Efficient Code-Based Cryptosystems for Post-Quantum Cryptography

by

Farshid Haidary Makoui

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

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

Efficient Code-Based Cryptosystems for Post-Quantum Cryptography

by

Farshid Haidary Makoui

B.Sc., Tehran Central Azad University, 1997
M.Eng., Toronto Metropolitan University, 2018

University of Victoria

Supervisory Committee

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

Dr. Riham AlTawy, Departmental Member
Department of Electrical and Computer Engineering

Dr. Venkatesh Srinivasan, Outside Member
Department of Computer Science

Abstract

There is increasing growth in e-commerce, blockchain, mobile services, medical and industrial IoT, online banking, and service applications. Cryptographic primitives play a crucial role in securing these applications. Thus, the security of cryptographic primitives is an important issue. The Shor algorithm illustrates how quantum attacks seriously threaten the safety of these primitives. Code-based cryptography is one of several approaches resistant to quantum attacks. To date, no attack has been able to break a code-based cryptosystem in polynomial time. Despite the remarkable level of security they offer, code-based cryptosystems have received minimal attention in practical applications. The main reason is the considerably large public and private key sizes. For example, the McEliece code-based cryptosystem uses binary Goppa codes with large block sizes. The use of code-based cryptography in digital signatures is also limited, primarily because the ciphertexts do not span the entire vector space. The Courtois-Finiasz-Sendrier (CFS) scheme is a widely recognized code-based digital signature scheme. However, its adoption is limited due to the low success rate of signing which in turn increases the signature processing time. This dissertation aims to address the above challenges by introducing new code-based algorithms with smaller key sizes and reduced processing times. A scheme is introduced to construct $2^{k\times(n-k)}$ generalized inverse matrices for a matrix H with dimensions $(n-k)\times n$. An algorithm is also given to construct a random inverse matrix from the $2^{k\times(n-k)}$ choices. Furthermore, a new public key generation algorithm is given that takes advantage of random inverse matrices to construct public and private keys. This algorithm plays a crucial role in the proposed code-based cryptosystem, enabling smaller key sizes compared to the traditional McEliece cryptosystem. The proposed code-based digital signature incorporates signing and verification algorithms with lower complexity and higher success rates than the CFS digital signature, leading to reduced processing times.

Contents

List of Tables

Chapter 1

Introduction

Post-quantum cryptography (PQC) deals with cryptographic methods that are considered resilient against potential cryptanalytic attacks carried out using quantum computers. The security of current algorithms relies on the difficulties associated with mathematical problems including integer factorization and discrete logarithms. These problems are fundamental to cryptographic algorithms such as Rivest, Shamir, and Adleman (RSA) [1], the Digital Signature Algorithm (DSA) [2] and elliptic-curve cryptography (ECC) [3, 4]. The Shor algorithm [5] demonstrates that quantum attacks pose a significant threat to cryptographic primitives by efficiently solving the integer factorization and discrete logarithm problems. Recent developments have intensified this threat and therefore, the National Institute of Standards and Technology (NIST) has considered various proposals for the post-quantum era. Post-quantum cryptography, as described in [6], is the development of cryptographic algorithms [7, 8, 9] to be secure against quantum attacks. This research has mainly focused on multivariate, hash-based, error correcting, code-based, and lattice-based algorithms. In 2016, NIST initiated a competition where, notably, 19 of the submissions were based on codes. In July 2022, NIST called for submissions for the fourth round. In this round, two code-based cryptosystems based on the McEliece framework were included, namely classic McEliece [10] and Bit Flipping Key Encapsulation (BIKE) [11].

The code-based cryptographic primitives in [12] offer resistance to quantum attacks. The security of code-based cryptosystems relies on the complexity of problems related to decoding and code distinguishability [12, 13, 14]. The inability to distinguish between a scrambled parity check matrix and other random matrices is an NP-problem [14, 15], and thus so is decoding a linear code without knowledge of its algebraic structure [16]. The first code-based cryptosystem was introduced by McEliece [17] and the second by Niederreiter [18] which is primarily applied in code-based digital signatures [19]. This chapter provides a brief introduction to code-based cryptographic primitives, in particular, public key encryption and digital signature schemes. The drawbacks of these primitives are also discussed.

1.1 Cryptography

Cryptography plays a crucial role in protecting data and ensuring secure communications by making them accessible only to the intended recipients. It encompasses the study, development, and implementation of methods to ensure the security of data during both transmission and storage. Cryptography is employed in various applications including data privacy, financial transactions and secure data storage.

The key terms used in this dissertation are as follows.

- Plaintext: A message that needs to be protected, denoted by $\boldsymbol{m}$.
- Ciphertext: A transformed version of the plaintext, denoted by $\boldsymbol{c}$.
- Key: An essential component in many cryptosystems, typically generated using mathematical algorithms. Keys can be symmetric or asymmetric. Note that while keys are integral to many cryptosystems, some cryptographic functions, such as hash functions, do not require keys and are known as unkeyed cryptosystems.
- Cryptosystem: A set of cryptographic algorithms designed to ensure secure communication and protect data integrity. It typically encompasses various

components including key generation, encryption, decryption, digital signatures, and authentication mechanisms.

- Key generation algorithm, $Gen(\lambda)$: An algorithm used to generate the public key (pk) and private key (sk) using an input (λ). Key generation algorithms play a crucial role in establishing secure communication channels and ensuring data confidentiality and integrity. In public-key cryptography, the security of the system relies on the confidentiality of the private key, which must be kept secret by the key holder. The public key, on the other hand, can be widely distributed.

- Encryption algorithm, $Enc(\boldsymbol{m}, pk)$: The process of converting plaintext into ciphertext using an algorithm and an encryption key, which is a public key in public-key encryption or a secret key in symmetric encryption [20].

- Decryption algorithm, $Dec(\boldsymbol{c}, sk)$: The reverse process of encryption which involves converting ciphertext back into plaintext. This is achieved through the use of a decryption algorithm and the appropriate decryption key, which is a private key in public-key encryption or a secret key in symmetric encryption.

- Signing algorithm, $Sign(sk, pk, \boldsymbol{doc})$: An algorithm used to generate a digital signature. Here, $\boldsymbol{doc}$ denotes the document being signed and σ denotes the resulting signature.

- Verification algorithm, $Ver(\sigma, pk, \boldsymbol{doc})$: An algorithm used to verify the authenticity and integrity of a digital signature. The verification algorithm takes three inputs: the digital signature σ, the document $\boldsymbol{doc}$ and the public key pk. It evaluates the validity of the signature and confirms the integrity of the data, ensuring that no alterations have occurred since the time of signing.

1.1.1 Binary linear block codes

In data communications, linear block codes play a crucial role in the reliability of data transmission. It involves encoding a message $\boldsymbol{m} = (m_1, m_2, \ldots, m_k)$ into a codeword

$\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ where the length of the codeword (n) is greater than or equal to the length of the message (k). Therefore, for a k-bit message, there exists a corresponding n-bit codeword, resulting in a total of 2^k codewords for the 2^k possible messages. A set of k linearly independent binary vectors of length n forms a subspace of F_2^n known as a $C(n,k)$ linear block code. Consequently, each codeword contains an additional $n-k$ bits compared with the corresponding message.

A code $C(n,k)$ defines a k-dimensional subspace of F_2^n [24]. Consequently, there exist k linearly independent codewords $\boldsymbol{g}_i$, $i = 1, 2, \ldots, k$. A codeword $\boldsymbol{c}$ can be expressed as a linear combination of these k codewords. The $\boldsymbol{g}_i$ are arranged into a $k \times n$ generator matrix $G = [\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_k]^T$, where T denotes transpose. A message $\boldsymbol{m}$ is then encoded as

$$\boldsymbol{c} = \boldsymbol{m}G. \tag{1.1}$$

For a k-dimensional subspace of F_2^n there exists an m-dimensional subspace, $m = n-k$, referred to as the dual space. This dual space, denoted $C^{\perp}$, is a linear code of length n. Every vector in the dual space is orthogonal to every vector in the original subspace. If H generates the dual space corresponding to the generator matrix G of the code $C(n,k)$, then $GH^T = \boldsymbol{0}$. A vector $\boldsymbol{c}$ is considered a codeword in $C(n,k)$ if and only if it satisfies the condition $\boldsymbol{c}H^T = \boldsymbol{0}$ [25]. The systematic form of a generator matrix G is defined as

$$G = (I_k | P''_{(k\times(n-k))}). \tag{1.2}$$

The corresponding systematic parity check matrix has the form

$$H = (P''^T_{(n-k)\times k} | I_{n-k}). \tag{1.3}$$

1.1.2 Hamming weight and minimum distance

The Hamming weight is the number of non-zero elements in a vector. Therefore, the Hamming weight of a binary codeword $\boldsymbol{c}$, denoted $w(\boldsymbol{c})$, is the number of non-zero bits in $\boldsymbol{c}$.

The minimum distance of a code $C(n, k)$, denoted $d_{\min}(C)$, is the smallest Hamming distance between any two distinct codewords. Therfore, the minimum distance is the minimum number of bit flips required to transform one codeword into another.

The error-correcting capability of a code is the maximum number of errors that can be corrected. A code with minimum distance $d_{\min}(C)$ can detect up to $d_{\min}(C) - 1$ errors and correct up to t errors where

$$t = \left\lfloor \frac{d_{\min}(C)-1}{2} \right\rfloor.$$

1.2 Syndrome decoding

The received word after transmission over a noisy communication channel is

$$\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e},$$

where $\boldsymbol{c}$ is the transmitted codeword and $\boldsymbol{e}$ is the error vector. The syndrome of the received word is

$$\begin{aligned} \boldsymbol{s} = \boldsymbol{r}H^T &= (\boldsymbol{c} + \boldsymbol{e})H^T \\ &= \boldsymbol{c}H^T + \boldsymbol{e}H^T \\ &= \boldsymbol{0} + \boldsymbol{e}H^T \\ &= \boldsymbol{e}H^T. \end{aligned}$$

There exists a one-to-one relationship between error vectors with maximum weight t and syndromes. The syndrome is used to identify the corresponding error vector $\boldsymbol{e}$ within the error-correcting capability of the code.

1.3 Code-based cryptosystems

The first code-based cryptosystem was introduced by McEliece in 1978 and is known as the McEliece cryptosystem [17]. The security of this cryptosystem is based on the hardness of the decoding and code distinguishability problems [13, 14]. The inability to distinguish between a scrambled parity check matrix and a random matrix is an NP-hard problem [15], so decoding a linear code without knowledge of its algebraic structure is also an NP-hard problem [16].

Another code-based cryptosystem, known as the Niederreiter cryptosystem, was introduced in 1986 [18]. It has found applications in digital signatures, notably the CFS scheme [19]. To date, no attack has been able to break a code-based cryptosystem in polynomial time. This section describes the McEliece and Niederreiter cryptosystems, providing an overview of their key structures, encryption and decryption algorithms, and drawbacks.

In the field of cryptography, Alice and Bob are commonly used characters to illustrate the encryption and decryption processes. Alice encrypts plaintext by transforming it into ciphertext, while Bob decrypts ciphertext back into plaintext. In the context of digital signatures, Alice signs the document and sends it to Bob, who then verifies the signature and checks the integrity of the received document.

1.3.1 The McEliece cryptosystem

In the McEliece cryptosystem, the plaintext bits are first scrambled, and then the corresponding codeword is permuted. Up to t bits are flipped where t is the error correcting capability of the code. This is a public key cryptosystem where the public key is the product of a non-singular $k \times k$ scrambling matrix, a $k \times n$ generator matrix of the code, and an $n \times n$ permutation matrix. The private key consists of these three matrices. The encryption and decryption algorithms of this system are given below.

A code $C(n,k)$ is selected with generator matrix G along with a scrambling matrix S and a permutation matrix P. The public key is $pk = SGP$ and the private key is $sk = (S, G, P)$. The encryption algorithm of the McEliece cryptosystem is as follows.

1. For a plaintext $\boldsymbol{m}$ of length k, Alice uses the public key of Bob to encrypt it via

$$\boldsymbol{c} = \boldsymbol{m}SGP.$$

2. She flips some of the bits of $\boldsymbol{c}$ by selecting a random vector $\boldsymbol{e}$ of length n and weight at most t. The Hamming weight of the error vector must be less than or equal to the error correcting capability of the code, $w(\boldsymbol{e}) \leq t$. The ciphertext is then

$$\boldsymbol{c'} = \boldsymbol{c} + \boldsymbol{e} = \boldsymbol{m}SGP + \boldsymbol{e}. \tag{1.4}$$

The decryption algorithm is as follows.

1. For a ciphertext $\boldsymbol{c'}$, find P^{-1} using the private key. Then multiply $\boldsymbol{c'}$ by P^{-1} to obtain

$$\boldsymbol{c'}P^{-1} = (\boldsymbol{m}SGP + \boldsymbol{e})P^{-1} = \boldsymbol{m}SG + \boldsymbol{e}P^{-1}. \tag{1.5}$$

2. As P is a permutation matrix, P^{-1} is also a permutation matrix, and therefore the vector $\boldsymbol{e}P^{-1}$ has the same weight as $\boldsymbol{e}$. Thus, $\boldsymbol{c'}P^{-1}$ can be decoded to obtain $\boldsymbol{m}S$.

3. Multiply $\boldsymbol{m}S$ by S^{-1} to obtain the plaintext $\boldsymbol{m}$.

It was shown in [26] that the probability of a successful attack against the McEliece cryptosystem is $\binom{n-t}{k}/\binom{n}{k}$ where t is the error correction capability of the code $C(n,k)$. This probability can be made negligible with an appropriate choice of n, k, and t. Based on this, the recommended parameters are $n = 1024$, $t = 50$, and $k \geq 524$. However, a drawback of the McEliece cryptosystem is the large key size required to provide sufficient security.

1.3.2 The Niederreiter cryptosystem

The Niederreiter cryptosystem can be considered the dual of the McEliece cryptosystem [18]. It is based on the hardness of the syndrome decoding problem. Similar to the McEliece cryptosystem, a code $C(n, k)$ is chosen with parity check matrix H along with an $(n-k) \times (n-k)$ scrambling matrix S and an $n \times n$ permutation matrix P. The Niederreiter cryptosystem is a public key encryption scheme where the public key is the product of S, H, and P, $pk = SHP$, and the private key is $sk = (S, H, P)$. The encryption algorithm for the Niederreiter cryptosystem is as follows.

1. For a plaintext $\boldsymbol{m}$ of length n, the corresponding ciphertext is

$$\boldsymbol{c'} = SHP\boldsymbol{m}^T. \tag{1.6}$$

The decryption algorithm is as follows.

1. For ciphertext $\boldsymbol{c'}$ find $S^{-1}\boldsymbol{c'} = HP\boldsymbol{m}^T$.
2. Use syndrome decoding to obtain $P\boldsymbol{m}^T$.
3. $P^{-1} \times P\boldsymbol{m}^T$ gives the plaintext $\boldsymbol{m}$.

1.3.3 CFS digital signature scheme

The CFS signature scheme is a well-known code-based digital signature scheme based on the Niederreiter cryptosystem [18]. In this scheme, a document is first hashed to compress it to n bits where n is the length of the code used in the cryptosystem. The CFS scheme considers the hashed document as the ciphertext. The signing algorithm, verification algorithm, security, and drawbacks of this scheme are given below.

The signing algorithm [19], consists of the following steps.

1. For a document $\boldsymbol{doc}$, hash it using a hash function $h()$ to obtain $h(\boldsymbol{doc})$ and set $i = 0$.

2. Find $h(h(\boldsymbol{doc})|i)$ where $|$ denotes concatenation.

3. Decrypt $h(h(\boldsymbol{doc})|i)$ using the decryption algorithm of the Niederreiter cryptosystem to get $\boldsymbol{sig}$. If this fails, increase i by 1 and repeat Step 2.

4. Output $(\boldsymbol{sig}, i)$ as the signature of the document $\boldsymbol{doc}$.

The verification algorithm of the CFS signature scheme is as follows.

1. For the signature $(\boldsymbol{sig}, i)$ of a document doc, find $h(h(\boldsymbol{doc})|i)$.

2. The signature is valid if

$$h(h(\boldsymbol{doc})|i) = SHP\boldsymbol{sig}^T, \tag{1.7}$$

   otherwise the signature is not valid.

