shown in Section 2.2.4 that random bit insertions in a sub-block can provide a high level of security. As the random interleaving proposed here is similar to the random insertions in Section 2.2.4, it will also provide a high level of security as will now be shown. An adversary can attempt to find the insertion positions via a chosen plaintext attack. He can choose two plaintexts $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ and by comparing the bits of their corresponding codewords with the associated ciphertext find the insertion positions. However, as mentioned previously, due to the dense error vectors employed in the cryptosystem many of the ciphertext bits will differ from the codewords, so all possible insertion positions have to be considered. Therefore, the success rate of this attack is $\left(\frac{n}{\beta}\right)^{-\beta}$. For a $C(1024, 896)$ with $\beta = 64$, this success rate will be 2^{-256}, which is extremely small, thus making this attack ineffective. The best situation for an adversary is if $\boldsymbol{c}_{1,e}$ and $\boldsymbol{c}_{2,e}$ are strings of all zeros and all ones, respectively. In this case, after the interleaving process the ciphertext will be a string of alternating groups of zeros and ones, so finding the insertion positions will be easy. The positions where

the bits in the ciphertext change are the insertion positions. However, the probability of $\boldsymbol{c}_{1,e}$ being the all zero string is at most $2^{-(n-k)}$ as there is at most only one random error vector $\boldsymbol{e}_1$ such that when added to a codeword $\boldsymbol{c}$ can result in the all zero string. Similarly, the probability of $\boldsymbol{c}_{2,e}$ being the all one string is at most $2^{-(n-k)}$. Therefore the probability of this occurring is at most $2^{-2(n-k)}$. In the opposite case (i.e. $\boldsymbol{c}_{1,e}$ and $\boldsymbol{c}_{2,e}$ being the all one and all zero strings, respectively), an adversary will be able to find the insertion positions. The probability of this situation occurring will also be at most $2^{-2(n-k)}$. Therefore, the overall probability of a ciphertext being groups of alternating zeros and ones is $2 \times 2^{-2(n-k)} = 2^{-2(n-k)+1}$. For a $C(1024, 896)$ this success rate is 2^{-255}, which is very small. Hence, this attack is also ineffective.

Although it has been demonstrated that the proposed cryptosystem provides a high level of security against chosen plaintext attacks, modifications can be made to further increase its security. For example, instead of always inserting sub-blocks of $\boldsymbol{c}_{2,e}$ into the corresponding sub-blocks of $\boldsymbol{c}_{1,e}$, the key can be adjusted to also allow sub-blocks of $\boldsymbol{c}_{1,e}$ to be inserted into $\boldsymbol{c}_{2,e}$. For this purpose, the key size is increased slightly to provide β additional bits, one for each pair of sub-blocks. If the corresponding bit is 0, the sub-block of $\boldsymbol{c}_{1,e}$ is inserted into the sub-block of $\boldsymbol{c}_{2,e}$. Otherwise, the sub-block of $\boldsymbol{c}_{2,e}$ is inserted into the sub-block of $\boldsymbol{c}_{1,e}$. Further, to improve the resistance against attacks, error vectors with weight less than t, where t is the number of errors the code can correct, can be discarded. As there are 2^{n-k} correctable error patterns, the probability of error vectors with weight less than t occurring is at most 2^{-k}. Therefore this will not have a significant effect on the number of random error vectors, so discarding these vectors will improve the security of the cryptosystem.

4.5 Conclusion

In this chapter, a symmetric key code based cryptosystem with a public code was presented. This cryptosystem is based on randomly flipping bits in two codewords and interleaving the resulting words. The encryption and decryption algorithms were described for a secure channel coding scheme. The key size and security of the proposed cryptosystem were analysed. It was shown that for a best known attack with computational complexity $O(2^{254})$, it has the smallest key length among all known code based cryptosystems. A security analysis was presented which shows that it is resistant to chosen plaintext attacks because of the number and weights of the

random error vectors. Therefore the proposed cryptosystem provides a high level of security.

Thus far, the idea in public key code based cryptosystems has been to gain security by hiding the structure of the code via scrambling, permuting, and bit flipping. The cryptosystems in Chapters 3 and 4 have public codes (i.e. the structure of the code is not hidden. In the next chapter the idea of having a public code is used in designing a public key cryptosystem. It will be shown that this idea will not compromise the security of the cryptosystem. Furthermore, it will be shown that the cryptosystem is IND-CPA secure which is a necessary requirement for a cryptosystem to be practical.

Chapter 5

Public Key Encryption with a Public Code

In this chapter, a public code is used in a public key infrastructure. This results in a smaller key size than with a private code. Another advantage of not hiding the code structure is that any code Alice and Bob desire can be used, especially codes with higher information rates. In the new public key cryptosystem a plaintext is padded with a random string of bits. The resulting string is then encoded and a number of bits are flipped. The technique employed in Sections 3.1.1, 3.2.1 and 4.1 is used to randomly flip bits. It will be shown that the proposed random padding and flipping technique results in a cryptosystem that has indistinguishability against chosen plaintext attacks. This means that a ciphertext does not reveal any information about the corresponding plaintext (other than its length).