1.3.4 CFS performance analysis

The complexity of signing is the main reason code-based signatures are not employed in practical applications. The signing success rate is the probability of obtaining a valid signature in Step 3. This rate is less than 1 because the ciphertexts do not cover the entire vector space, so a signature may not be decodable [21, 22, 23]. In the CFS signature scheme, an integer i is appended to $\boldsymbol{doc}$ iteratively to ensure that a decodable signature is obtained. This increases the complexity and computation time. It has been shown that on average, it will take $t!$ executions of the CFS algorithm to find a valid signature, so the success rate is $\frac{1}{t!}$ [19].

1.4 Contributions and outline

To date, the McEliece cryptosystem is resistant to cryptographic attacks. However, it has yet to find practical application primarily due to its large key size. Similarly, the

CFS code-based digital signature is resistant to cryptographic attacks, but its adoption is hindered by the relatively slow signing process. This dissertation addresses these issues by using non-square full-rank matrices within the context of key generation. Note that the inverse of a full-rank square matrix is unique while the inverse of a full-rank non-square matrix is not.

Chapter 2 introduces an algorithm to construct $2^{m(n-m)}$ distinct inverse matrices for a full-rank, non-square matrix of dimension $m \times n$, $n > m$. Therefore, for a parity check matrix H of the code $C(n, k)$ of size $(n - k) \times n$, there are $2^{k \times (n-k)}$ distinct inverse matrices. An algorithm to generate a random inverse matrix is also given. The idea is to generate a random inverse of the parity check matrix for $C(n, k)$ which can be used in a public key generation scheme to enhance the security of the code. This makes it a challenging problem to determine the matrix used. The proposed method has better computational efficiency than the Moore-Penrose (MP) and Gauss-Jordan (GJ) methods.

Chapter 3 introduces an efficient code-based cryptosystem based on the McEliece cryptosystem. This scheme leverages the random inverse matrix algorithm for key generation. This increases the complexity for potential adversaries as the probability of constructing a particular inverse is negligible for appropriate values of n and k. The proposed cryptosystem mitigates the primary drawback of the McEliece code-based cryptosystem by reducing the size of the public key from 292.5 kB to 26.0 kB. This reduction is achieved by using smaller code parameters while still maintaining a high level of security.

Chapter 4 introduces a new code-based digital signature based on the McEliece cryptosystem. This scheme employs the random inverse matrix algorithm introduced in Chapter 2 for key generation. The primary reason code-based signatures are not commonly used in practical applications is the complexity of the signing process coupled with low success rates. The proposed digital signature has lower complexity and provides faster signing with a higher success rate than the CFS digital signature. It

is shown that the probability of an adversary successfully forging the signature is negligible.

Chapter 5 introduces the concept of a dual inverse matrix A, which serves as both the transpose and inverse of the parity check matrix. It is used in a new key generation algorithm. A code-based cryptosystem is introduced that uses the dual inverse matrix in encryption and decryption. A digital signature is also given that employs the dual inverse matrix for signing and verification. The size of this digital signature is compared with lattice-based digital signature schemes such as Bliss and Dilithium [45, 46, 47, 49].

Chapter 6 provides a summary of the contributions and suggests future research directions in the context of key generation.

Chapter 2

Inverse Matrix Construction for Cryptography Applications

The applications of non-square binary inverse matrices span various domains including digital communications [27], navigation systems [28], data storage [29], error-correction coding and code-based cryptography [30] [31]. An invertible matrix must have full rank. A non-square matrix with m rows and n columns, $n > m$, is considered full rank if its rows are linearly independent. The Gauss-Jordan (GJ) algorithm [32] is widely used for solving linear systems of equations in the form $Ax = b$ [33] and also in constructing inverses for non-square matrices. The Moore-Penrose (MP) [34, 35] provides a unique inverse matrix and has been used in data analysis and machine learning [36]. For a full-rank binary matrix with dimensions $m \times n$, $n > m$, there exist $2^{m(n-m)}$ distinct inverse matrices. Consequently, for a full-rank parity check matrix of the code $C(n, k)$ with dimensions $(n - k) \times n$, there are $2^{k\times(n-k)}$ distinct inverse matrices where $n > k$ and $m = n - k$. This chapter introduces an algorithm for constructing all inverses of a non-square parity check matrix H. An algorithm is also given to generate a random inverse matrix. It is shown that this method is more efficient than the Gauss-Jordan and Moore-Penrose methods. It reduces the complexity of inverse matrix construction which is important when dealing with large matrices. The limitation of the Moore-Penrose [37] and Gauss-Jordan methods with

large-sized matrices [38] are discussed.

2.1 Non-square inverse matrix

The parity check matrix, denoted as H, is a full-rank binary matrix with the dimensions $m \times n$, where $n > m$ and $m = n - k$. Therefore, the inverse of the parity check matrix, denoted as H^{-1}, is characterized by dimensions $n \times m$ and must satisfy the equation $HH^{-1} = I_m$, where H is an $m \times n$ matrix and I_m is the identity matrix of order m

$$HH^{-1} = \begin{pmatrix} p_{1,1} & p_{2,1} & p_{3,1} & \cdots & p_{n,1} \\ p_{1,2} & p_{2,2} & p_{3,2} & \cdots & p_{n,2} \\ p_{1,3} & p_{2,3} & p_{3,3} & \cdots & p_{n,3} \\ \vdots & \vdots & \vdots & & \vdots \\ p_{1,m} & p_{2,m} & p_{3,m} & \cdots & p_{n,m} \end{pmatrix} \times \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,m} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,m} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{n,1} & a_{n,2} & a_{n,3} & \cdots & a_{n,m} \end{pmatrix} = I_m. \tag{2.1}$$

This results in a linear equation system with m^2 equations and mn variables (a_{ij}). Consequently, there are $2^{m(n-m)}$ solutions available for matrix H^{-1}. It is also shown in [39] that the inverse parity check matrix H^{-1} is not unique, and there can be a total of $2^{k\times(n-k)}$ possible matrices, where $m = n - k$.

2.2 Inverse matrix construction

The equation $HH^{-1} = I_m$ results in a linear system with m^2 equations and mn variables, indicating an excess of mk variables compared to the number of equations. This section introduces a method to reduce the number of variables, thereby creating a linear equation system with an equal number of m^2 equations and variables. The approach enables the construction of all $2^{k\times(n-k)}$ available inverse matrices for the parity check matrix H.

The proposed method is based on the observation that H^{-1} is not unique and has m columns, each of which can have 2^k available values. This leads to a diverse set of 2^k distinct vectors for each column in the matrix.

Let Z_i be a set of 2^k column vectors of the length n, where $1 \leq i \leq m$, such that the i-th column of H^{-1} belongs to a column set Z_i, comprising 2^k column vectors

$$Z_i = \left\{ \begin{array}{ccccc} z_{1,1} & z_{1,2} & z_{1,3} & \cdots & z_{1,2^k} \\ z_{2,1} & z_{2,2} & z_{2,3} & \cdots & z_{2,2^k} \\ \vdots & \vdots & \vdots & & \vdots \\ z_{k,1} & z_{k,2} & z_{k,3} & \cdots & z_{k,2^k} \\ --- & --- & --- & -- & --- \\ z_{(k+1),1} & z_{(k+1),2} & z_{(k+1),3} & \cdots & z_{(k+1),2^k} \\ z_{(k+2),1} & z_{(k+2),2} & z_{(k+2),3} & \cdots & z_{(k+2),2^k} \\ \vdots & \vdots & \vdots & & \vdots \\ z_{n,1} & z_{n,2} & z_{n,3} & \cdots & z_{n,2^k} \end{array} \right\}. \tag{2.2}$$

Hence, there are 2^k columns for each Z_i set. To reduce the number of variables, the Z_i set can be partitioned into subsets Z_i^1 and Z_i^2. The first subset, Z_i^1, encompasses rows 1 to k, while the second subset, Z_i^2, includes rows $k+1$ to n. By defining the 2^k possible binary vectors of length k for the subset Z_i^1, ranging from all zeros to all ones, this approach eliminates mk variables, enabling the solution of the linear equations with m^2 equations and variables. This simplifies the construction of elements in the second subset Z_i^2.

The subsets Z_i^1 and Z_i^2 can be represented as follows

$$Z_i^1 = \begin{Bmatrix} z_{1,1} & z_{1,2} & z_{1,3} & \cdots & z_{1,2^k} \\ z_{2,1} & z_{2,2} & z_{2,3} & \cdots & z_{2,2^k} \\ z_{3,1} & z_{3,2} & z_{3,3} & \cdots & z_{3,2^k} \\ \vdots & \vdots & \vdots & & \vdots \\ z_{k,1} & z_{k,2} & z_{k,3} & \cdots & z_{k,2^k} \end{Bmatrix}, \tag{2.3}$$

$$Z_i^2 = \begin{Bmatrix} z_{(k+1),1} & z_{(k+1),2} & z_{(k+1),3} & \cdots & z_{(k+1),2^k} \\ z_{(k+2),1} & z_{(k+2),2} & z_{(k+2),3} & \cdots & z_{(k+2),2^k} \\ z_{(k+3),1} & z_{(k+3),2} & z_{(k+3),3} & \cdots & z_{(k+3),2^k} \\ \vdots & \vdots & \vdots & & \vdots \\ z_{n,1} & z_{n,2} & z_{n,3} & \cdots & z_{n,2^k} \end{Bmatrix} \tag{2.4}$$

The elements in Z_i^2, $z_{(k+b),d}$, $1 \leq b \leq n-k$ and $1 \leq d \leq 2^k$, must satisfy

$$\left(\begin{array}{ccc|c} p_{1,1} & \cdots & p_{k,1} & \\ p_{1,2} & \cdots & p_{k,2} & I_{n-k} \\ \vdots & \ddots & \vdots & \\ p_{1,(n-k)} & \cdots & p_{k,(n-k)} & \end{array}\right) \times \left(\begin{array}{c} z_{1,d} \\ z_{2,d} \\ \vdots \\ z_{k,d} \\ \hline z_{(k+1),d} \\ z_{(k+2),d} \\ \vdots \\ z_{n,d} \end{array}\right) = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{2.5}$$

Hence, when $b = 1$

$$z_{(k+1),d} = 1 + p_{1,1}z_{1,d} + p_{2,1}z_{2,d} + \cdots + p_{k,1}z_{k,d}, 1 \leq d \leq 2^k,$$

and if $b \neq 1$

$$z_{(k+b),d} = p_{1,b}z_{1,d} + p_{2,b}z_{2,d} + \cdots + p_{k,b}z_{k,d}, 1 \leq d \leq 2^k,$$

so that

$$\left(\begin{array}{ccc|c} p_{1,1} & \cdots & p_{k,1} & \\ p_{1,2} & \cdots & p_{k,2} & I_{n-k} \\ \vdots & \ddots & \vdots & \\ p_{1,(n-k)} & \cdots & p_{k,(n-k)} & \end{array}\right) \times \left(\begin{array}{c} z_{1,d} \\ z_{2,d} \\ \vdots \\ z_{k,d} \\ --- \\ z_{(k+1),d} \\ z_{(k+2),d} \\ \vdots \\ z_{n,d} \end{array}\right) = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}. \tag{2.6}$$

When $b = 2$

$$z_{(k+2),d} = 1 + p_{1,2}z_{1,d} + p_{2,2}z_{2,d} + \cdots + p_{k,2}z_{k,d}, 1 \leq d \leq 2^k,$$

and if $b \neq 2$

$$z_{(k+b),d} = p_{1,b}z_{1,d} + p_{2,b}z_{2,d} + \cdots + p_{k,b}z_{k,d}, 1 \leq d \leq 2^k.$$

Similarly, the columns of Z_{n-k} must satisfy

$$\left(\begin{array}{ccc|c} p_{1,1} & \cdots & p_{k,1} & \\ p_{1,2} & \cdots & p_{k,2} & I_{n-k} \\ \vdots & \ddots & \vdots & \\ p_{1,(n-k)} & \cdots & p_{k,(n-k)} & \end{array}\right) \times \left(\begin{array}{c} z_{1,d} \\ z_{2,d} \\ \vdots \\ z_{k,d} \\ --- \\ z_{(k+1),d} \\ z_{(k+2),d} \\ \vdots \\ z_{n,d} \end{array}\right) = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}, \tag{2.7}$$

so when $b = n-k$ the result is 1 and when $b \neq n-k$ the result is 0. Thus, if $b = n-k$

$$z_{n,d} = 1 + p_{1,(n-k)}z_{1,d} + p_{2,(n-k)}z_{2,d} + \cdots + p_{k,(n-k)}z_{k,d}),$$

and if $b \neq n-k$

$$z_{(k+b),d} = p_{1,b}z_{1,d} + p_{2,b}z_{2,d} + \cdots + p_{k,b}z_{k,d}, 1 \leq d \leq 2^k.$$

2.2.1 Example

Consider a code $C(n,k)$ with $k = 3$ and $n = 6$ having generator matrix

$$G = (I_k|P_{k\times(n-k)}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix},$$

and

$$H = (P^T|I_{n-k}) = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

The sets of columns are Z_1, Z_2, and Z_3 resulting in $2^{k\times(n-k)} = 2^{3\times3} = 512$ possible matrices. The sets Z_i^1 and Z_i^2 can be expressed as

$$Z_i^1 = \begin{Bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{Bmatrix},$$

and

$$Z_i^2 = \begin{Bmatrix} z_{(k+1),1} & z_{(k+1),2} & z_{(k+1),3} & \cdots & z_{(k+1),8} \\ z_{(k+2),1} & z_{(k+2),2} & z_{(k+2),3} & \cdots & z_{(k+2),8} \\ z_{(k+3),1} & z_{(k+3),2} & z_{(k+3),3} & \cdots & z_{(k+3),8} \end{Bmatrix}.$$

Combining Z_i^1 and Z_i^2 gives

$$Z_i = \left\{ \begin{array}{cccccccc} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ -- & -- & -- & -- & -- & -- & -- & -- \\ z_{4,1} & z_{4,2} & z_{4,3} & z_{4,4} & z_{4,5} & z_{4,6} & z_{4,7} & z_{4,8} \\ z_{5,1} & z_{5,2} & z_{5,3} & z_{5,4} & z_{5,5} & z_{5,6} & z_{5,7} & z_{5,8} \\ z_{6,1} & z_{6,2} & z_{6,3} & z_{6,4} & z_{6,5} & z_{6,6} & z_{6,7} & z_{6,8} \end{array} \right\}.$$

For $i = 1$, we have

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 0 \\ - \\ z_{4,1} \\ z_{5,1} \\ z_{6,1} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix},$$

so

$$\begin{aligned} z_{41} &= 1 + (0)(0) + (1)(0) + (1)(0) = 1, \\ z_{51} &= (1)(0) + (1)(0) + (0)(0) = 0, \\ z_{61} &= (1)(0) + (0)(0) + (1)(0) = 0. \end{aligned}$$

The elements of Z_1^2 are

$$\begin{aligned} z_{4,d} &= 1 + p_{1,1}z_{1,d} + p_{2,1}z_{2,d} + p_{3,1}z_{3,d}, \\ z_{5,d} &= p_{1,2}z_{1,d} + p_{2,2}z_{2,d} + p_{3,2}z_{3,d}, \\ z_{6,d} &= p_{1,3}z_{1,d} + p_{2,3}z_{2,d} + p_{3,3}z_{3,d}, \end{aligned}$$

so

$$Z_1 = \left\{ \begin{array}{cccccccc} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ -- & -- & -- & -- & -- & -- & -- & -- \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{array} \right\}.$$

The elements of Z_2^2 are

$$\begin{aligned} z_{4,d} &= p_{1,1}z_{1,d} + p_{2,1}z_{2,d} + p_{3,1}z_{3,d}, \\ z_{5,d} &= 1 + p_{1,2}z_{1,d} + p_{2,2}z_{2,d} + p_{3,2}z_{3,d}, \\ z_{6,d} &= p_{1,3}z_{1,d} + p_{2,3}z_{2,d} + p_{3,3}z_{3,d}, \end{aligned}$$

so

$$Z_2 = \left\{ \begin{array}{cccccccc} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ -- & -- & -- & -- & -- & -- & -- & -- \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{array} \right\}.$$

The elements of Z_3^2 are

$$\begin{aligned} z_{4,d} &= p_{1,1}z_{1,d} + p_{2,1}z_{2,d} + p_{3,1}z_{3,d}, \\ z_{5,d} &= p_{1,2}z_{1,d} + p_{2,2}z_{2,d} + p_{3,2}z_{3,d}, \\ z_{6,d} &= 1 + p_{1,3}z_{1,d} + p_{2,3}z_{2,d} + p_{3,3}z_{3,d}, \end{aligned}$$

so

$$Z_3 = \left\{ \begin{array}{cccccccc} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ -- & -- & -- & -- & -- & -- & -- & -- \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array} \right\}.$$

Hence, selecting a random column vector from each set Z_1, Z_2, and Z_3, allows for the construction of a total of $2^{k\times(n-k)} = 2^{3\times3} = 512$ possible H^{-1} inverse matrices.