5.1 Encryption algorithm

In this section the key generation, encryption, and decryption algorithms for the proposed cryptosystem are provided. The proposed cryptosystem is a three-tuple of probabilistic polynomial time algorithms $(Gen(.), Enc(.), Dec(.))$ which are described below.

- $Gen(.)$ takes input 1^l, and generates the following:

    1. A code $C(n, k)$ (given by its generator matrix G and parity check matrix H) where n and k denote the length and dimension, respectively,

2. a right inverse of H, denoted by H^{-1}, and
3. an $(n-k) \times (n-k)$ non-singular matrix S.

The output is the public key $pk = (G, S(H^{-1})^T)$ and the secret key $sk = H^T S^{-1}$;

- $Enc(.)$ takes as inputs the public key pk and a plaintext and outputs a ciphertext $\boldsymbol{c}'$. $Enc(.)$ will be denoted as $Enc(pk, .)$ throughout the chapter;
- $Dec(.)$ takes as inputs the secret key sk and a ciphertext $\boldsymbol{c}'$ and outputs the corresponding plaintext. $Dec(.)$ will be denoted as $Dec(sk, .)$ throughout the chapter.

The encryption algorithm $Enc(pk, .)$ is as follows.

1. For a binary plaintext $\boldsymbol{m}$ of length m, Alice chooses a random binary string $\boldsymbol{r}$ of length r. She then uses Bob's public key and obtains the codeword corresponding to $[\boldsymbol{r}|\boldsymbol{m}]$, $\boldsymbol{c} = [\boldsymbol{r}|\boldsymbol{m}]G$, where $[\boldsymbol{r}|\boldsymbol{m}]$ denotes the concatenation of $\boldsymbol{r}$ and $\boldsymbol{m}$.
2. She then chooses a random $n-k$ bit vector $\boldsymbol{s}$ and finds $\boldsymbol{s}S(H^{-1})^T$. If the weight of $\boldsymbol{s}S(H^{-1})^T$ is less than the minimum distance of the code, Alice discards it and chooses another $\boldsymbol{s}$.
3. The ciphertext is $\boldsymbol{c}' = Enc(pk, \boldsymbol{r}|\boldsymbol{m}) = \boldsymbol{c} + \boldsymbol{s}S(H^{-1})^T$.

In the next section the decryption algorithm is provided.

5.2 Decryption algorithm

The decryption algorithm $Dec(sk, .)$ is as follows.

1. For a ciphertext $\boldsymbol{c}'$ Bob finds $\boldsymbol{c}'H^T S^{-1} = \left(\boldsymbol{c} + \boldsymbol{s}S(H^{-1})^T\right) H^T S^{-1} = \boldsymbol{s}$.
2. Using $\boldsymbol{s}$, Bob can find the random error vector Alice used from $\boldsymbol{s}S(H^{-1})^T$.
3. The random error vector is used to obtain the codeword corresponding to the plaintext $[\boldsymbol{r}|\boldsymbol{m}]$ from $\boldsymbol{c} = \boldsymbol{c}' + \boldsymbol{s}S(H^{-1})^T = [\boldsymbol{r}|\boldsymbol{m}]G$.
4. Bob decodes $\boldsymbol{c}$ to find $[\boldsymbol{r}|\boldsymbol{m}]$.
5. The first r bits of $[\boldsymbol{r}|\boldsymbol{m}]$ are discarded to obtain the plaintext $\boldsymbol{m}$.

In the next section the security of the proposed cryptosystem is analysed.

5.3 Security analysis

Code based cryptosystems have been subjected to three types of attacks: structural attacks [38], [39] decoding attacks [11], and information set decoding attacks [9] and [42]. Structural attacks try to recover the secret code from the public key. The code in the proposed cryptosystem is public, so it is immune against to these attacks. The context of structural attacks consists of trying to find the secret key from the private key. We now consider the difficulty in finding the secret key from the public key for the proposed cryptosystem, i.e. finding $H^T S^{-1}$ from $(G, S(H^{-1})^T)$. Note that the secret key is the inverse of $S(H^{-1})^T$ which is part of the public key. Let $A = S(H^{-1})^T = \left(a_{u,v}\right)$, $1 \leqslant u \leqslant n-k$ and $1 \leqslant v \leqslant n$. To find the secret key a matrix $B = H^T S = \left(b_{i,j}\right)$, $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant n-k$, must be found such that $AB = I_{n-k}$. Without loss of generality consider the set of equations obtained by multiplying rows of A and the first column of B

$$\begin{cases} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} + a_{1,3}b_{3,1} + \ldots + a_{1,n}b_{n,1} = 1, \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} + a_{2,3}b_{3,1} + \ldots + a_{2,n}b_{n,1} = 0, \\ a_{3,1}b_{1,1} + a_{3,2}b_{2,1} + a_{3,3}b_{3,1} + \ldots + a_{3,n}b_{n,1} = 0, \\ \qquad\qquad\qquad\qquad \vdots \\ a_{n-k,1}b_{1,1} + a_{n-k,2}b_{2,1} + a_{n-k,3}b_{3,1} + \ldots + a_{n-k,n}b_{n,1} = 0. \end{cases}$$