2.2.2 Random inverse matrix construction

The inverse of a parity check matrix can be partitioned into two sections A_1 and A_2 where A_1 comprises rows 1 to k and A_2 comprises rows $k+1$ to n so that

$$H^{-1} = \left(\begin{array}{cccc} a_{1,1} & a_{1,2} & \cdots & a_{1,(n-k)} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,(n-k)} \\ a_{3,1} & a_{3,2} & \cdots & a_{3,(n-k)} \\ \vdots & \vdots & & \vdots \\ a_{k,1} & a_{k,2} & \cdots & a_{k,(n-k)} \\ --- & --- & -- & --- \\ a_{(k+1),1} & a_{(k+1),2} & \cdots & a_{(k+1),(n-k)} \\ a_{(k+2),1} & a_{(k+2),2} & \cdots & a_{(k+2),(n-k)} \\ a_{(k+3),1} & a_{(k+3),2} & \cdots & a_{(k+3),(n-k)} \\ \vdots & \vdots & & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,(n-k)} \end{array} \right) = \begin{pmatrix} A_1 \\ - \\ A_2 \end{pmatrix}. \tag{2.8}$$

A random inverse of a parity check matrix can be obtained using the following steps.

1. Select a random matrix A_1 with dimensions $(n-k)\times k$.

2. Construct the corresponding matrix A_2 with dimensions $(n-k) \times (n-k)$.

If $n = 20$ and $k = 12$, A_1 consists of $n - k = 8$ randomly selected binary vectors with $k = 12$ rows. One such matrix is

$$A_1 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

The elements of A_2 are

$$A_2 = \begin{pmatrix} a_{(k+1),1} & a_{(k+1),2} & a_{(k+1),3} & \cdots & a_{(k+1),(n-k)} \\ a_{(k+2),1} & a_{(k+2),2} & a_{(k+2),3} & \cdots & a_{(k+2),(n-k)} \\ a_{(k+3),1} & a_{(k+3),2} & a_{(k+3),3} & \cdots & a_{(k+3),(n-k)} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{n,1} & a_{n,2} & a_{n,3} & \cdots & a_{n,(n-k)} \end{pmatrix}, \tag{2.9}$$

where

$$a_{(k+b),d} = \sum_{i=1}^{k} p_{ib} a_{id}, (b \neq d),$$

and

$$a_{(k+b),d} = 1 + \sum_{i=1}^{k} p_{ib} a_{id}, (b = d).$$

In general, this can be expressed as

$$a_{(k+b),d} = 2^{|b-d|} \bmod 2 + \sum_{i=1}^{k} p_{ib} a_{id}. \tag{2.10}$$

For example, $a_{(k+1),1}$ in A_2 is given by

$$a_{(k+1),1} = 1 + p_{11}a_{11} + p_{21}a_{21} + \cdots + p_{k1}a_{k1}.$$

Let A_1 be a $k \times (n-k)$ matrix consisting of $n-k$ column vectors, and A_2 be the $(n-k) \times (n-k)$ matrix with entries determined based on A_1. If $B_1 = P''^{T}_{(n-k)\times k}$ and $B_2 = I_{n-k}$, then

$$HH^{-1} = \Big(B_1|B_2\Big) \times \begin{pmatrix} A_1 \\ - \\ A_2 \end{pmatrix} = B_1A_1 + B_2A_2 = I_{n-k},$$

so

$$A_2 = (B_2)^{-1}(B_1A_1 + I_{n-k}), \tag{2.11}$$

which gives

$$\begin{aligned} HH^{-1} &= \Big(B_1|B_2\Big) \times \begin{pmatrix} A_1 \\ - \\ A_2 \end{pmatrix} = \Big(B_1|B_2\Big) \times \left(\begin{array}{c} A_1 \\ \hline B_1A_1 + I_m \end{array} \right), \\ &= B_1A_1 + B_2(B_1A_1 + I_m) \\ &= B_1A_1 + B_1A_1 + I_m = I_m. \end{aligned}$$

2.2.3 Random inverse matrix construction example

Consider a code $C(n,k)$ with $k = 11$ and $n = 20$ having generator matrix

$$G_{11\times 20} = \left(\begin{array}{c|cccccccccc} & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ I_k & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{array}\right),$$

and parity check matrix

$$H_{(n-k)\times n} = \left(\begin{array}{cccccccccc|c} 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & I_{n-k} \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & \end{array}\right).$$

To construct a random inverse for the above matrix H, randomly select a matrix A_1 with the dimensions $k \times (n-k)$

$$A_1 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}.$$

Construct A_2 using (2.11), $A_2 = (B_2)^{-1}(B_1A_1 + I_{n-k})$

$$B_1A_1 = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$$B_1A_1 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Therefore

$$A_2 = (B_2)^{-1} \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} + I_{n-k},$$

$$A_2 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix},$$

so

$$H^{-1} = \begin{pmatrix} A_1 \\ - \\ A_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}.$$

2.2.4 Inverse matrix construction analysis

This section compares the computation time of the proposed algorithm for random inverse matrices with the MP method. Table 1 presents the computation time for several values of n and k. This shows that the proposed method has a lower time for all values.

The computational complexity is now considered in terms of the number of arithmetic operations. To solve a system of v equations with v unknowns, the number of arith-

Matrix size	MP (ms)	Proposed (ms)
$n = 500$, $k = 213$	94	16
$n = 1568$, $k = 524$	2172	594
$n = 2048$, $k = 768$	5109	2368
$n = 2896$, $k = 1024$	14735	5211

Table 2.1: Random Inverse Matrix Computation Time

GJ	MP	Proposed
$4/3(n-k)^3 + 3(n-k)^2$ $-7/3(n-k)$	$(n-k)^2(4n-1)$ $-n(n-k)$	$(2k-1)(n-k)^2$ $+(n-k)$

Table 2.2: Computational Complexity for Three Algorithms

metic operations needed to obtain the row echelon form (REF) with the GJ method is $v(v+1)/2$ divisions, $(2v^3+3v^2-5v)/6$ multiplications, and $(2v^3+3v^2-5v)/6$ additions/subtractions [40]. Then, the reduced row echelon form (RREF) is constructed to solve the system of linear equations which doubles the number of operations which gives $4/3v^3 + 3v^2 - 7/3v$.

Hence, for a parity check matrix H with dimensions $(n-k) \times n$, the GJ computational complexity to obtain the inverse is $4/3(n-k)^3 + 3(n-k)^2 - 7/3(n-k)$ (excluding the nullspace and vector set calculations). The MP algorithm requires $(n-k)^2(2n-1)$ arithmetic operations to obtain the full-rank matrix HH^T and an additional $(n-k)(2n^2-2nk-n)$ operations to obtain $H^{-1} = H^T(HH^T)^{-1}$ (excluding determinant calculations). Thus, the computational complexity for the MP method is $(n-k)^2(4n-1) - n(n-k)$. The proposed approach requires no arithmetic operations for constructing A_1 as it is determined through random selection. Constructing A_2 requires $(2k-1)(n-k)^2 + (n-k)$ arithmetic operations. Table 2 presents the number of arithmetic operations required with the MP, GJ, and proposed algorithm to construct a random inverse matrix. This shows that the proposed method has lower computational complexity.

The proposed algorithm simplifies the construction of inverse matrices with reduced complexity and can construct all $2^{m(n-m)}$ inverses of the given non-square matrix in any field, regardless of its size. In contrast, the Moore-Penrose method constructs pseudo-inverses [37] which is challenging when the matrix has elements from a finite field. Similarly, Gauss-Jordan elimination also has difficulties when applied to large matrices [38]. In particular, the task of selecting row combinations using Gauss-Jordan elimination is known to be NP-hard, and the time complexity increases exponentially.

2.3 Random inverse of non-systematic binary matrices

Let B be a full rank non-square matrix with dimensions $m \times n$ where $m < n$, such that

$$B_{m\times n} = \begin{pmatrix} b_1 & B_1 & b_2 & B_2 & \dots & b_y & B_x & b_{y+1} \end{pmatrix}, \tag{2.12}$$

where $B_1, B_2, \dots, B_x$, are the matrices with m rows and $n_1, n_2, \dots, n_x$ columns, respectively, such that $n_1 + n_2 + \dots + n_x = n - m$, and $b_1, b_2, \dots, b_{y+1}$, are the column vectors with m rows, $y + 1 = m$. Then not all individual matrices B_i, $i = 1, ..., x$, and vectors b_j, $j = 1, ..., y + 1$, need to be present, each of them can independently be either a null matrix or a null vector. A full-rank matrix B has a minimum of m linearly independent column vectors b_j, $j = 1, ..., y + 1$. For example the following matrix B is a full rank matrix with $rank = 5$

$$B_{5\times 9} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} b_1 & b_2 & b_3 & B_3 & b_4 & b_5 \end{pmatrix},$$

where B_1, B_2, and B_4 are not used, demonstrating that not all matrices are required to be present.

Let A be the inverse of B with

$$A_{n\times m} = \begin{pmatrix} a_1 \\ A_1 \\ a_2 \\ A_2 \\ \vdots \\ a_y \\ A_x \\ a_{y+1} \end{pmatrix}, \tag{2.13}$$

where $A_1, A_2, \ldots, A_x$, are matrices with n_1, n_2,..., n_x rows, respectively, such that $n_1 + n_2 + \ldots + n_x = n - m$, and $a_1, a_2, \ldots, a_{y+1}$, are row vectors with m columns, $y + 1 = m$. Then, the product of B and A is

$$\sum_{i=1}^{x} B_i A_i + \sum_{j=1}^{y+1} b_j a_j = I_m. \tag{2.14}$$

A random matrix A can be generated by randomly selecting $A_1, A_2, \ldots, A_x$ and constructing the corresponding row matrix variables $a_1, a_2, \ldots, a_{y+1}$.

Define A_a and B_b such that

$$(A_a)_{m\times m} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{y+1} \end{pmatrix}, (B_b)_{m\times m} = \begin{pmatrix} b_1 & b_2 & \ldots & b_{y+1} \end{pmatrix}. \tag{2.15}$$

Therefore

$$\sum_{i=1}^{x} B_i A_i + B_b A_a = I_m \tag{2.16}$$

$$B_b A_a = I_m + \sum_{i=1}^{x} B_i A_i$$

$$A_a = (B_b)^{-1}(I_m + \sum_{i=1}^{x} B_i A_i). \tag{2.17}$$

Hence, the columns of the binary matrix B_b are linearly independent, resulting in a determinant of 1. Therefore, matrix B_b is a non-singular matrix. Consequently, by selecting random matrices A_i, matrix A_a can be constructed as described.

For example, consider matrix B with $m = 5$ and $n = 9$ such that

$$B_{5\times 9} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} b_1 & b_2 & b_3 & B_3 & b_4 & b_5 \end{pmatrix},$$

So $A_{9\times 5}$ is

$$A_{9\times 5} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ A_3 \\ a_4 \\ a_5 \end{pmatrix},$$

where A_3 can be randomly selected as

$$A_3 = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

Given matrices B_3 and $(B_b)_{m\times m}$

$$B_3 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

$$(B_b)_{5\times 5} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

the inverse of $(B_b)_{m\times m}$ is

$$(B_b)^{-1} = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Therefore, matrix A_a can be constructed as

$$A_a = (B_b)^{-1}(I_5 + B_3 A_3),$$

where

$$A_a = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

$$(A_a)_{5\times 5} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Having matrices A_3 and A_a, the inverse matrix A can be constructed as

$$A_{9\times 5} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ A_3 \\ a_4 \\ a_5 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

where $B_{(5\times 9)} \times A_{(9\times 5)} = I_5$.

2.4 Conclusion

This chapter presented the construction of inverse matrices for a non-square parity check matrix H. The proposed approach involves defining a column set Z_i with column i of H^{-1} corresponding to a subset of Z_i. This results in a reduction of mk variables, enabling the solution of the linear system of equations with m^2 equations and variables. Consequently, it facilitates the construction of all possible inverse matrices. An algorithm was given to construct an inverse matrix from matrices A_1 and A_2 where A_1 has $n-k$ random binary vectors and A_2 is obtained based on A_1. The computational complexity of the proposed approach was shown to be lower than that of the Moore-Penrose and Gauss-Jordan methods.

Chapter 3

A New Code-Based Cryptosystem with Random Inverse Matrix

Code-based cryptographic methods have received limited attention in practical applications. The main obstacle is the large size of the public and private keys, primarily due to the key size requirements essential for maintaining the security of the cryptosystem. The McEliece-type cryptosystem introduced in [41] utilizes a large minimum distance self-dual code. This results in a reduction of the key size by approximately 38.5% when compared to classical McEliece cryptosystems. In [42], a code-based cryptosystem using generalized Reed-Solomon codes was introduced with a smaller public key of size 88.1 kB. This chapter presents a code-based cryptosystem based on the McEliece scheme that has a higher level of security and smaller public key sizes.

3.1 A new code-based cryptosystem

The proposed cryptosystem encompasses key generation, encryption, and decryption algorithms. In this scheme, the encryption and decryption algorithms use the generalized random inverse parity check matrix proposed in the previous chapter to ensure the security and effectiveness of the cryptosystem.

Code-Based Cryptosystem

Key Generation: $(pk, sk) \leftarrow Gen(\lambda)$, where λ is the algorithm input.

Encryption: $\boldsymbol{c'} \leftarrow Enc(\boldsymbol{m}, pk)$, where $\boldsymbol{c'}$ is the ciphertext for the plaintext $\boldsymbol{m}$.

Decryption: $\boldsymbol{m} \leftarrow Dec(\boldsymbol{c'}, sk)$.

– The key generation algorithm, denoted $Gen(\lambda)$, utilizes the random inverses of the parity check matrix to generate the public and private keys based on the algorithm input, denoted as λ.

– The encryption algorithm, denoted $Enc(\boldsymbol{m}, pk)$, takes the message $\boldsymbol{m}$ and the public key pk as inputs to generate the encrypted message $\boldsymbol{c}$, also referred to as the ciphertext.

– The decryption algorithm, denoted $Dec(\boldsymbol{c}, sk)$, takes the encrypted message $\boldsymbol{c}$ and the private key sk as inputs to decrypt the ciphertext $\boldsymbol{c}$ and obtain the plaintext message.

3.1.1 Key generation

The key generation algorithm generates both the public and private keys. This involves the generator matrix G associated with the code $C(n, k)$ and a random inverse of the parity check matrix H. Furthermore, the key generation algorithm enhances security using operations such as row and column interchange of the generator matrix and the inverse of the corresponding parity check matrix. This enhances the security of the keys and makes the code more resilient against potential structural attacks.

The algorithm $Gen(\lambda)$ employs the following matrices in generating both the public and private keys.

– Matrix G: This is a generator matrix of the code $C(n, k)$ and has size $k \times n$.

– Matrix H^{-1}: This is a randomly generated inverse of the parity check matrix of the code with dimensions $n \times (n-k)$.

– Matrix S: This is a non-singular scrambling matrix with dimensions $k \times k$.

– Matrix P: This is either a non-singular matrix or permutation matrix with dimensions $n \times n$.

– Matrix L: This is a non-singular random matrix with dimensions $(n-k)\times(n-k)$.

The proposed algorithm generates both a public key, which is shared within the network, and a private key, which is kept confidential.

The randomly generated inverse of H is utilized in key generation. As mentioned in the previous chapter, the inverse of the parity check matrix, H^{-1}, is not unique as there are $2^{k\times(n-k)}$ inverses. The algorithm introduced in the previous chapter divides H^{-1} into two parts, A_1 and A_2

$$H = \left(B_1|B_2\right), H^{-1}_{n\times(n-k)} = \begin{pmatrix} A_1 \\ - \\ A_2 \end{pmatrix},$$

so

$$A_2 = B_2^{-1}(B_1A_1 + I_m),$$

where A_1 is a randomly chosen matrix and B_2^{-1} is the identity matrix.

Key Generation Algorithm $Gen(\lambda)$

---

1. Obtain a generator matrix G and corresponding parity check matrix H for $C(n,k)$.

2. Select a random H^{-1} from the $2^{k\times(n-k)}$ choices using a random matrix A_1 and the corresponding matrix A_2
   $H^{-1} = (\frac{A_1}{A_2})$.

3. As in the McEliece cryptosystem, use the generator matrix G, the scrambling matrix S and the permutation matrix P to mask G
   $p_1 = G^{'} = SGP$.

4. Use the non-singular random matrix L and P to mask H^{-1}
   $p_2 = L^{-1}(H^{-1})^T P$.

5. Construct a parity check matrix corresponding to $G^{'} = SGP$
   $Q = H^{'T} = P^{-1}H^T L$, $H' = L^T H(P^{-1})^T$.

6. Public key: $pk \leftarrow (SGP, L^{-1}(H^{-1})^T P)$.

7. Private key: $pr(sk) \leftarrow (S^{-1}, P^{-1}, G, Q)$.

Theorem 1. The public key $L^{-1}(H^{-1})^T P$ has many inverses and the probability of constructing a particular inverse of $L^{-1}(H^{-1})^T P$ can be made negligible.

Proof: The parity check matrix is a full rank matrix and is not unique [39]. The inverse of this matrix has $n-k$ columns, each of which can have 2^k different values, so the number of inverse matrices is $2^{k\times(n-k)}$ [39]. The public key $L^{-1}(H^{-1})^T P$ is a full rank matrix, so the probability of constructing a particular inverse of $L^{-1}(H^{-1})^T P$ is $\frac{1}{2^{k\times(n-k)}}$ which is negligible for appropriate values of n and k.

3.1.2 Encryption scheme

The encryption scheme takes the plaintext as input and transforms it into the corresponding ciphertext.

Encryption $Enc(\boldsymbol{m}, pk)$

---

1. Construct the codeword by computing the product of pk_1 and the plaintext
   $\boldsymbol{c} = \boldsymbol{m}(p_1) = \boldsymbol{m}(SGP)$.

2. Randomly generate an $n-k$ bit vector $\boldsymbol{s}$.

3. Construct the error vector using $\boldsymbol{s}(L^{-1}(H^{-1})^T P)$
   $\boldsymbol{e} = \boldsymbol{s}(p_2)$.

4. Form the ciphertext
   $\boldsymbol{c'} = \boldsymbol{c} + \boldsymbol{e} = \boldsymbol{m}(p_1) + \boldsymbol{s}(p_2)$.

5. Output $\boldsymbol{c'}$.

Note that if the weight of $\boldsymbol{e}$, denoted $\nu = w(\boldsymbol{s}p_2)$, is smaller than minimum distance of the code, another random vector $\boldsymbol{s}$ should be selected.

In addition to her own public and private keys, Alice has additional public keys, including the public key of Bob. Consequently, when encrypting a message, Alice uses the public key of Bob to convert the plaintext into ciphertext before transmitting it over the network. Only Bob has access to the corresponding private key, so only he can decrypt the ciphertext and reveal the message.

3.1.3 Decryption scheme

The decryption scheme takes the ciphertext as input and converts it to the corresponding plaintext.

Decryption $Dec(\boldsymbol{c'}, sk)$

---

1. Construct the vector $\boldsymbol{s}$ using the private key Q and ciphertext $\boldsymbol{c'}$

$$\boldsymbol{s} = \boldsymbol{c'}(P^{-1}H^T L).$$

$$\begin{aligned} \boldsymbol{c'} &= \boldsymbol{m}(p_1) + \boldsymbol{s}(p_2) \\ \boldsymbol{c'} &= (\boldsymbol{m}(SGP) + \boldsymbol{s}(L^{-1}(H^{-1})^T P)) \end{aligned}$$

Multiply with $P^{-1}H^TL$

$$\begin{aligned}
\boldsymbol{c'}(P^{-1}H^TL) &= (\boldsymbol{m}(SGP) + \boldsymbol{s}(L^{-1}(H^{-1})^TP))(P^{-1}H^TL) \\
&= \boldsymbol{m}(SGP)(P^{-1}H^TL) + \boldsymbol{s}(L^{-1}(H^{-1})^TP)(P^{-1}H^TL) \\
&= \boldsymbol{m}S(GH^T)L + \boldsymbol{s}(L^{-1}(H^{-1})^T(H^T)L) \\
&= \boldsymbol{0} + \boldsymbol{s}(I) \\
&= \boldsymbol{s}.
\end{aligned}$$

2. Use vector s and the public key p_2 to construct $\boldsymbol{s}(p_2)$.

3. Form $\boldsymbol{c}$

$$\boldsymbol{c} = \boldsymbol{c'} + \boldsymbol{s}(p_2).$$

4. Find the plaintext $\boldsymbol{m}$ by decoding $\boldsymbol{c}$

$$\begin{aligned}
\boldsymbol{m}SG &= (\boldsymbol{m}SGP)(P^{-1}) \\
\boldsymbol{m}S &= \text{ decode } \boldsymbol{m}SG \\
\boldsymbol{m} &= (\boldsymbol{m}S)(S^{-1}).
\end{aligned}$$

3.1.4 Example

Consider the following non-systematic matrices G and H with $n = 12$ and $k = 7$

$$G = \begin{pmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0
\end{pmatrix},$$

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

therefore

$$B_b = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}, B_i = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

The scrambling matrix is

$$S = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix},$$

the non-singular matrix L is

$$L = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix},$$

and the non-singular matrix P is

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

Key Generation

Randomly select a matrix A_i and then construct the matrix A_a to obtain H^{-1}

$$A_i = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix},$$

$$A_a = (B_b)^{-1}(I_m + \sum_{i=1}^{x} B_i A_i).$$

Therefore

$$A_a = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix} \left(\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix} + I_m \right),$$

$$A_a = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix},$$

$$H^{-1} = \left(\frac{A_i}{A_a} \right) = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Construct the public keys $p_1 = SGP$ and $p_2 = L^{-1}(H^{-1})^T P$, and the private key $Q = P^{-1} H^T L$.

Encryption

1. Alice encrypts the message $\boldsymbol{m} = 1101001$

$$\begin{aligned} \boldsymbol{c} &= \boldsymbol{m}(p_1) \\ &= 101110111011. \end{aligned}$$

2. She chooses a random vector $\boldsymbol{s}$ of size $n - k = 5$

$$\boldsymbol{s} = 10011.$$

3. She constructs an error pattern $\boldsymbol{e}$ using $\boldsymbol{s}$ and her public key (p_2)

$$\begin{aligned} \boldsymbol{e} &= \boldsymbol{s}(p_2) \\ &= 10011(p_2) \\ &= 001111110100. \end{aligned}$$

4. She constructs the ciphertext $\boldsymbol{c}^{'}$

$$\begin{aligned} \boldsymbol{c}^{'} &= \boldsymbol{m}(p_1) + \boldsymbol{s}(p_2) \\ &= 101110111011 + 001111110100 \\ &= 100001001111, \end{aligned}$$

   and sends this to Bob.

Decryption

1. Bob uses his private key to obtain $\boldsymbol{s}$

$$\begin{aligned} \boldsymbol{s} &= \boldsymbol{c'}(Q) \\ &= 100001001111(Q) \\ &= 10011. \end{aligned}$$

2. Using $\boldsymbol{s}$, he constructs $\boldsymbol{s}(pub_2)$ using his public key (pub_2)

$$\begin{aligned} \boldsymbol{s}(pub_2) &= 10011(pub_2) \\ &= 001111110100. \end{aligned}$$

3. He obtains the codeword $\boldsymbol{c}$

$$\begin{aligned} \boldsymbol{c} &= \boldsymbol{c}^{'} + \boldsymbol{s}(pub_2) \\ &= 100001001111 + 001111110100 \\ &= 101110111011. \end{aligned}$$

4. He computes $\boldsymbol{m}SG$

$$\begin{aligned} \boldsymbol{m}SG &= \boldsymbol{c}P^{-1} \\ &= 000011110001. \end{aligned}$$

5. He decodes $\boldsymbol{m}SG$ to obtain $\boldsymbol{m}S$

$$\boldsymbol{m}S = 1000111.$$

6. He then recovers the message $\boldsymbol{m}$ using the inverse of S

$$\begin{aligned} \boldsymbol{m} &= 1000111(S^{-1}) \\ &= 1101001. \end{aligned}$$

3.2 Performance and security analysis

The McEliece cryptosystem is vulnerable to ciphertext distinguishability attacks, decoding attack, and structural attacks [26] [39] [55]. The primary objective of a decoding attack is to recover the plaintext from a given ciphertext without possessing the corresponding decryption key. Decoding attacks exploit vulnerabilities in the encryption scheme. In a distinguishability attack, the objective is to distinguish the plaintext corresponding to a given ciphertext. In a structural attack, the objective is to compromise the cryptosystem by obtaining information such as the generator matrix or

the private key. By gaining access to the generator matrix, an attacker can potentially derive the structure of the code and exploit any vulnerabilities in the system. Therefore, the complexity of the generator matrix and private key generation process are crucial in resisting structural attacks. Thus, the encryption scheme should resist such attacks and make it computationally infeasible to recover the plaintext without knowledge of the corresponding decryption key. In [26] [55], it was shown that the likelihood of a successful attack against the McEliece cryptosystem is $\binom{n-t}{k}/\binom{n}{k}$ where t is the error correction capability of the code $C(n,k)$. This probability can be made negligible with an appropriate choice of the code parameters n, k, and t.

A drawback of the McEliece cryptosystem is the large key size required to provide sufficient security. Thus, the tradeoff between security and performance is important when considering this cryptosystem for an application. The probability of a successful attack against the proposed scheme is

$$\frac{1}{2^{k(n-k)}} << \binom{n-t}{k}/\binom{n}{k},$$

which can be made negligible with an appropriate choice of parameters. Further, the security is not affected by the error correction capability of the code. The parameters $n = 256$ and $k = 128$ are suitable as $n - k \geq 128$, and result in public and private key sizes that are smaller than with the McEliece cryptosystem. Suitable Goppa code parameters for the McEliece cryptosystem are $n = 1024$, $k = 524$, and $t = (n-k)/\log_2 n = 50$ [43]. The public and private keys in the proposed cryptosystem are represented by the following matrices. SGP with dimensions $k \times n$, $L^{-1}(H^{-1})^T P$ with dimensions $(n-k) \times n$, S with dimensions $k \times k$, G with dimensions $k \times n$, P with dimensions $n \times n$, and $P^{-1}H^T L$ with dimensions $n \times (n-k)$. Hence, the total key size is $3n^2 + k^2 = 13 \times 2^{14}$ bits, including n^2 public keys and $2n^2 + k^2$ private keys. For the McEliece cryptosystem, the corresponding matrices are SGP with dimensions $k \times n$, S with dimensions $k \times k$, G with dimensions $k \times n$, and P with dimensions $n \times n$, so the total key size is $(n+k)^2$ [39]. Table 1 compares the sizes of the public and private keys with the McEliece and proposed code-based cryptosystems for the

Table 3.1: Key Size Comparison (kB)

Scheme	McEliece	Proposed
Public Key	65.5	8.0
Private Key	227.0	18.0
Public and Private Keys	292.5	26.0

given parameters.

3.3 Conclusion

The McEliece cryptosystem has been shown to be secure against quantum attacks but it has drawbacks including problems with the error correction capability of the chosen code. In addition, the use of a binary Goppa code increases complexity and results in a large key size which limits the practical applications. Thus, a code-based cryptosystem based on a random inverse parity check matrix was introduced to address these limitations. It was shown that this cryptosystem is secure against structural, ciphertext distinguishability and decoding attacks. Furthermore, the security is better than that of the McEliece cryptosystem for the same code parameters. The proposed cryptosystem offers smaller key sizes, which effectively mitigates the primary drawback of the McEliece code-based cryptosystem. The reduced key sizes associated with the proposed cryptosystem make it better suited for practical applications such as with resource constrained devices.

Chapter 4

A New Code-Based Digital Signature Based on the McEliece Cryptosystem

Digital signature schemes are widely used in many day to day applications, primarily to ensure the authenticity, integrity, and non-repudiation of digital documents and transactions. The CFS digital signature scheme is a widely recognized code-based digital signature scheme that relies on the Niederreiter cryptosystem. However, it is not often used due to the significant computation time required for signing. This is because the number of valid codewords is smaller than the size of the vector space [23]. Consequently, there is a possibility that a signature may not be decodable [21]. This results in an increase in the computation time required for signing because the algorithm needs to locate a valid signature that can be subsequently verified. In this chapter, a new code-based digital signature is proposed which is based on the McEliece cryptosystem. Key generation involves the construction of a public key using randomly generated inverse matrices. The signing algorithm has lower complexity and requires less computation time than the CFS scheme to sign a document. Furthermore, the verification algorithm is able to detect forgeries. It is also shown that the proposed scheme is secure against public key structural attacks.

4.1 Proposed code-based digital signature

The proposed code-based digital signature scheme encompasses key generation, signing, and verification algorithms. Within this framework, a code-based cryptosystem is employed with a parity check-based random inverse matrix, constituting the core of the public key infrastructure. First, a document is hashed to compress its size to n bits where n is the length of the code used in the cryptosystem. The proposed code-based digital signature algorithms are as follows.

Proposed Code-based Digital Signature Algorithms

- Key Generation: $(pk, sk) \leftarrow Gen(\lambda)$ where λ denotes the key generation scheme.
- Document/Message Signing: $\sigma \leftarrow Sign(sk, pk, \boldsymbol{doc})$ where σ and $\boldsymbol{doc}$ denote the signature and document, respectively.
- Signature Verification: $Ver(\sigma, pk, \boldsymbol{doc}) \in 0, 1$.

The following matrices are used in the proposed public key algorithm.

- G, a generator matrix of size $k \times n$.
- H, a parity check matrix of size $(n - k) \times n$.
- S, a non-singular scrambling matrix of size $k \times k$.
- P, a permutation matrix of size $n \times n$.
- L, a non-singular matrix of size $(n - k) \times (n - k)$.

The following subsection presents an algorithm to randomly construct an inverse matrix.

4.1.1 Random inverse matrix

As shown in Chapter 2, both matrix H and its inverse H^{-1} can be partitioned into two distinct parts, B_1, B_2 and A_1, A_2. This satisfies $HH^{-1} = I_m$, $m = n - k$

$$HH^{-1} = B_1A_1 + B_2A_2 = I_m,$$

where A_1 is randomly selected and A_2 is constructed using

$$A_2 = B_2^{-1}(B_1A_1 + I_m).$$

Theorem 2. The inverse of the matrix H is not unique, and the probability of constructing an inverse of H^{-1} equal to H is negligible.

Proof: The non-square matrix H has full rank and size $(n - k) \times n$. Hence, its inverse has $n - k$ columns, each of which can take 2^k different values. The number of valid H^{-1} matrices is then $2^{k\times(n-k)}$ [39]. Therefore, each H^{-1} has $2^{k\times(n-k)}$ distinct inverse matrices. Consequently, the probability of constructing an inverse of H^{-1} equal to H is $2^{-(k\times(n-k))}$, which is negligible for reasonable values of n and k.

4.1.2 Key generation

The proposed algorithm generates a public key (pk) and a private key (pr). The public key is shared within the network while the private key is kept secret.

Key Generation Algorithm $Gen(\lambda)$

---

1. Obtain a generator matrix G and corresponding parity matrix H for $C(n, k)$.
2. Select a random H^{-1} from the $2^{k\times(n-k)}$ choices using a random matrix A_1 and the corresponding matrix A_2
   $H^{-1} = (\frac{A_1}{A_2})$.

3. As in the McEliece cryptosystem, use the generator matrix G, the scrambling matrix S and the permutation matrix P to mask G
   $p_1 = G' = SGP$.