This set of equations has $n-k$ equations and n unknowns, so the solution is not unique. These set of equations can be written as $AB_1 = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \end{pmatrix}^T$, $1 \leqslant i \leqslant n$, where B_1 and T denote the first column of B and transposition, respectively. From linear algebra, it is known that a solution of this equation will have the form $B_1 = B_1^h + B_1^p$, $1 \leqslant i \leqslant n$, where B_1^h and B_1^p denote the homogeneous and particular solutions, respectively. As the matrix A has dimension $n-k$, its null space has dimension k. Hence there are 2^k solutions for b_1^h, $1 \leqslant i \leqslant n$. As for the particular solution, because A has rank $n-k$ there exists a unique linear combination of its columns that yields $\begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \end{pmatrix}^T$. Hence there exists a unique solution for B_1^p. Therefore, the number of solutions to the set of equations is 2^k. A similar argument holds for other columns of B. Since B has $n-k$ columns, the total number of solutions for B is $2^{k(n-k)}$. Hence the probability of an adversary finding the secret key is $2^{-k(n-k)}$. With an appropriate choice of the code dimension (k) and length (n) this probability is negligible. Therefore, the proposed cryptosystem can be considered secure against this attack.

Decoding attacks try to find the random error vector used in encrypting a plaintext [11]. These attacks usually have a smaller work factor than structural attacks and so are more effective [12]. Decoding attacks are designed to solve the decoding problem, i.e. determine the random error vector used in encrypting a plaintext. This is accomplished by finding the codeword with minimum weight in the code given by the generator matrix

$$G' = \begin{pmatrix} G \\ \boldsymbol{c}' \end{pmatrix},$$

where $\boldsymbol{c}'$ is the ciphertext. It is obvious that a decoding attack is effective only when the error vector has weight less than the minimum distance of the code. The proposed method for finding random error vectors in the encryption algorithm yields vectors with weights exceeding the minimum distance of the code. Hence decoding attacks are not a threat for the proposed cryptosystem.

An information set decoding attack attemps to find k error free bits of the ciphertext corresponding to the plaintext. Algorithms to find these bits were given in [9] and [42]. These algorithms rely on the fact that the random error vectors can be removed using error correction methods. As previously discussed, error correction techniques cannot remove the error vectors used in the proposed cryptosystem. To date, no algorithm has been developed to find k error free bits when more bits than the minimum distance of the code have been flipped. Hence, the proposed cryptosystem is immune against these attacks.

Thus far it is proven that the proposed cryptosystem is immune against known attacks on code based cryptosystems. In the remainder of this section it is shown that the cryptosystem is IND-CPA secure. Let $\gamma = (Gen, Enc, Dec)$ be a public key encryption scheme and consider the following game. A challenger executes Gen to obtain pk and sk. He keeps sk and gives pk to an adversary (distinguisher). The adversary makes unlimited queries to Enc (i.e. executes Enc with different plaintexts). He then chooses two plaintexts $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$. The challenger chooses a plaintext $\boldsymbol{m}_t$, $t \in \{1, 2\}$, encrypts it and gives the result to the adversary. The adversary must distinguish which plaintext is encrypted. Let $\boldsymbol{m}_{t'}$, $t' \in \{1, 2\}$, denote the corresponding response. If $t = t'$, the adversary wins and a 1 is output, otherwise a zero is output. Let

$$\text{Adv}_\gamma = \left| Pr(t = t') - \frac{1}{2} \right|$$

be the advantage in winning this game over random guessing. If Adv_γ is negligible,

the cryptosystem is IND-CPA secure. Note that the McEliece cryptosystem is not IND-CPA secure. To show this, let $\boldsymbol{c}'$ denote the ciphertext corresponding to $\boldsymbol{m}_t$. The adversary can verify which plaintext was encrypted just by finding $\boldsymbol{c}' + \boldsymbol{m}_t G$, $t = 1, 2$, and checking to see which one has a weight less than the error correcting capability of the code.

It was shown in [48] that randomly padding a plaintext in the McEliece cryptosystem can provide IND-CPA security. In order to show that the proposed cryptosystem is IND-CPA secure, two assumptions are used. First, consider the following problem. Let

$$C = \begin{pmatrix} c_{1,1} & c_{1,2} & c_{1,3} & \dots & c_{1,q} \\ c_{2,1} & c_{2,2} & c_{2,3} & \dots & c_{2,q} \\ c_{3,1} & c_{3,2} & c_{3,3} & \dots & c_{3,q} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{q,1} & c_{q,2} & c_{q,3} & \dots & c_{q,q} \end{pmatrix}, D = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_q \end{pmatrix}$$

be two binary matrices where d_i, $1 \leqslant i \leqslant q$, is chosen from the Bernoulli distribution with parameter $\theta \in (0, \frac{1}{2})$. Finding $\boldsymbol{x}$ from the two-tuple $(C, C\boldsymbol{x} + D)$ is believed to be computationally hard to solve [48]. This problem is known as the learning parity with noise (LPN) problem. The hardness of LPN, which is a standard cryptographic assumption, is the first needed for the proof of IND-CPA security.