4. Use the non-singular random matrix L and P to mask H^{-1}
   $p_2 = L^{-1}(H^{-1})^T P$.

5. Verification of the digital signatures requires
   $p_3 = P^{-1}(H^{-1}H)^T P$.

6. Construct a parity check matrix corresponding to $G' = SGP$
   $Q = H'^T = P^{-1}H^T L$, $H' = L^T H(P^{-1})^T$.

7. Public key: $pk \leftarrow (p_1, p_2, p_3)$.

8. Private key: $pr(sk) \leftarrow (S^{-1}, P^{-1}, G, Q)$, where sk is the private key.

Lemma 1. The public key $pk = (p_1, p_2, p_3)$ satisfies the following

$$(p_1)(p_3) = \mathbf{0} \tag{4.1}$$
$$(p_2)(p_3) = p_2 \tag{4.2}$$
$$(p_3)(p_3) = p_3. \tag{4.3}$$

Proof: For (4.1)

$$\begin{aligned}(p_1)(p_3) &= (SGP)(P^{-1}(H^{-1}H)^T P) \\ &= SG(H^{-1}H)^T P \\ &= SG(H)^T(H^{-1})^T P \\ &= S(GH^T)(H^{-1})^T P \\ &= \mathbf{0}.\end{aligned}$$

For (4.2)

$$\begin{aligned}(p_2)(p_3) &= (L^{-1}(H^{-1})^T P)(P^{-1}(H^{-1}H)^T P) \\ &= L^{-1}(H^{-1})^T (H^{-1}H)^T P \\ &= L^{-1}(H^{-1}HH^{-1})^T P \\ &= L^{-1}(H^{-1})^T P \\ &= p_2.\end{aligned}$$

For (4.3)

$$\begin{aligned}(p_3)(p_3) &= (P^{-1}(H^{-1}H)^T P)(P^{-1}(H^{-1}H)^T P) \\ &= P^{-1}(H^{-1}H)^T (H^{-1}H)^T P \\ &= P^{-1}(H^{-1}HH^{-1}H)^T P \\ &= P^{-1}(H^{-1}H)^T P \\ &= p_3.\end{aligned}$$

The following lemma provides the relationship between the private and public keys.

Lemma 2. The public key $pk = (p_1, p_2, p_3)$ and the private key (Q) are related as follows

$$(p_1)(Q) = \mathbf{0} \tag{4.4}$$

$$(p_2)(Q) = \boldsymbol{I} \tag{4.5}$$

$$(p_3)(Q) = Q \tag{4.6}$$

$$(Q)(p_2) = p_3. \tag{4.7}$$

Proof: For (4.4)

$$\begin{aligned}(p_1)(Q) &= (SGP)(P^{-1}H^T L) \\ &= S(GH^T)L \\ &= \mathbf{0}.\end{aligned}$$

For (4.5)

$$\begin{aligned}(p_2)(Q) &= (L^{-1}(H^{-1})^T P)(P^{-1}H^T L)\\ &= L^{-1}(H^{-1})^T H^T L\\ &= L^{-1}(HH^{-1})^T L\\ &= \boldsymbol{I}.\end{aligned}$$

For (4.6)

$$\begin{aligned}(p_3)(Q) &= (P^{-1}(H^{-1}H)^T P)(P^{-1}H^T L)\\ &= P^{-1}(H^{-1}H)^T H^T L\\ &= P^{-1}(HH^{-1}H)^T L\\ &= P^{-1}(H)^T L\\ &= Q.\end{aligned}$$

For (4.7)

$$\begin{aligned}(Q)(p_2) &= (P^{-1}H^T L)(L^{-1}(H^{-1})^T P)\\ &= P^{-1}H^T(H^{-1})^T P\\ &= P^{-1}(H^{-1}H)^T P\\ &= p_3.\end{aligned}$$

4.1.3 Signing algorithm

The signing algorithm of the proposed signature scheme uses both keys to sign a document as follows.

Signing Algorithm $Sign(sk, pk, \boldsymbol{doc})$

---

1. Use the hash function to compress the size of the document to n bits
   $h(\boldsymbol{doc}) \leftarrow$ hash document $\boldsymbol{doc}$
   $h(h(\boldsymbol{doc})) \leftarrow$ hash $h(\boldsymbol{doc})$.

2. Let $\boldsymbol{s}$ denote an $n-k$ bit vector such that
   $\boldsymbol{s} \leftarrow h(\boldsymbol{doc})(Q)$.

3. Construct a codeword $\boldsymbol{c}$ using $h(\boldsymbol{doc})$ and $\boldsymbol{s}$
   $\boldsymbol{sig}SGP \leftarrow h(\boldsymbol{doc}) + \boldsymbol{s}(p_2)$.

4. Decode the codeword $\boldsymbol{c}$ to obtain $\boldsymbol{sig}$
   $\boldsymbol{sig}SG \leftarrow (\boldsymbol{sig}SGP)(P^{-1})$
   $\boldsymbol{sig}S \leftarrow$ decode $\boldsymbol{sig}SG$
   $\boldsymbol{sig} \leftarrow (\boldsymbol{sig}S)(S^{-1})$.

5. Use $h(h(\boldsymbol{doc}))$ and the private key sk to construct the $n-k$ bit vector $\boldsymbol{d}$
   $\boldsymbol{d} \leftarrow h(h(\boldsymbol{doc}))(Q) + \boldsymbol{s}$.

6. Output $\sigma = (\boldsymbol{sig}, \boldsymbol{d})$ and send $(\boldsymbol{sig}, \boldsymbol{d})$ and the document $\boldsymbol{doc}$ to the receiver for signature verification.

Theorem 3. $h(\boldsymbol{doc}) + \boldsymbol{s}(p_2)$ is a valid codeword of the code $C(n,k)$ with generator matrix $G' = SGP$.

Proof: Matrices S and P have full rank as they are non-singular matrices. Therefore, the rank of SGP is k and the rank of $P^{-1}H^TL$ is $n-k$. Further, the row vectors of SGP and the column vectors of $P^{-1}H^TL$ are orthogonal. Therefore, $P^{-1}H^TL$ generates the nullspace of the space spanned by SGP. Hence, the transpose of $P^{-1}H^TL$ is a parity check matrix for the code generated by SGP.

For a codeword $\boldsymbol{c} \in C(n,k)$, $\boldsymbol{c}H'^T = \boldsymbol{0}$ where $G' = SGP = p_1$ is the generator matrix and $H'^T = P^{-1}H^TL = Q$ is the parity check matrix

$$\boldsymbol{c} = \boldsymbol{sig}SGP = h(\boldsymbol{doc}) + \boldsymbol{s}(p_2).$$

The vector $\boldsymbol{s}$ is equal to $h(\boldsymbol{doc})(Q)$ so

$$\begin{aligned} \boldsymbol{sig}SGP &= h(\boldsymbol{doc}) + h(\boldsymbol{doc})(Q)(p_2) \\ \boldsymbol{sig}SGP &= h(\boldsymbol{doc}) + h(\boldsymbol{doc})(p_3). \end{aligned}$$

Therefore, $\boldsymbol{c}H'^T = (\boldsymbol{sig}SGP)(Q)$ and

$$\begin{aligned} \boldsymbol{c}H'^T &= h(\boldsymbol{doc})(Q) + h(\boldsymbol{doc})(p_3)(Q) \\ &= h(\boldsymbol{doc})(Q) + h(\boldsymbol{doc})(Q) \\ &= \boldsymbol{0}. \end{aligned}$$

4.1.4 Verification algorithm

The verification algorithm of the proposed code-based digital signature scheme is as follows.

Verification Algorithm $Ver(\sigma, pk, \boldsymbol{doc})$

---

1. Use the hash function $h(\cdot)$ to hash the received document to construct $h(\boldsymbol{doc})$ and $h(h(\boldsymbol{doc}))$, and assign
   $a \leftarrow \boldsymbol{sig}SGP$.

2. Use the public key (p_2, p_3) and $\boldsymbol{d}$ to compute $v_1 = \boldsymbol{s}(p_2)$ which is an n bit vector
   $v_1 \leftarrow h(h(\boldsymbol{doc}))(p_3) + \boldsymbol{d}(p_2)$.

3. Use the public key (p_3) to compute $v_2 = \boldsymbol{s}(p_2)$ which is an n bit vector
   $v_2 \leftarrow h(\boldsymbol{doc})(p_3)$.

4. If the condition
   $$v_1 = v_2,$$
   is not true, verification fails.

5. Use $\boldsymbol{s}(p_2)$ and $h(\boldsymbol{doc})$ to compute
   $\boldsymbol{c} \leftarrow h(\boldsymbol{doc}) + \boldsymbol{s}(p_2)$.

6. If $a = \boldsymbol{c}$
   $$Ver(\sigma, pk, \boldsymbol{doc}) = 1$$
   otherwise
   $$Ver(\sigma, pk, \boldsymbol{doc}) = 0.$$

4.1.5 Digital signature example

Let $n = 12$ and $k = 7$ be the parameters of the code $C(n, k)$ with

$$G = \left(\begin{array}{cc|ccccc} & | & 0 & 1 & 1 & 1 & 0 \\ & | & 0 & 0 & 0 & 1 & 1 \\ & | & 0 & 1 & 0 & 0 & 1 \\ I_k & | & 0 & 0 & 1 & 1 & 0 \\ & | & 0 & 1 & 0 & 1 & 0 \\ & | & 1 & 0 & 0 & 1 & 0 \\ & | & 1 & 0 & 1 & 0 & 0 \end{array}\right),$$

$$H_{(n-k)\times n} = \left(\begin{array}{ccccccc|c} 0 & 0 & 0 & 0 & 0 & 1 & 1 & | \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & | \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & | \; I_{n-k} \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & | \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & | \end{array}\right).$$

Randomly select a matrix A_1 of size $k \times (n-k)$ and construct the corresponding matrix A_2 of size $(n-k) \times (n-k)$ to obtain the random inverse parity check matrix

$$H^{-1} = \begin{pmatrix} A_1 \\ - \\ A_2 \end{pmatrix},$$

$$A_1 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix},$$

$$A_2 = (B_2)^{-1}(B_1A_1 + I_{n-k}),$$

$$A_2 = (B_2)^{-1}\left(\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \times A_1 + I_{n-k}\right),$$

$$A_2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

The scrambling matrix is

$$S_{k\times k} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix},$$

the non-singular matrix is

$$L_{(n-k)\times(n-k)} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix},$$

and the permutation matrix is

$$P_{n\times n} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

1. Alice hashes a document ***doc*** using the hash function $h(\cdot)$ to obtain $h(\boldsymbol{doc})$ of length n bits
   $h(\boldsymbol{doc}) = 100011010010$ and $h(h(\boldsymbol{doc})) = 101001000101$.

2. Construct an $n-k$ bit vector $\boldsymbol{s}$ such that $\boldsymbol{s} = h(\boldsymbol{doc})(Q)$
   $\boldsymbol{s} = 11110$.

3. Construct a codeword $h(\boldsymbol{doc}) + \boldsymbol{s}(p_2)$ of the code $C(n,k)$

$$\begin{aligned} \boldsymbol{c} &= \boldsymbol{sig}SGP = h(\boldsymbol{doc}) + \boldsymbol{s}(p_2) \\ &= 100011010010 + (11110)(p_2) \\ &= 100011010010 + 011111011001 \\ &= 111100001011. \end{aligned}$$

4. Decode the codeword to obtain $\boldsymbol{sig} = 0000110$.

5. Decode $\boldsymbol{d}$ using sk and $\boldsymbol{s}$

$$\begin{aligned} \boldsymbol{d} &= h(h(\boldsymbol{doc}))(Q) + \boldsymbol{s} \\ &= (101001000101)(Q) + 11110 \\ &= 11011 + 11110 \\ &= 00101. \end{aligned}$$

6. Output $(\boldsymbol{sig}, \boldsymbol{d})$ along with the document $\boldsymbol{doc}$.

Bob verifies the signature as follows.

1. Use the hash function $h(\cdot)$ to hash the received document to construct $h(\boldsymbol{doc})$ and $h(h(\boldsymbol{doc}))$ and assign $\boldsymbol{a} = \boldsymbol{sig}SGP$
   $h(\boldsymbol{doc}) = 100011010010$
   $h(h(\boldsymbol{doc})) = 101001000101$
   $\boldsymbol{a} = \boldsymbol{sig}SGP = (0000110)(p_1)$
   $\boldsymbol{a} = 111100001011.$

2. Use the public key of Alice (p_2, p_3) and $\boldsymbol{d}$ to compute

$$\begin{aligned} v_1 &= h(h(\boldsymbol{doc}))(p_3) + \boldsymbol{d}(p_2) \\ &= 101001000101(p_3) + 00101(p_2) \\ &= 101001000101 + 110110011100 \\ &= 011111011001. \end{aligned}$$

3. Use the public key of Alice (p_3) to compute

$$\begin{aligned} v_2 &= h(\boldsymbol{doc})(p_3) \\ &= 100011010010(p_3) \\ &= 011111011001. \end{aligned}$$

4. Bob checks the condition $v_1 = v_2$. If it is not true, verification fails.

5. Having $v_1 = \boldsymbol{s}(p_2)$ and given $h(\boldsymbol{doc})$, Bob computes

$$\begin{aligned} \boldsymbol{c} &= h(\boldsymbol{doc}) + \boldsymbol{s}(p_2) \\ &= 100011010010 + 011111011001 \\ &= 111100001011. \end{aligned}$$

6. Verification is successful if $\boldsymbol{a} = \boldsymbol{c}$.

It is recommended that every time the same document is signed, the signing algorithm should generate a different digital signature. This can be achieved by simply concatenating an n bit random vector $\boldsymbol{r}$ with the document. However, this increases the size of the signature, and the random vector should be output along with $(\boldsymbol{sig}, \boldsymbol{d})$.

4.2 Performance and security analysis

As mentioned previously, the computation time and low success rate are the main obstacles to the widespread use of code-based signatures. On average, the CFS code-based signature algorithm will have to be executed $t!$ times to obtain a valid signature. Modified CFS schemes have been developed with smaller values of t to reduce $t!$ [44]. Table 1 compares the success rate of several code-based signature schemes and the proposed scheme in terms of signature generation and verification complexity. The approximate number of operations for signing and verification with the proposed scheme is $3n^2$, including the integrity check.

Scheme	Error correction capability (t)	Success rate	Signature generation complexity	Verification complexity
Proposed	any	1	$O(n^2)$	$O(n^2)$ (with integrity check)
CFS	50	$(50!)^{-1}$	$O(t! \times n^2)$	$O(n^2)$
Modified CFS	10	$(10!)^{-1}$	$O(t! \times n^2)$	$O(n^2)$

Table 4.1: Code-based Signature Success Rate and Complexity

This section presents an analysis of the correctness, integrity, and probability of attack success against the proposed scheme. To protect the integrity of the transmitted signature, the verification algorithm should detect if it has been changed after generation or if a different secret key has been used to construct the signature. The proposed verification algorithm constructs v_1 and v_2 as follows

$$\begin{aligned} v_1 &\leftarrow h(h(\boldsymbol{doc}))(p_3) + \boldsymbol{d}(p_2) \\ v_2 &\leftarrow h(\boldsymbol{doc})(p_3) \end{aligned}$$

Then we obtain

$$\begin{aligned} \boldsymbol{d} &= h(h(\boldsymbol{doc}))(Q) + \boldsymbol{s} \\ \boldsymbol{d}(p_2) &= (h(h(\boldsymbol{doc}))(Q) + \boldsymbol{s})(p_2) \\ \boldsymbol{d}(p_2) &= h(h(\boldsymbol{doc}))(Q)(p_2) + \boldsymbol{s}(p_2). \end{aligned}$$

From (4.7) in Lemma 2

$$v_1 = \boldsymbol{s}(p_2) = h(h(\boldsymbol{doc}))(p_3) + \boldsymbol{d}(p_2). \tag{4.8}$$

Then we obtain

$$\begin{aligned} \boldsymbol{sig}SGP &= h(\boldsymbol{doc}) + \boldsymbol{s}(p_2) \\ \boldsymbol{s}(p_2) &= \boldsymbol{sig}(p_1) + h(\boldsymbol{doc}) \\ \boldsymbol{s}(p_2)(p_3) &= \boldsymbol{sig}(p_1)(p_3) + h(\boldsymbol{doc})(p_3). \end{aligned}$$

From (4.1) and (4.2) in Lemma 1, $(p_1)(p_3) = \mathbf{0}$ and $(p_2)(p_3) = p_2$, so

$$v_2 = \boldsymbol{s}(p_2) = h(\boldsymbol{doc})(p_3). \tag{4.9}$$

Thus, the same value of $\boldsymbol{s}(p_2)$ is obtained using two different approaches. In addition, v_1 does not depend on the signature $\boldsymbol{sig}$ and v_2 does not depend on the secret key although the integrity condition is met when $v_1 = v_2$. If an adversary uses a private key other than the correct one or forges $\boldsymbol{sig}$, then the condition $v_1 = v_2$ will not be met so verification fails. Hence, the integrity of the signature is ensured by the condition $v_1 = v_2$.

There are several cryptanalysis attacks that an adversary can use in an attempt to break a signature scheme [7]. Public key and forgery attacks are common attacks against digital signatures. A public key attack (called a structural attack), is typically more successful than a forgery attack. In this attack, an adversary tries to discover the secret key, so the security of the public key is critical. To increase the security of the public key, the proposed algorithm masks the generator and parity check matrices. Encrypted vectors $\boldsymbol{s}$ and $\boldsymbol{d}$ are used which increases the security. Therefore an adversary cannot break the scheme and forge a signature that can be verified through generic, directed, or adaptive chosen-message attacks [53]. The unforgeability attack is as follows [51].