The second assumption is that G is a pseudorandom matrix. This means that the probability of distinguishing whether G is generated by $Gen(.)$ or is a random $k \times n$ matrix is negligible, in particular for any efficient (i.e. probabilistic poly-time) distinguisher D,

$$\mathrm{Adv}_D(k, n) = \big|Pr(D(G) = 1) - Pr(D(R) = 1)\big| \leq \delta(l),$$

where δ is a negligible function, for $G = Gen(1^l)$, and R chosen uniformly at random. This assumption was used in [49] and [50] for designing identification and digital signature schemes. This assumption and the computational hardness of the LPN problem was used in [48] to show that if a cryptosystem is *admissible*, it is IND-CPA secure. The definition of an admissible cryptosystem is given below.

Definition. A public key cryptosystem with message space $\mathcal{M}$ and random space $\mathcal{R}$ is admissible if there exist two polynomial time algorithms Enc_1 and Enc_2 satisfying the following properties.

1. Divisibility: Enc_1 takes the public key pk and a random string $\boldsymbol{r}'$ and outputs an n bit string. Enc_2 takes the public key and a plaintext $\boldsymbol{m} \in \mathcal{M}$ and outputs an n bit string. The ciphertext corresponding to $\boldsymbol{m}$ is $Enc_1(pk, \boldsymbol{r}') + Enc_2(pk, \boldsymbol{m})$. Thus, the encryption scheme can be divided into two algorithms Enc_1 and Enc_2.

2. Pseudorandomness: For every distinguisher D, the probability of D distinguishing between $Enc_1(pk, \boldsymbol{r}')$ and a random string $\boldsymbol{y}$ of length n is negligible, i.e.

$$\left|Pr(D(pk, Enc_1(pk, \boldsymbol{r}')) = 1) - Pr(D(pk, \boldsymbol{y}) = 1)\right| < \epsilon.$$

Let $G = [G_{rand}^T | G_{msg}^T]$, where G_{rand} and G_{msg} denote the first r rows and last m rows of G, respectively. If $Enc_1(pk, \boldsymbol{r}')$ and $Enc_2(pk, \boldsymbol{m})$ are defined as $Enc_1(pk, [\boldsymbol{r}|\boldsymbol{s}]) = \boldsymbol{r}G_{rand} + \boldsymbol{s}S(H^{-1})^T$ and $\boldsymbol{m}G_{msg}$, respectively, a ciphertext can be found via $Enc_1(pk, \boldsymbol{r}') + Enc_2(pk, \boldsymbol{m})$. Therefore, the proposed cryptosystem satisfies the divisibility property. To show that pseudorandomness is satisfied, first it must be shown that if random errors are added with a distribution other than the Bernoulli distribution, the LPN problem remains hard to solve. It was shown in [48] that if the average weight of a set of vectors is $\lfloor \theta(n+1) \rfloor$ the LPN problem remains hard to solve. In the proposed cryptosystem, the average weight of the random error vectors is $n/2$, so if

$$\theta = \frac{n}{2(n+1)}$$

the LPN problem is hard to solve. The following proposition proves the pseudorandomness of $\boldsymbol{r}G_{rand} + \boldsymbol{s}S(H^{-1})^T$.

Proposition. Let D be a distinguisher. If

$$Pr(D(pk, Enc_1(pk, \boldsymbol{r}')) = 1) - Pr(D(pk, \boldsymbol{y}) = 1) \geq \epsilon,$$

where $\boldsymbol{y}$ is a random string of length n, then there is a distinguisher D' (against the pseudorandomness of G) such that

$$4(n+1)\mathrm{Adv}_{LPN}(p, \theta) + 2\mathrm{Adv}_{D'}(k, n) \geq \epsilon.$$

Proof. The distinguisher D succeeds iff it returns a 1 when the input has the form $\boldsymbol{r}G_{rand} + \boldsymbol{s}S(H^{-1})^T$. Denote this event by *Succ*. Algorithm 1 gives a distinguisher D' which distinguishes between a random matrix and a generator matrix. Let M be a

$k \times n$ matrix. Divide M into the two matrices M_1 and M_2 such that M_1 is a $r \times n$ matrix and M_2 is a $m \times n$ matrix with $k = r + m$.