1. The challenger uses the given code $C(n, k)$ to generate a public key (pk) and secret key (sk), and then shares (pk) with an adversary.

2. The adversary (a polynomial-time probabilistic machine), provides chosen messages (documents) $(doc_1, \cdots, doc_q)$, and the challenger provides valid signatures $(\sigma_1, \cdots, \sigma_q)$ in response.

3. The challenger provides algorithm access to the adversary.

4. The adversary forges a new message and signature (doc^*, σ^*) and sends it to the challenger for verification. Note that (doc^*) does not belong to the previously chosen messages $(doc_1, \cdots, doc_q)$.

5. The adversary wins if the verification is successful $V(doc^*, \sigma^*) = 1$.

An algorithm is secure when the probability of success is sufficiently low (negligible)

$$Pr[(Adv, \gamma) = 1] < \epsilon(\gamma),$$

where Adv denotes the adversary and γ denotes the security parameter [51, 52]. Consider the steps above. Let (sk_2) be the key the adversary uses to sign a document and outputs $(\boldsymbol{sig}, \boldsymbol{d})$ to be verified by the challenger. The challenger uses the verification algorithm and reaches Step 4

$$v_1 = v_2,$$

$$\boldsymbol{sig}(p_1) + h(\boldsymbol{doc}) = h(h(\boldsymbol{doc}))(p_3) + \boldsymbol{d}(p_2).$$

The left side $(h(\boldsymbol{doc})(p_3))$ is independent of the adversary secret key (sk_2) while $\boldsymbol{d}$ on the right side was constructed using the adversary secret key during the signing process. Then

$$\boldsymbol{d} = (h(h(\boldsymbol{doc})) + h(\boldsymbol{doc}))(sk_2),$$

and therefore

$$\begin{aligned} \boldsymbol{sig}(p_1) + h(\boldsymbol{doc}) &= h(h(\boldsymbol{doc}))(p_3) + \boldsymbol{d}(p_2) \\ \boldsymbol{sig}(p_1) + h(\boldsymbol{doc}) &= h(h(\boldsymbol{doc}))(p_3) + (h(h(\boldsymbol{doc}) + h(\boldsymbol{doc}))(sk_2)(p_2), \end{aligned}$$

so

$$\boldsymbol{sig}(p_1) + h(\boldsymbol{doc}) + h(h(\boldsymbol{doc}))(p_3) = (h(h(\boldsymbol{doc}) + h(\boldsymbol{doc}))(sk_2)(p_2),$$

and multiplying both side with p_3 gives

$$\boldsymbol{sig}(p_1)(p_3) + h(\boldsymbol{doc})(p_3) + h(h(\boldsymbol{doc}))(p_3)(p_3) = (h(h(\boldsymbol{doc}) + h(\boldsymbol{doc}))(sk_2)(p_2)(p_3)$$

$$(p_3) = (sk_2)(p_2).$$

From (4.7) in Lemma 2, this is true if and only if $(sk_2) = (sk)$.

Suppose an adversary selects $(L^{-1}(A^{-1})^T P)^{-1}$ as the secret key. Then

$$\begin{aligned}(sk_2)(p_2) &= (L^{-1}(A^{-1})^T P)^{-1}(L^{-1}(H^{-1})^T P) \\ &= (P^{-1}((A^{-1})^{-1})^T L)(L^{-1}(H^{-1})^T P) \\ &= P^{-1}(H^{-1}((A^{-1})^{-1}))^T P,\end{aligned}$$

so if $(A^{-1})^{-1} = A = H$, $(sk_2)(p_2)$ would be equal to (p_3) and the condition for the signed document is satisfied. In this case, the adversary can forge a signature with respect to the given public key and so is the winner in which case

$$Pr[(Adv, \gamma) = 1] \leftarrow Pr[sk_2 = sk] \leftarrow Pr[(A^{-1})^{-1} = A] < \epsilon(\gamma)$$

The matrix L is a square matrix so the inverse L^{-1} has the same size. However, the parity check matrix H is a full rank non-square matrix of size $(n-k) \times n$, so A should be a full rank matrix with the same dimensions. As noted in Theorem 2, the inverse of H is not unique. Hence, the probability that $(H^{-1})^{-1} = H$ is negligible, so the proposed scheme is secure against structural attacks.

Theorem 4. The inverse of the matrix $L^{-1}(A^{-1})^T P$ is not unique and the probability of constructing a particular inverse of $L^{-1}(A^{-1})^T P$ is negligible.

Proof: The matrix $(A^{-1})^T$ has full rank and size $(n-k) \times n$. Therefore, the public key $L^{-1}(A^{-1})^T P$ is a full rank matrix, and its inverse has $n-k$ columns, each of which can have 2^k different values. The number of valid inverse matrices is then $2^{k\times(n-k)}$ [39]. Hence, the probability of constructing a particular inverse of the public key $L^{-1}(A^{-1})^T P$ is $\frac{1}{2^{k\times(n-k)}}$, which is negligible for reasonable values of n and k.
As a result of Theorem 4, the probability of an adversary constructing a secret key using the public key is $2^{-(k\times(n-k))}$. Therefore, the probability of an adversary signing a document that can be verified is negligible

$$Pr[(Adv, \gamma) = 1] < \frac{1}{2^{k\times(n-k)}}.$$

Hence, the probability of the adversary forging the signature by accessing the algo-

rithm is also negligible, so the proposed algorithm is secure.

4.3 Conclusion

The CFS digital signature scheme was examined and its drawbacks described. An approach to overcoming these drawbacks was proposed. It was also shown that code-based digital signature schemes require significant computation time if the ciphertexts are only part of the vector space. In particular, the CFS scheme requires $t!$ executions on average, where t is the error correction capability of the code, to obtain a valid signature. The proposed code-based digital signature scheme provides a practical solution to this drawback. It was shown to be secure in that an adversary cannot forge a signature that can be verified and it is secure against a structural public key attack. Further, the probability of constructing the secret key from the public key was shown to be negligible. Moreover, results were presented which show that the proposed scheme requires less computation time for signing than other code-based digital signature schemes.

Chapter 5

A Code-Based Cryptosystem with Dual Inverse Matrix

This chapter introduces a new cryptosystem based on a dual inverse matrix. This matrix, denoted A, functions as both the transpose and inverse of the matrix H. This chapter presents a code-based encryption scheme that incorporates the dual inverse matrix A in the key generation algorithm. The security analysis demonstrates that this cryptosystem is resilient against threats typically encountered by existing code-based cryptosystems, including the information set decoding attack. Furthermore, a digital signature scheme is presented that utilizes the dual inverse matrix A. It established key relations and an integrity check to mitigate security concerns and prevent forgery prior to the verification process. This approach differs from using a random inverse of the parity check matrix employed in the preceding chapters.

5.1 Code-based encryption with dual inverse matrix

The proposed algorithm employs a dual inverse matrix for code-based cryptosystem key generation, encryption and decryption.

Code-Based Cryptosystem

Key Generation: $(pk, sk) \leftarrow Gen(\lambda)$ where λ is the algorithm input.

Encryption: $\boldsymbol{c'} \leftarrow Enc(\boldsymbol{m}, pk)$ where $\boldsymbol{c'}$ is the ciphertext for the plaintext $\boldsymbol{m}$.

Decryption: $\boldsymbol{m} \leftarrow Dec(\boldsymbol{c'}, sk)$.

5.1.1 Dual inverse matrix A

The proposed algorithm employs a matrix A with dimensions $n \times (n-k)$. This matrix is both a transpose of the parity check matrix (H^T) and an inverse of the parity check matrix (H^{-1}), i.e. $HA = I_{n-k}$ and $GH^T = GA = \mathbf{0}$. Thus, A can be constructed using H^T and an auxiliary matrix P' that satisfies $A = H^T P'$ giving

$$GA = \mathbf{0} \text{ and } GH^T = \mathbf{0}, \text{ so } A = H^T P',$$

and

$$HA = H(H^T P') = (HH^T)P' = I_{n-k}.$$

Then $P' = (HH^T)^{-1}$ so A can be formed if HH^T is non-singular. The proposed code-based cryptosystem has three-tuple public keys, and encryption and decryption uses the dual inverse matrix A.

5.1.2 Key generation

The algorithm $Gen(\lambda)$ uses the generator matrix G and matrix A to generate the keys. To provide resistance against structural attacks, the algorithm scrambles G and H and employs the following matrices.

An $n \times (n-k)$ dual inverse matrix A.
A $k \times k$ non-singular scrambling matrix S.
An $n \times n$ non-singular or permutation matrix P.

An $(n-k) \times (n-k)$ non-singular matrix L.

Key Generation $Gen(\lambda)$

---

1. Use the parity check matrix corresponding to G to construct $P' = (HH^T)^{-1}$.
2. Form the matrix $A = H^T P'$.
3. Construct $pk_1 = G^{'} = SGP$ to mask the generator matrix G.
4. Construct $pk_2 = L^{-1}HP$ to mask the parity check matrix H.
5. Obtain the transpose of H' using G' $Q = H'^T = ((AL)^T(P^{-1})^T)^T = P^{-1}AL$.
6. Form the public key $pk \leftarrow (pk_1, pk_2)$.
7. Form the private key $pr(sk) \leftarrow (S^{-1}, P^{-1}, G, Q)$.

Key Generation Algorithm

The generator matrix G and parity check matrix H are masked using a random non-singular scrambling matrix S and a random permutation matrix P, respectively, and the dual inverse matrix A is masked using a random non-singular matrix L and P.

Lemma 3. The relations between the public key $pk = (pk_1, pk_2)$ and the private key Q are as follows

$$pk_1 Q_A = \mathbf{0} \tag{5.1}$$

$$pk_2 Q_A = \boldsymbol{I} \tag{5.2}$$

Proof: For (5.1)

$$\begin{aligned} pk_1 Q_A &= (SGP)(P^{-1}AL) \\ &= S(GA)L \\ &= \mathbf{0}. \end{aligned}$$

For (5.2)

$$\begin{aligned} pk_2 Q_A &= (L^{-1}HP)(P^{-1}AL) \\ &= L^{-1}(HA)L \\ &= \boldsymbol{I}. \end{aligned}$$

5.1.3 Encryption scheme

The encryption scheme takes the plaintext as input and transforms it into the corresponding ciphertext.

Encryption $Enc(\boldsymbol{m}, pk)$

---

1. Construct the codeword by computing the product of pk_1 and the plaintext $\boldsymbol{c} = \boldsymbol{m}(pk_1) = \boldsymbol{m}(SGP)$.
2. Randomly generate an $n - k$ bit vector $\boldsymbol{s}$.
3. Construct the error vector using $\boldsymbol{s}(L^{-1}HP)$ $\boldsymbol{e} = \boldsymbol{s}(pk_2)$.
4. Form the ciphertext $\boldsymbol{c'} = \boldsymbol{c} + \boldsymbol{e} = \boldsymbol{m}(pk_1) + \boldsymbol{s}(pk_2)$.
5. Output $\boldsymbol{c'}$.

Note that if the weight of $\boldsymbol{s}(pk_2)$ is smaller than minimum distance of the code, another random vector $\boldsymbol{s}$ should be selected.

5.1.4 Decryption scheme

The decryption scheme takes the ciphertext as input and converts it to the corresponding plaintext.

Decryption $Dec(\boldsymbol{c'}, sk)$

1. Construct the vector $\boldsymbol{s}$ using the private key Q and ciphertext $\boldsymbol{c'}$

$$\boldsymbol{s} = \boldsymbol{c'}(P^{-1}AL).$$

$$\begin{aligned} \boldsymbol{c'} &= \boldsymbol{m}(SGP) + \boldsymbol{s}(L^{-1}HP) \\ \boldsymbol{c'}(P^{-1}AL) &= (\boldsymbol{m}(SGP) + \boldsymbol{s}(L^{-1}HP))(P^{-1}AL) \\ &= \boldsymbol{m}(SGP)(P^{-1}AL) + \boldsymbol{s}(L^{-1}HP)(P^{-1}AL) \\ &= \boldsymbol{m}S(GA)L + \boldsymbol{s}L^{-1}(HA)L \\ &= \boldsymbol{0} + \boldsymbol{s}(I) \\ &= \boldsymbol{s}. \end{aligned}$$

2. Use $pk_2 = L^{-1}HP$ to construct $\boldsymbol{s}(L^{-1}HP)$.

3. Form $\boldsymbol{c}$

$$\boldsymbol{c} = \boldsymbol{c'} + \boldsymbol{s}(L^{-1}HP).$$

4. Find the plaintext $\boldsymbol{m}$ by decoding $\boldsymbol{c}$

$$\begin{aligned} \boldsymbol{m}SG &= (\boldsymbol{m}SGP)(P^{-1}) \\ \boldsymbol{m}S &= \text{ decode } \boldsymbol{m}SG \\ \boldsymbol{m} &= (\boldsymbol{m}S)(S^{-1}). \end{aligned}$$

5.1.5 Example

Consider a code $C(n, k)$ with parameters $n = 12$ and $k = 5$

generator matrix

$$G = \left(\begin{array}{c|ccccccc} & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ I_k & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right),$$

parity check matrix

$$H^T = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

non-singular matrix

$$L = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix},$$

dual inverse matrix

$$A_{n\times(n-k)} = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix},$$

permutation matrix

$$P_{n\times n} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix},$$

and scrambling matrix

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

1. Alice uses her public key $pk_1 = SGP$ to encrypt the message $\boldsymbol{m} = 11010$ to obtain the ciphertext

$$\boldsymbol{c} = \boldsymbol{m}(pk_1) = 110011101101.$$

2. Alice selects a random $n - k$ bit vector $\boldsymbol{s} = 1001011$.

3. Alice constructs an error pattern using $\boldsymbol{s}$ and her public key pk_2

$$\boldsymbol{e} = \boldsymbol{s}(pk_2) = 1001011(pk_2) = 000000111011.$$

4. Alice forms the ciphertext using the public keys pk_1 and pk_2

$$\begin{aligned} \boldsymbol{c}^{'} &= \boldsymbol{m}(pk_1) + \boldsymbol{s}(pk_2) \\ &= 110011101101 + 000000111011 \\ &= 110011010110. \end{aligned}$$

5. The ciphertext is transmitted by Alice to Bob.

Bob receives $\boldsymbol{c'}$ and decrypts it as follows.

1. Bob uses his private key to obtain

$$\begin{aligned} \boldsymbol{s} = \boldsymbol{c'}(Q) & \\ &= 110011010110(Q) \\ &= 1001011. \end{aligned}$$

2. Bob uses his public key $L^{-1}HP$ and $\boldsymbol{s}$ to get

$$\begin{aligned} \boldsymbol{e} &= \boldsymbol{s}(pk_2) \\ &= 1001011(pk_2) \\ &= 000000111011. \end{aligned}$$

3. Bob constructs

$$\begin{aligned} \boldsymbol{c} &= \boldsymbol{c}^{'} + \boldsymbol{s}(pk_2) \\ &= \boldsymbol{m}(pk_1) \\ &= 110011101101. \end{aligned}$$

4. Bob uses P^{-1} to obtain

$$\begin{aligned} \boldsymbol{m}(SG) &= \boldsymbol{m}(SGP)(P^{-1}) \\ &= 110110101100. \end{aligned}$$

5. Bob decodes $\boldsymbol{m}(SG)$ to get

$$\boldsymbol{m}(S) = 11011.$$

6. Bob uses S^{-1} to obtain the plaintext from the codeword

$$\begin{aligned} \boldsymbol{m} &= \boldsymbol{m}(S)(S^{-1}) \\ &= 11011(S^{-1}) \\ &= 11010. \end{aligned}$$

5.1.6 Performance and security analysis

The key size, security, and complexity are now analyzed. The public and private keys in the proposed cryptosystem have the following matrices. SGP with dimensions $k \times n$, $L^{-1}HP$ with dimensions $(n-k) \times n$, S with dimensions $k \times k$, G with dimensions $k \times n$, P with dimensions $n \times n$, and $P^{-1}AL$ with dimensions $(n-k) \times n$. Hence,

Table 5.1: Key Size Comparison (kB)

Scheme	McEliece	Proposed with Matrix A
Public Key	65.5	8.0
Private Key	227.0	18.0
Public and Private Keys	292.5	26.0

the total key size is $3n^2 + k^2 = 13 \times 2^{14}$ bits. Table 5.1 gives the sizes of the public and private keys using the dual inverse matrix A with the McEliece and proposed code-based cryptosystems for the given parameters.

The McEliece code-based cryptosystem has been considered a strong candidate for post-quantum cryptography, mainly due to its robustness against cryptographic attacks posed by the emergence of quantum computing. However, like many cryptographic schemes, it is not immune to specific attacks such as information set decoding (ISD) which targets the underlying code structure. This attack is a significant threat to the McEliece cryptosystem as it exploits the inherent structure of the error-correcting code used in the scheme. In this attack, an adversary forms an information set of syndromes by selecting a subset of syndromes generated from a received ciphertext. This attack relies on the assumption that random error vectors can be eliminated through the linear structure of the code. A successful ISD attack allows adversaries to decode the message without knowledge of the private key, compromising the security of the cryptosystem and posing a major threat to message confidentiality. Hence, selecting suitable code parameters, including code length, dimension, and error weight, is necessary to make it resistant to information set attacks. ISD was initially introduced in 1962 [54], and subsequently improved in 1987 [55], 1988 [59], and 1989 [56], primarily in the context of binary linear codes. In 2010, the ISD algorithm was extended to linear codes over arbitrary finite fields [57].
ISD attacks target the underlying code structure [54, 55, 56, 57]. They rely on the assumption that random error vectors can be eliminated considering the linearity of

the code. The selection of k bits from a ciphertext $\boldsymbol{c'} \in F_2^n$ to create a vector $\boldsymbol{c}_k \in F_2^k$ was considered in [55]. If the k bits selected are error-free, then the product of $\boldsymbol{c}_k$ and the inverse of the matrix G'_k, which are the corresponding k columns of G', gives the message

$$\boldsymbol{c_k} G_k'^{-1} = \boldsymbol{m}.$$

Then a codeword is constructed using $\boldsymbol{c_k} G_k'^{-1}$ and this is added to the ciphertext to obtain $\eta = \boldsymbol{c} + \boldsymbol{c}_k G_k'^{-1} G'$. If $\boldsymbol{c}_k G_k'^{-1}$ is the message $\boldsymbol{m}$, then η will have Hamming weight $w(\eta) \leq t$, otherwise $w(\eta) \geq 2t$ [55]. The proposed encryption algorithm establishes a mapping between an n bit error pattern $\boldsymbol{e} \in F_2^n$ and an $n-k$ bit random vector $\boldsymbol{s} \in F_2^{n-k}$. The McEliece encryption algorithm employs an error vector of maximum weight t while the proposed approach ensures that the weight of the error vector exceeds the minimum distance of the code. As discussed in [58], attempts to identify k error-free bits when the Hamming weight of the error vector is greater than the minimum distance of the code have proven unsuccessful. This is primarily due to a substantial number of codeword bits differing from the ciphertext bits. Conversely, Bob can use the received ciphertext to construct the random vector and recover the message. The proposed encryption algorithm involves the generation of a random vector which makes it intractable for an adversary to differentiate between ciphertexts and gain knowledge about the underlying messages. This provides excellent security against attacks that recover the message from the ciphertext. Further, the weight of this vector is independent of t, so the proposed algorithm provides a higher level of security. The total key size of the proposed cryptosystem is $3n^2 + k^2$, consisting of public key size n^2 and private key size $2n^2 + k^2$. Consequently, the proposed cryptosystem effectively addresses the primary drawback of the McEliece cryptosystem, notably reducing the public a key size from 292.5 to 26.0 kilobytes. As a result, the proposed cryptosystem with a smaller key size offers better security than the McEliece cryptosystem.

The complexity is now considered based on the number of arithmetic operations. The McEliece cryptosystem constructs the ciphertext $c' = m \times G' + e$ where m

is a message of length k, G' has size $k \times n$, and e has length n. The number of arithmetic operations required is $2nk$. The ciphertext in the proposed cryptosystem is $c' = m \times pub_1 + s \times pub_2$ where m has length k', pub_1 has size $k' \times n'$, s has length $n' - k'$, and pub_2 has size $(n' - k') \times n'$. The number of arithmetic operations required is then $2n'(n' - 1)$. Therefore, considering the difference in lengths between the McEliece and proposed cryptosystems, $n = 4n'$, when encrypting a message of length $k = 524$ the proposed cryptosystem requires 51% fewer operations.

5.2 Code-based digital signature scheme with dual inverse matrix

The proposed code-based digital signature scheme is a probabilistic algorithm for key generation, signing, and verification. The dual inverse matrix A described below is used in the key generation, signing, and verification algorithms.

The proposed code-based digital signature algorithms are as follows.

Proposed Code-based Digital Signature Algorithms

- Key Generation: $(pk, sk) \leftarrow Gen(\lambda)$ where λ denotes the key generation scheme.
- Document/Message Signing: $\sigma \leftarrow Sign(sk, pk, \boldsymbol{doc})$ where σ and $\boldsymbol{doc}$ denote the signature and document, respectively.
- Signature Verification: $Ver(\sigma, pk, \boldsymbol{doc}) \in 0, 1$.

5.2.1 Key generation

As mentioned above, the matrix A satisfies $GA = \mathbf{0}$ and $HA = I_{n-k}$. This property is leveraged in the construction of public and private keys to obtain a digital signature. In addition to the generator matrix G, parity check matrix H, and dual inverse

matrix A, the key generation algorithm employs a $k \times k$ scrambling matrix S, an $(n-k) \times (n-k)$ non-singular matrix L, and an $n \times n$ non-singular matrix P.

Key Generation Algorithm

---

1. Given a generator matrix G for $C(n,k)$ with non-singular HH^T and $A = H^T P'$.
2. Construct $P' = (HH^T)^{-1}$.
3. As in the McEliece cryptosystem, use the generator matrix G, the scrambling matrix S, and the permutation matrix P to mask G

$$pk_1 = G^{'} = SGP.$$

4. Use the non-singular matrix L and P to mask H

$$pk_2 = L^{-1}HP.$$

5. Verification of a digital signature requires

$$pk_3 = P^{-1}AHP.$$

6. Construct a parity check matrix $H^{'}$ corresponding to $G^{'} = SGP$

$$Q_A = H^{'T} = ((AL)^T(P^{-1})^T)^T = P^{-1}AL.$$

7. The public and private keys are defined as

$$p_{ub}k \leftarrow (pk_1, pk_2, pk_3), p_{ri}k \leftarrow (S, P, G, Q_A).$$

As explained earlier, to increase security, the generator matrix G of the code $C(n,k)$ is masked using two random non-singular matrices S and P. The parity check matrix H

is also concealed using the non-singular matrices L and P. The verification algorithm uses pk_3 to validate the digital signatures, and ensure their integrity and authenticity.

Theorem 5. The public key $L^{-1}HP$ has many inverses and the probability of constructing a particular inverse of $L^{-1}HP$ can be made negligible.

Proof: The parity check matrix is a full rank matrix and is not unique [39]. The inverse of this matrix has $n-k$ columns, each of which can have 2^k different values, so the number of inverse matrices is $2^{k\times(n-k)}$ [39]. The public key $L^{-1}HP$ is a full rank matrix, so the probability of constructing a particular inverse of $L^{-1}HP$ is $\frac{1}{2^{k\times(n-k)}}$ which is negligible for appropriate values of n and k.

5.2.2 Signing algorithm

The proposed signature scheme uses both keys to sign a document as follows.

Signing Algorithm

---

1. Hash document $\boldsymbol{doc}$ and hash the result to n bits
   $h(\boldsymbol{doc}) \leftarrow$ hash document $\boldsymbol{doc}$
   $h(h(\boldsymbol{doc})) \leftarrow$ hash $h(\boldsymbol{doc})$.

2. Let $\boldsymbol{s}$ be the $n-k$ bit vector given by
   $\boldsymbol{s} \leftarrow h(\boldsymbol{doc})(P^{-1}AL)$.

3. Compute
   $\boldsymbol{sig}SGP \leftarrow h(\boldsymbol{doc}) + \boldsymbol{s}(L^{-1}HP)$.

4. Decode the codeword $\boldsymbol{c}$ to obtain $\boldsymbol{sig}$
   $\boldsymbol{sig}SG \leftarrow (\boldsymbol{sig}SGP)(P^{-1})$
   $\boldsymbol{sig}S \leftarrow$ decode $\boldsymbol{sig}SG$
   $\boldsymbol{sig} \leftarrow (\boldsymbol{sig}S)(S^{-1})$.

5. Construct the $n-k$ bit vector $\boldsymbol{d}$
   $\boldsymbol{d} \leftarrow h(h(\boldsymbol{doc}))(P^{-1}AL) + \boldsymbol{s}$.

6. Output $(\boldsymbol{sig}, \boldsymbol{d})$ and document $\boldsymbol{doc}$.

Theorem 6. $h(\boldsymbol{doc}) + \boldsymbol{s}(L^{-1}HP)$ is a valid codeword of the code $C(n,k)$ with generator matrix $G' = SGP$.

The row vectors of SGP and the column vectors of $P^{-1}AL$ are orthogonal as $P^{-1}AL$ generates the nullspace of the code generated by SGP. Consequently, the transpose of $P^{-1}AL$ is a parity check matrix corresponding to the generator matrix SGP. It satisfies $\boldsymbol{c}(H')^T = \boldsymbol{0}$ for a codeword $\boldsymbol{c} \in C(n,k)$

$$\boldsymbol{c} = \boldsymbol{sig}SGP = h(\boldsymbol{doc}) + \boldsymbol{s}(L^{-1}HP).$$

Using $\boldsymbol{s} = h(\boldsymbol{doc})(P^{-1}AL)$, obtain

$$\boldsymbol{sig}SGP \;=\; h(\boldsymbol{doc}) + h(\boldsymbol{doc})(P^{-1}AL)(L^{-1}HP),$$

and then

$$\begin{aligned}
\boldsymbol{c}(H')^T &= \boldsymbol{sig}SGP((AL)^T(P^{-1})^T)^T \\
&= h(\boldsymbol{doc})(P^{-1}AL) + h(\boldsymbol{doc})(P^{-1}AL)((L^{-1}HP)(P^{-1}AL)) \\
&= h(\boldsymbol{doc})(P^{-1}AL) + h(\boldsymbol{doc})(P^{-1}AL)(L^{-1}HAL) \\
&= h(\boldsymbol{doc})(P^{-1}AL) + h(\boldsymbol{doc})(P^{-1}AL)\boldsymbol{I} \\
&= \boldsymbol{0}.
\end{aligned}$$

5.2.3 Verification algorithm

The verification algorithm ensures the authenticity and integrity of the signature.

Verification Algorithm

---

1. Use the hash function $h()$ to hash the received document to construct $h(\boldsymbol{doc})$ and $h(h(\boldsymbol{doc}))$

$$\boldsymbol{a} \leftarrow \boldsymbol{sig}SGP.$$

2. Use the public key and $\boldsymbol{d}$ to obtain $v_1 = \boldsymbol{s}(L^{-1}HP)$

$$\begin{gathered}\boldsymbol{d} = h(h(\boldsymbol{doc}))(P^{-1}AL) + \boldsymbol{s} \\ \boldsymbol{d}(L^{-1}HP) = (h(h(\boldsymbol{doc}))(P^{-1}AL) + \boldsymbol{s})(L^{-1}HP) \\ \boldsymbol{d}(L^{-1}HP) = h(h(\boldsymbol{doc}))(P^{-1}AL)(L^{-1}HP) + \boldsymbol{s}(L^{-1}HP),\end{gathered}$$

so

$$v_1 \leftarrow h(h(\boldsymbol{doc}))(P^{-1}AHP) + \boldsymbol{d}(L^{-1}HP).$$

3. Use the public key $(P^{-1}AHP)$ to obtain $v_2 = \boldsymbol{s}(L^{-1}HP)$ which is an n-bit vector

$$\begin{gathered}\boldsymbol{sig}SGP = h(\boldsymbol{doc}) + \boldsymbol{s}(L^{-1}HP) \\ \boldsymbol{s}(L^{-1}HP) = \boldsymbol{sig}(SGP) + h(\boldsymbol{doc}) \\ \boldsymbol{s}(L^{-1}HP)(P^{-1}AHP) = \boldsymbol{sig}(SGP)(P^{-1}AHP) + h(\boldsymbol{doc})(P^{-1}AHP),\end{gathered}$$

so

$$v_2 \leftarrow \boldsymbol{s}(L^{-1}HP) = h(\boldsymbol{doc})(P^{-1}AHP).$$

4. The integrity condition is satisfied if

$$v_1 = v_2,$$

otherwise, verification fails.

5. Use $v_1 = \boldsymbol{s}(L^{-1}HP)$ and $h(\boldsymbol{doc})$ to compute

$$\boldsymbol{c} \leftarrow h(\boldsymbol{doc}) + \boldsymbol{s}(L^{-1}HP).$$

6. Verification is successful if

$$\boldsymbol{a} = \boldsymbol{c},$$

otherwise, it fails.

Changes made by an adversary should be detected by the verification algorithm. The

integrity condition in Step 4 checks the validity of the signature. Note that v_1 does not depend on the signature $\boldsymbol{sig}$ and v_2 does not depend on the private key, but the integrity condition is satisfied if $v_1 = v_2$.

5.2.4 Example

Let $n = 12$ and $k = 5$ with

$$G = \left(\begin{array}{c|cccccc} & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ I_k & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right),$$

and

$$H = \left(\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 1 & \\ 0 & 1 & 0 & 1 & 1 & \\ 0 & 0 & 1 & 0 & 0 & \\ 1 & 0 & 1 & 1 & 1 & I_{n-k} \\ 0 & 1 & 1 & 0 & 0 & \\ 1 & 0 & 1 & 1 & 0 & \\ 1 & 0 & 1 & 0 & 1 & \end{array}\right),$$

$$A_{n\times(n-k)} = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix},$$

$$P = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$L = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}, S = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Alice generates the signature as follows.

1. Use the hash function $h()$ with the document $\boldsymbol{doc}$ to obtain

$$\begin{aligned} h(\boldsymbol{doc}) &= 100110010001 \\ h(h(\boldsymbol{doc})) &= 110001110111. \end{aligned}$$

2. Construct the $(n-k)$-bit vector $\boldsymbol{s} = h(\boldsymbol{doc})(P^{-1}AL)$

$$\boldsymbol{s} = 0101111.$$

3. Construct a codeword $h(\boldsymbol{doc}) + \boldsymbol{s}(L^{-1}HP)$ of the code $C(n,k)$

$$\begin{aligned} \boldsymbol{c} &= \boldsymbol{sig}SGP = h(\boldsymbol{doc}) + \boldsymbol{s}(L^{-1}HP) \\ &= 100110010001 + (0101111)(L^{-1}HP) \\ &= 100110010001 + 000110110010 \\ &= 100000100011. \end{aligned}$$

4. Decode the codeword to obtain

$$\boldsymbol{sig} = 01010.$$

5. Use Q_A and $\boldsymbol{s}$ to obtain

$$\begin{aligned}\boldsymbol{d} &= h(h(\boldsymbol{doc}))(P^{-1}AL) + \boldsymbol{s} \\ &= (110001110111)(P^{-1}AL) + 0010111 \\ &= 1000110 + 0010111 \\ &= 1101001.\end{aligned}$$

6. Output $(\boldsymbol{sig}, \boldsymbol{d})$ along with the document $\boldsymbol{doc}$.

Bob verifies the signature as follows.

1. Use the hash function $h()$ and the received document to obtain

$$\begin{aligned}h(\boldsymbol{doc}) &= 100110010001 \\ h(h(\boldsymbol{doc})) &= 110001110111\end{aligned}$$

$$\boldsymbol{a} = \boldsymbol{sig}SGP = (0000110)(SGP) = 100000100011.$$

2. Use the public key of Alice and $\boldsymbol{d}$ to compute

$$\begin{aligned}v_1 &= h(h(\boldsymbol{doc}))(P^{-1}AHP) + \boldsymbol{d}(pk_2) \\ &= 110001110111(P^{-1}AHP) + 1101001(L^{-1}HP) \\ &= 110111000110 + 110001110100 \\ &= 000110110010.\end{aligned}$$

3. Use the public key of Alice to compute

$$\begin{aligned}v_2 &= h(\boldsymbol{doc})(P^{-1}AHP) \\ &= 100110010001(P^{-1}AHP) \\ &= 000110110010.\end{aligned}$$

4. Check the integrity condition $v_1 = v_2$. If it is met, continue, otherwise verification is failed.

5. Use $v_1 = \boldsymbol{s}(pk_2)$ and $h(\boldsymbol{doc})$ to compute $\boldsymbol{c}$

$$\begin{aligned} \boldsymbol{c} &= h(\boldsymbol{doc}) + \boldsymbol{s}(L^{-1}HP) \\ &= 100110010001 + 000110110010 \\ &= 10000010011. \end{aligned}$$

6. Verification is successful if $\boldsymbol{a} = \boldsymbol{c}$

$$Ver(\sigma, pk, \boldsymbol{doc}) = 1.$$

As previously mentioned, it is advisable that the signing algorithm produces a distinct digital signature each time a document is signed. One way to achieve this is by appending an n-bit random vector r to the document. However, this increases the signature size as it requires the inclusion of the random vector in the output alongside $(\boldsymbol{sig}, \boldsymbol{d})$.