Algorithm 1 Distinguisher $D'(M, S(H^{-1})^T)$

choose b randomly from $\{0, 1\}$
if $b = 1$ **then**
  select a random $\boldsymbol{r}$ of length r and $\boldsymbol{s}$ of length $n - k$
  find $\boldsymbol{r}M_1$ and $\boldsymbol{s}S(H^{-1})^T)$
  execute $D(pk, \boldsymbol{r}M_1 + \boldsymbol{s}S(H^{-1})^T)$ to find b'
else
  select a random $\boldsymbol{y}$ of length n
  execute $D(pk, \boldsymbol{y})$ to find b'
if $b = b'$ **then return** 1
else return 0

Let *Rand* denote generating a random $k \times n$ matrix M and *Real* denote M being generated using the *Gen* algorithm. The probability of D' succeeding is

$$Pr(b = b'|Real) - Pr(b = b'|Rand), \tag{5.1}$$

and $Pr(b = b'|Real) = Pr(Succ)$. It was shown in [48] that

$$2(n + 1)\text{Adv}_{LPN}(p, \theta) \geq Pr(b = b'|Rand) - \frac{1}{2}. \tag{5.2}$$

From (5.1) and (5.2), it follows that

$$Pr(b = b'|Real) - Pr(b = b'|Rand) \geq Pr(Succ) - \frac{1}{2} - 2(n + 1)\text{Adv}_{LPN}(p, \theta). \tag{5.3}$$

From the assumption that G is indistinguishable we have that

$$\text{Adv}_G(k, n) \geq Pr(b = b'|Real) - Pr(b = b'|Rand). \tag{5.4}$$

Combining (5.3) and (5.4) it follows that

$$\text{Adv}_G(k, n) + 2(n + 1)\text{Adv}_{LPN}(p, \theta) \geq Pr(Succ) - \frac{1}{2},$$

and thus

$$2\text{Adv}_G(k, n) + 4(n + 1)\text{Adv}_{LPN}(p, \theta) \geq \epsilon.$$

□

Therefore, the proposed cryptosystem also satisfies the pseudorandomness property, and hence is IND-CPA secure.

5.4 Key size

In this section the key size of the proposed cryptosystem is evaluateded. From Section 2, the public key is $pk = (G, S(H^{-1})^T)$ and the secret key is $sk = H^T S^{-1}$, so the key size is

$$(k \times n) + (n-k) \times n + n \times (n-k) = n(2n-k).$$

The key size of the McEliece cryptosystem is

$$k \times n + k \times k + k \times n + n \times n = (k+n)^2.$$

For a fair comparison, suppose the same code is used in both cryptosystems. Then

$$\frac{n(2n-k)}{(n+k)^2} = \frac{2-R}{(1+R)^2},$$

where $R = \frac{k}{n}$ is the code rate. Note that if $R \geq 0.31$ the ratio of the key size of the proposed system to the McEliece cryptosystem is less than 1. Thus implying that the key size of the proposed system is smaller than the McEliece cryptosystem. For high rate codes (i.e. code rates close to 1), this ratio approaches 0.25. Thus if $R \approx 1$, the key size of the proposed cryptosystem is only 25% that of the McEliece cryptosystem. Further, since the proposed cryptosystem is immune against known attacks on code based cryptosystems, shorter codes can be used so the key size can be less than 25% that of the McEliece cryptosystem.

5.5 Conclusion

In this chapter a new public key cryptosystem with a public code was presented. This cryptosystem does not use scrambling or permutation matrices. Different from previous public key code based cryptosystems, the error vectors have weights exceeding the error correcting capability of the code. The encryption and decryption algorithms

were presented. The security of the cryptosystem was evaluated and shown to be resistant to previously proposed attacks on code based cryptosystems. The main reason is the large weight of the error vectors employed. It was shown that the proposed cryptosystem has IND-CPA security which is a practical requirement for public key cryptosystems. Further, the method used to find the error vectors results in a smaller key size than the McEliece cryptosystem. If a high rate code is used in both cryptosystems, the key size of the proposed cryptosystem is only 25% that of the McEliece cryptosystem.

Chapter 6

Summary and Future Work

In this chapter a summary of the ideas and techniques used in this dissertation to overcome the drawbacks of code based cryptosystems is provided. In addition, ideas for future work, specifically how to improve the techniques used in this dissertation are provided.

6.1 Summary

Using error correcting codes in cryptosystems was first proposed by McEliece in 1979. This is a public key cryptosystem and is based on the NP-completeness of the decoding problem. In this cryptosystem, bits of a plaintext are scrambled, and bits of the codeword are permuted, and randomly flipped to obtain the ciphertext. This set of operations hides the structure of the code making it hard for an adversary to find the codeword (and consequently the plaintext). Although the McEliece cryptosystem has not been broken yet, it has not been applied in any communication system because of its low information rate and large key size. The use of permutation and scrambling matrices contributes to this size. The low information rate comes from the necessity to use low rate codes in order to hide their structure from an adversary. In this dissertation, encryption schemes that do not require permutation and scrambling matrices are designed. With this approach the key size is reduced and high rate codes can be used.

The first idea was to use random bit deletions and insertions for security. In this encryption scheme, the code along with a PRNG is kept secret. It was shown that the number of deletions and insertions can be adjusted to provide good security and

reliable communication. Further, with an appropriate choice of code parameters the success rate of an adversary being able to find the code and insertion and deletion positions is negligible. A unique feature of this cryptosystem is that if the code is revealed the system still has good security. In this situation for a $C(2048, 1536)$ code the probability of an adversary breaking the system is 2^{-192}. It was also shown that the key size in this situation was only 2458 bits. This suggests that in symmetric key cryptosystems the code need not be kept secret. This is the case for subsequent encryption schemes.

In the third chapter two new symmetric key cryptosystems were presented. Both encryption schemes have public codes. The first encryption scheme is based on randomly flipping, deleting, and inserting bits in a codeword. The random bit flipping is done via random error vectors that have weights exceeding the error correcting capability of the code. The encryption and decryption algorithms were given. The security of the cryptosystem was also analysed. It was shown that although the code is public the security is higher than similar cryptosystems presented in Chapter 2. Specifically, if a $C(2048, 1826)$ code is used the probability of an adversary breaking the cryptosystem is 2^{-254}. The key size is also very small compared to other code based cryptosystems with similar structure. For a 254 bit security level the key size is 446 bits.

In the second part of the third chapter a symmetric key code based cryptosystem was presented which is based on random bit flipping and block deletion. It was shown that the complexity of the encryption and decryption is the same as that in the first part of Chapter 3. At the same time the key size (434 bits) is smaller for the same security level (256 bits). This is comparable to many well known symmetric key encryption schemes (e.g. AES). These results show that this code based cryptosystem can be used in practical applications.

In Chapter 4 random interleaving was employed to design a symmetric key cryptosystem. It can be viewed as an extension of bit insertion used in the second and third chapters. However instead of inserting blocks of random bits, blocks of bits of another codeword are inserted into each other. This means that two plaintexts are encrypted simultaneously to produce one ciphertext. It was shown that this encryption scheme is resistant to classical attacks against code based cryptosystems. Specifically, for a $C(1024, 896)$ code, it was shown that the success rate of an adversary breaking the cryptosystem is 2^{-255}, which is extremely small. The key size is also smaller than the previous symmetric key cryptosystems in this dissertation. It was shown that a

255 bit security level requires a key size of 256 bits. However this key size is used to encrypt two plaintexts simultaneously while the other cryptosystems encrypt only one plaintext at a time. Thus, the effective key size of this cryptosystem (the key size used for encrypting one plaintext) is 128 bits.

In Chapter 5, the ideas presented in previous chapters (having a public code and random bit flipping) were used to design a public key code based encryption scheme. In this cryptosystem a plaintext is padded with a random string of bits and the resulting string is encoded. Then bits of the codeword are randomly flipped to obtain the ciphertext. It was shown that this encryption scheme has indistinguishability against chosen plaintext attacks. This means that a ciphertext reveals no information regarding the corresponding plaintext other than its length. This code based cryptosystem is among the first public key cryptosystems to have a public code, resulting in a significant reduction in key size. A comparison between the key size of the proposed and McEliece cryptosystems showed that if the same code is used, the key size of the proposed cryptosystem is 75% smaller than the McEliece cryptosystem. In addition, there is no need to hide the structure of the code so high rate codes can be used in the proposed encryption scheme. This is a significant step in overcoming the drawbacks of code based cryptosystems and making them suitable for practical applications.

Note that because the operations used in encryption and decryption algorithms are mod 2 addition (XOR), these cryptosystems are more robust against side channel attacks than other cryptosystems such as AES and RSA.

6.2 Future Work

Many improvements to code based cryptosystems have been made in this dissertation to overcome their drawbacks. However there is still work that can be done. The public key cryptosystem presented in Chapter 5 pads a plaintext with random bits prior to encoding. These random bits are encoded along with the plaintext. Random padding is essential in making the cryptosystem IND-CPA secure. Without it an adversary can determine which plaintext is encrypted in the approach in Section 5.3. The rate of the code in this cryptosystem is $R = \frac{r+m}{n}$. For a given code rate R, the more bits that are padded the shorter the plaintext will be. If a method to eliminate the random padding is found it would result in a much higher code rate.

Note that all of the encryption schemes in this dissertation use binary block codes. However, the application of non-binary codes in similarly structured cryptosystems is

not investigated. Hence another idea for further extending this topic is to investigate whether non-binary codes can be used in designing encryption schemes.

In this dissertation combinations of random bit/block deletion, random bit insertion, random interleaving, and bit flipping beyond the error correcting capability of the code were used to construct encryption schemes. However, linear block codes are used in other cryptographic primitives (e.g. hash functions, authentication schemes and digital signatures). Hence another suggestion for future work is to investigate if the techniques used in the previous chapters can be used in constructing other code based cryptographic primitives.

Bibliography

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. Journal*, vol. 27, no. 27, pp. 379–423 (Part I), 623–656 (Part II), 1948.

[2] R. G. Gallager, "Low-Density Parity-Check Codes," Cambridge, MA, MIT Press, 1963.

[3] M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 25, no. 5, pp. 533–547, 1981.

[4] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261-1271, 1996.

[5] D. MacKay and R. Neal, "Good codes based on very sparse matrices," *5th IMA Conf. Cryptography and Coding*, Berlin, Springer-Verlag, pp. 100–111, 1995.

[6] D. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 3, pp. 399-431, 1999.

[7] J. Daemen, V. Rijmen, "The Design of Rijndael," Springer-Verlag, New York, 2002.

[8] J.A. Buchmann, "Introduction to Cryptography," Springer-Verlag, New York, 2nd edition, 2004.

[9] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," *Jet Propulsion Lab. DSN Tech. Rep. 42.44*, pp. 114–116, 1978.

[10] E. R. Berlekamp, R. J. McEliece, and H. C. A. Van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. Inform. Theory*, vol. 24, no. 3, pp. 384–386, 1978.

[11] J. Stern, "A method for finding codewords of small weight," *Coding Theory and Applications, Springer-Verlag LNCS*, vol. 388, pp. 106–113, 1989.

[12] M. Baldi, M. Bianchi, F. Chiaraluce, "Security and complexity of the McEliece cryptosystem based on quasi-cyclic low density parity-check codes," *IET Inform. Security*, vol. 7, no. 3, pp. 212–220, 2013.

[13] N. Sendrier, "Code based cryptography: State of the art and perspectives," *IEEE Privacy Security*, vol. 15, no. 4, pp. 44–50, 2017.

[14] M. Baldi, "Post-quantum cryptographic schemes based on codes," *Proc. Int. Conf. on High Perf. Computing and Simulation*, Genoa, Italy, pp. 908–910, 2017.

[15] C. S. Park, "Improving code rate of McEliece's public-key cryptosystem," *Elec. Letters*, vol. 25, no. 21, pp. 1466–1467, 1989.

[16] M.C. Lin, H.L. Fu, "Information rate of McEliece's public-key cryptosystem," *Elec. Letters*, vol. 26, no. 1, pp. 16–18, 1990.

[17] A. Payandeh, M. Ahmadian, M. R. Aref, "Adaptive secure channel coding based on punctured turbo codes," *IEE Proc. Commun.*, vol. 153, no. 2, pp. 313–316, 2006.

[18] N. Kolokotronis, A. Katsiotis, N. Kalouptsidis, "Secretly pruned convolutional codes: Security analysis and performance results," *IEEE Trans. Inform. Forensics and Security*, vol. 11, no. 7, pp. 1500–1514, 2016.

[19] M. Baldi, M. Bianchi, F. Chiaraluce, "Coding with scrambling, concatenation, and HARQ for the AWGN wire-tap channel: A security gap analysis," *IEEE Trans. Inform. Forensics and Security*, vol. 7, no. 3, pp. 883–894, 2012.

[20] S. A. Barbulescu, "Secure satellite communications and turbo-like codes," *Proc. Int. Symp. Turbo Codes,* Brest, France, pp. 227–230, 2003.

[21] C. Monico, J. Rosenthal, A. Shokrollahi, "Using low density parity check codes in the McEliece cryptosystem," *Proc. IEEE Int. Symp. on Inform. Theory*, Sorrento, Italy, pp. 215, 2000.

[22] M. Baldi, M. Bianchi, F. Chiaraluce, "Optimization of the parity-check matrix density in QC-LDPC code based McEliece cryptosystems," *Proc. IEEE Int. Conf. Commun. Workshops*, Budapest, Hungary, pp. 707–711, 2013.

[23] H. Khayami, T. Eghlidos, M. R. Aref, "A joint encryption-encoding scheme using QC-LDPC codes based on finite geometry," *Available at https://arxiv.org/abs/1711.04611.*

[24] B. Mafakheri, T. Eghlidos, H. Pilaram, "An efficient secure channel coding scheme based on polar codes," *ISC Int. Journal on Inform. Security*, vol. 9, no. 2, pp. 13–20, 2017.

[25] P. Gaborit, "Shorter keys for code based cryptography," *Proc. Int. Workshop on Coding and Cryptography*, Bergen, Norway, pp. 81–91, 2005.

[26] M. Baldi, F. Chiaraluce, "Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes," *Proc. IEEE Int. Symp. Inform. Theory*, Nice, France, pp. 2591–2595, 2007.

[27] A. A. Sobhi Afshar, T. Eghlidos, M. R. Aref, "Efficient secure channel coding based on quasi-cyclic low density parity check codes," *IET Commun.*, vol. 3, no. 2, pp. 279–292, 2009.

[28] M. Baldi, F. Chiaraluce, R. Garello, F. Mininni, "Quasi-cyclic LDPC codes in the McEliece cryptosystem," *Proc. IEEE Int. Conf. Commun.*, Glasgow, UK, pp. 951–956, 2007.

[29] M. Baldi, M. Bodrato, F. Chiaraluce, "A new analysis of the McEliece cryptosystem based on QC-LDPC codes," *Security and Cryptography for Networks, LNCS*, vol. 5229, pp. 246–262, 2008.

[30] T.P. Berger, P.-L. Cayrel, P. Gaborit, A. Otmani, "Reducing key length of the McEliece cryptosystem," *Prog. in Cryptology, AFRICACRYPT 2009, Springer-Verlag LNCS*, vol. 5580, pp. 77–97, 2009.

[31] R. Misoczki, P. S. L. M. Barreto, "Compact McEliece keys from Goppa codes," *Selected Areas in Cryptography, LNCS* vol. 5867, pp. 376–392, 2009.

[32] M. Baldi, "LDPC codes in the McEliece cryptosystem: attacks and countermeasures," *NATO Science for Peace and Security Series D: Inform. and Commun. Security*, vol. 23, pp. 160–174, 2009.

[33] R. Hooshmand, M. R. Aref, "Polar code-based secure channel coding scheme with small key size," *IET Commun.*, vol. 11, no. 15, pp. 2357–2361, 2017.

[34] R. Hooshmand, M. R. Aref, "Efficient secure channel coding scheme based on low-density lattice codes," *IET Commun.*, vol. 10, no. 11, pp. 1365-1373, 2016.

[35] J. C. De Souza, R. M. C. De Souza, "Product codes and private-key encryption," *Proc. IEEE Int. Symp. Inform. Theory*, pp. 489, 1995.

[36] H. M. Sun, S. P. Shieh, "Cryptanalysis of private-key encryption schemes based on burst-error-correcting codes," *Proc. ACM Conf. Computer and Commun. Security*, pp. 153–156, 1996.

[37] T. N. R. Rao, "Joint encryption and error correction schemes," *Proc. Int. Symp. on Computer Architecture*, Ann Arbor, MI, pp. 240–241, 1984.

[38] T. N. R. Rao, K. H. Nam, "A private key algebraic coded cryptosystem," *Proc. Crytpology*, pp.35–48, 1986.

[39] R. Struik, J. Tilburg, "The Rao-Nam scheme is insecure against a chosen plaintext attack," *Advances in Cryptology*, in Proc. Cryptology, pp. 445–457, 1988.

[40] V. G. Umana, G. Leander, "Practical key recovery attacks on two McEliece variants," *Proc. Int. Conf. on Symbolic Computation and Cryptography*, Egham, UK, pp. 27–44, 2010.

[41] J.-C. Faugre, A. Otmani, L. Perret, J.-P. Tillich, "Algebraic cryptanalysis of McEliece variants with compact keys," *Prog. in Cryptology, EUROCRYPT 2010, Springer-Verlag LNCS*, vol. 6110, pp. 279–298, 2010.

[42] C. Wieschebrink, "Two NP-complete problems in coding theory with an application in code based cryptography," *Proc. IEEE Int. Symp. Inform. Theory*, Seattle, WA, pp. 1733–1737, 2006.

[43] M. Esmaeili, M. Dakhilalian, T. A. Gulliver, "New secure channel coding scheme based on randomly punctured quasi-cyclic low-density parity check codes," *IET Commun.*, vol. 8, no. 14, pp. 2556–2562, 2014.

[44] T. Xia, B. Xia, "Quasi-cyclic codes from extended difference families," *Proc. IEEE Wireless Commun. and Networking Conf.*, New Orleans, LA, pp. 1036–1040, 2005.

[45] H. M. Sun, S. P. Shieh, "On private key cryptosystems based on product codes," *Proc. Australian Conf. Inform. Security and Privacy*, pp. 68–79, 1998.

[46] A. I. Barbero, Ø. Ytrehus, "Modifications of the Rao-Nam cryptosystem," *Proc. Int. Conf. Coding Theory, Cryptography and Related Areas*, Guanajuato, Mexico, pp. 1–13, 1998.

[47] P. J. Lee, E. F. Brickell, "An observation on the security of McEliece's public key cryptosystem," *Advances in Cryptology, Proc. EUROCRYPT 1988, Springer-Verlag LNCS*, pp. 275–280, 1988.

[48] R. Nojima, H. Imai, K. Kobara, K. Morozov, "Semantic security for the McEliece cryptosystem without random oracles," *Designs, Codes and Cryptography*, vol. 49, no. 1-3, pp. 298–305, 2008.

[49] P.-L. Cayrel, P. Gaborit, M. Girault, "Identity based identification and signature schemes using correcting codes," *Proc. Int. Workshop on Coding and Cryptography*, Versailles, France, pp. 69–78, 2007.

[50] N. Courtois, M. Finiasz, N. Sendrier, "How to achieve a McEliece-based digital signature scheme," *Advances in Cryptology, ASIACRYPT 2001, Springer-Verlag LNCS*, vol. 2248, pp. 157–174, 2001.