5.2.5 Performance and security analysis

There are different types of cryptanalysis attack models that adversaries can use to discover the weaknesses of an algorithm, gain access to the contents of the messages/document, and break the secret key. The proposed algorithm masks the generator matrix using permutation and scrambling matrices.

The algorithm is secure when the probability of success is negligible

$$Pr[(Adv, \gamma) = 1] < \epsilon(\gamma),$$

where Adv denotes the adversary and γ denotes the security parameter. We analyze an adversary constructing the private key from the public key. The challenger provides full access to the adversary to input any document and receive the corresponding signature. Then the adversary uses its private-key (Q_{adv}) to sign a document and output $(\boldsymbol{sig}, \boldsymbol{d})$ to be verified by the challenger. The challenger uses the verification

algorithm and reaches Steps (2), (3) and (4) to challenge the integrity condition ($v_1 = v_2$). From the verification algorithm

$$v_1 \leftarrow \boldsymbol{s}(L^{-1}HP) = h(h(\boldsymbol{doc}))(P^{-1}AHP) + \boldsymbol{d}(L^{-1}HP),$$

and

$$v_2 \leftarrow \boldsymbol{s}(L^{-1}HP) = h(\boldsymbol{doc})(P^{-1}AHP).$$

Hence, v_2 does not depend on the adversary private key (Q_{adv}), while the adversary needs to construct $\boldsymbol{d}$ using its own private key to sign the document

$$\boldsymbol{d} = h(h(\boldsymbol{doc}))(Q_{adv}) + \boldsymbol{s}$$

$$\boldsymbol{d}(L^{-1}HP) = [h(h(\boldsymbol{doc})) + h(\boldsymbol{doc})](Q_{adv})(L^{-1}HP).$$

Therefore, for $v_1 = v_2$

$$[h(h(\boldsymbol{doc})) + h(\boldsymbol{doc})](P^{-1}AHP) = [h(h(\boldsymbol{doc})) + h(\boldsymbol{doc})](Q_{adv})(L^{-1}HP)$$

$$P^{-1}AHP = (Q_{adv})(L^{-1}HP).$$

This is valid if and only if $Q_{adv} = Q$. Consider that an adversary selects $(L^{-1}H''P)^{-1}$ as the private key, so

$$\begin{aligned} P^{-1}AHP &= (Q_{adv})(L^{-1}HP) \\ &= (L^{-1}H''P)^{-1}(L^{-1}HP) \\ &= (P^{-1}(H'')^{-1}L)(L^{-1}HP) \\ &= P^{-1}(H'')^{-1}HP. \end{aligned}$$

If $(H'')^{-1} = A$, then Q_{adv} would be equal to the private key Q and the document signed by the adversary can be verified successfully. As mentioned in Theorem 5, the inverse of a full rank non-square matrix is not unique and there are $2^{k\times(n-k)}$ possible inverses. Therefore, the probability of $(H'')^{-1}$ being equal to A and consequently, the probability of a key selected by an adversary being equal to the private key $Q_{adv} = Q$

is insignificant for appropriate values of n and k. Thus, the probability that an adversary signs a verifiable document is negligible

$$Pr(Adv, \gamma) = 1 \leftarrow Pr[(sk_{adv}) = sk]$$

$$Pr[(sk_{adv}) = sk] \leftarrow Pr[(H'')^{-1} = A]$$

$$Pr[(Adv, \gamma) = 1] < \epsilon(\gamma).$$

The size of the public and private keys in the cryptosystem proposed in Chapter 3 is $3n^2 + k^2$. Adding pk_3, the total key size becomes $4n^2 + k^2$, with public key size $2n^2$ and private key size $2n^2 + k^2$. The size of the signature and the speed of the signing process are the main factors that influence the choice of a digital signature algorithm. Speed is important for applications such as online banking, e-commerce, and blockchains (Bitcoin, Ethereum). On average, the CFS code-based signature schemes require $t!$ executions to obtain a valid signature so the speed is proportional to the error correction capability of the code [44]. Table 5.2 compares the success rates and signature sizes of the proposed and lattice-based schemes. This shows that the proposed scheme has the smallest signature size and the highest success rate.
The primary competition for the proposed scheme comes from lattice-based signatures which are designed to resist attacks from quantum computers. However, the signatures obtained are large [46]. Table 5.2 compares the signature length of several lattice-based signature schemes (Bliss, qTesla, Dilithium), Hash-based (SPHINICS+), and that of the proposed scheme [45, 46, 47]. This shows that the proposed code-based scheme has smaller signatures, resulting in reduced computation time for the signing process. Signature size plays an important role in the selection of a digital signature scheme, making it a primary consideration. The proposed algorithm generates signatures with lengths 32 and 64 bytes, in contrast to alternatives like Bliss-IV, which has a signature of length 6556 bytes and a relatively low success rate of 0.19 [45], and qTesla-III with a signatures of length 2848 bytes [46, 47]. The signature size of SPHINCS+ [48] is also very large. Speed is a critical factor in numerous digital signature applications, including online banking, e-commerce, and blockchain

Table 5.2: Signature Size Comparison (bytes)

Scheme	Security (bits)	Success rate	Public key	Signature
Bliss-I [45][46]	128	0.63	2048	5734
Bliss-II [45][46]	128	0.14	2048	5120
Bliss-III [45][46]	160	0.36	3072	6144
Bliss-IV [45][46]	192	0.19	3072	6656
qTeslaIII [46][47]	256	1	3103	2908
SPHINICS+ [48]	128	1	32	7856
SPHINICS+ [48]	192	1	48	16224
SPHINICS+ [48]	256	1	64	29792
Dilithium-2 [49][50]	128	1	1312	2420
Dilithium-3 [49][50]	192	1	1952	3293
Dilithium-5 [49][50]	256	1	2592	4595
proposed $n = 256$	128	1	16384	32
proposed $n = 512$	256	1	32768	64

technologies like Bitcoin and Ethereum.

5.3 Conclusion

This chapter introduced a new public key generation algorithm based on the concept of a dual inverse matrix A where $GA = \mathbf{0}$ and $HA = \boldsymbol{I}$. A code-based cryptosystem that leverages the dual inverse matrix in constructing public keys was presented that increases the overall security of the cryptosystem. The security of this cryptosystem was assessed, and the results demonstrate its resilience against known attacks on code-based cryptosystems, including the information set decoding attack. Then, a code-based digital signature was presented which employs the dual inverse matrix in the key construction, signing and verification algorithms. It was shown that the proposed digital signature ensures security, as adversaries cannot forge a signature that can be verified. Furthermore, it was demonstrated that this digital signature scheme has a smaller signature size compared with other schemes.

Chapter 6

Summary and Future Work

This chapter provides a summary of the results presented in this dissertation to address the limitations of code-based cryptosystems. Furthermore, it outlines directions for future research to improve the methods introduced in this work and suggests additional applications.

6.1 Summary

In 1994, Shor illustrated the potential of quantum attacks to be a serious threat to widely used cryptographic primitives. Recent advances have emphasized this threat so the National Institute of Standards and Technology (NIST) is considering proposals for the post-quantum era. One approach to designing cryptographic primitives resilient against quantum attacks is to utilize encryption schemes that have shown resistance to this threat. Code-based cryptography, like the McEliece cryptosystem, are among these schemes. However, they are not employed in practical applications because of the large sizes of the public and private keys, and for the code-based digital signatures, the complexity of the signing process and low success rates.

In Chapter 2, a scheme was introduced to construct $2^{m(n-m)}$ distinct inverse matrices for any full-rank, non-square matrix with m rows and n columns, $n > m$. Additionally,

a scheme was introduced to generate a random inverse from the $2^{m(n-m)}$ choices. The proposed approach has lower computational complexity than the Moore-Penrose and Gauss-Jordan methods.

In Chapter 3, an efficient code-based cryptosystem, derived from the McEliece cryptosystem, was introduced. This system uses the random inverse algorithm from Chapter 2 to construct keys. The randomness of the inverse matrix increases the complexity for potential adversaries. This approach addresses the primary drawback of the McEliece cryptosystem by reducing the size of the public key from 292.5 kB to 26.0 kB.

Chapter 4 introduced a new code-based digital signature scheme using the McEliece cryptosystem and the random inverse matrix algorithm. This practical code-based digital signature offers several advantages, including reduced complexity and faster signing with a high success rate. Moreover, it ensures integrity checks with a negligible probability of an adversary successfully forging the signature. The signature generation complexity of the proposed digital signature was compared with the CFS digital signature.

Chapter 5 introduced the concept of a dual inverse matrix that serves as both the transpose and inverse of the parity check matrix. This matrix was used in a key generation algorithm. A code-based cryptosystem was presented that incorporates the dual inverse matrix in encryption and decryption. The security analysis focused on assessing the resilience of the proposed cryptosystem against information set decoding attacks. Furthermore, a digital signature was introduced that utilizes the dual inverse matrix in signing and verification. The success rate and signature size of the proposed digital signature were compared with other schemes including lattice-based schemes Bliss-I to Bliss-IV, qTesla III, and Dilithium-2 to Dilithium-5.

6.2 Future work

The results in this dissertation suggest several avenues of future research. Future work may involve exploring the following directions.

Data Storage Security: Future research can investigate how the proposed cryptographic techniques, in particular the use of random inverse matrices in cryptosystems, can enhance security and performance in domains such as data storage and machine learning, especially in applications handling large-size matrices.

Integration with Blockchain Technologies: Exploring the integration of the proposed cryptographic techniques with blockchain technologies is another area for future work. Blockchain, as a decentralized and secure distributed ledger, relies on cryptographic algorithms to ensure the integrity of transactions. This aligns with the security objectives of code-based cryptography. The robustness of code-based cryptography makes it a candidate for enhancing the security of blockchain systems.

Randomized Algorithms and Data Structures: The proposed method for generating random inverse matrices can be considered in the context of randomized algorithms and data structures.

Bibliography

[1] R. Rivest, A. Shamir, and L. Adleman "A method for obtaining digital signatures and public-key cryptosystem," *Communications of the ACM*, vol. 21, pp. 120–126, 1978.

[2] National Institute of Standards and Technology, "Digital Signature Standard (DSS)," *NIST Federal Information Processing Standards*, Gaithersburg, MD, USA, FIPS 186–2, 2000.

[3] R. E. Blahut, Cryptography and Secure Communication, Cambridge University Press, Cambridge, UK, 2014.

[4] B. Li, B. Lie, Y. Zhang, Yunlong, and S. Lei, "A novel and high-performance modular square scheme for elliptic curve cryptography over GF," *IEEE Transactions on Circuits and Systems II, Express Briefs*, vol. 66, no. 4, pp. 647–651, 2019.

[5] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings of the Annual Symposium on Foundations of Computer Science*, Santa Fe, NM, USA, pp. 124–134, 1994.

[6] M. Baldi, "Post-quantum cryptographic schemes based on codes," in *Proceedings of the International Conference on High Performance Computing and Simulation*, Genoa, Italy, pp. 908–910, 2017.

[7] T. N. R. Rao and K. H. Nam, "Private-key algebraic-coded cryptosystem," in *Advances in Crytpology, Proceedings of CRYPTO*, Lecture Notes in Computer Science, vol. 263, Springer, Berlin, Germany, pp. 35–48, 1986.

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

[9] T. N. R. Rao, "Joint encryption and error correction schemes," in *Proceedings of the Annual International Symposium on Computer Architecture*, NY, USA, pp. 240–241, 1984.

[10] M. R. Albrecht and D. J. Bernstein, "Classic McEliece (merger of classic McEliece and NTS-KEM)," *NIST*, Post-Quantum Cryptography, Round 4 Submissions, 2022.

[11] N. Aragon and P. Barreto, "BIKE: Bit flipping key encapsulation," *NIST*, Post-Quantum Cryptography, Round 4 Submissions, 2022.

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

[13] P.-L. Cayrel and M. Meziani, "Post-quantum cryptography: Code-based signatures," in *Advances in Computer Science and Information Technology*, Lecture Notes in Computer Science, vol. 6059, Springer, Berlin, Germany, pp. 82–99, 2013.

[14] R. Nojima, H. Imai, K. Kobara, and K. Morozov, "Semantic security for the McEliece cryptosystem without random oracles," *Design, Codes and Cryptography*, vol. 49, no. 1-3, pp. 289–305, 2008.

[15] P.-L. Cayrel, P. Gaborit, and M. Girault, "Identity based identification and signature schemes using correcting codes," in *Proceedings of the International Workshop on Coding and Cryptography*, Versailles, France, pp. 69–78, 2007.

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

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

[18] H. Niederreiter, "Knapsack-type cryptosystems and algebraic coding theory," *Problems of Control and Information Theory*, vol. 15, pp. 159–166, 1986.

[19] N. Courtois, M. Finiasz, and N. Sendrier, "How to achieve a McEliece-based digital signature scheme," in *Advances in Cryptology, Proceedings of ASIACRYPT*, Lecture Notes in Computer Science, vol. 2248, Springer, Berlin, Germany, pp. 157–174, 2001.

[20] C. Paar and J. Pelzl, Understanding Cryptography, Springer, Berlin, Germany, pp. 150–153, 2010.

[21] S. B. Xu, J. M. Doumen, and H. C. A. Van Tilborg, "On the security of digital signature scheme based on error correcting codes," *Designs, Codes and Cryptography*, vol. 28, no. 2, pp. 187–199, 2003.

[22] W. Xinmei, "Digital signature scheme based on error correcting codes," *Electronics Letters*, vol. 26, no. 13, pp. 898–899, 1990.

[23] P. Cayrel, M. Meziani, T. H. Kim, and H. Adeli, "Post-quantum cryptography: Code-based signatures," in *Advances in Computer Science and Information Technology*, Lecture Notes in Computer Science, vol. 6059, Springer, Berlin, Germany, pp. 82–99, 2010.

[24] D. Forney, "Principles of digital communication II," *MIT*, MIT Open Courseware, 2005.

[25] S. B. Wicker, Error Control Systems for Digital Communication and Storage, Prentice-Hall, Englewood Cliffs, NJ, USA, p. 84, 1994.

[26] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, "Public-key encryption," ch. 8, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, USA, 1997.

[27] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, Jul. 1948.

[28] R. Acharya, Understanding Satellite Navigation, Academic Press, London, UK, 2014.

[29] A. Thomasian, *Storage Systems: Organization, Performance, Coding, Reliability, and Their Data Processing*, Academic Press, London, UK, 2011.

[30] S. Saraf, S. Dhingra, and G. Pinheiro, "Parallel algorithm for finding inverse of a matrix and its application in message sharing (coding theory)," *International Journal of Computer Applications*, vol. 136, no. 9, pp. 24–28, Feb. 2016.

[31] D. J. Bernstein, T. Lange and C. Peters, "Classical McEliece: conservative code-based cryptography," *Journal of Cryptography*, vol. 32, no. 3, pp. 937–965, 2019.

[32] P. S. Stanimirović and M. D. Petković, "Gauss–Jordan elimination method for computing outer inverses," *Applied Mathematics and Computation*, vol. 219, no. 9, pp. 4667–4679, Jan. 2013.

[33] N. Guglielmi, M. L. Overton, and G. W. Stewart, "An efficient algorithm for computing the generalized null space decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 36, no. 1, pp. 38–54, Jan. 2015.

[34] J. C. A. Barata and M. S. Hussein, "The Moore-Penrose pseudoinverse: A tutorial review of the theory," *Brazilian Journal of Physics*, vol. 42, no. 1-2, pp. 146–165, Apr. 2012.

[35] H. Chen and Y. Wang, "A family of higher-order convergent iterative methods for computing the Moore–Penrose inverse," *Applied Mathematics and Computation*, vol. 218, no. 8, pp. 4012–4016, Dec. 2011.

[36] J. Tapson and A. van Schaik, "Learning the pseudoinverse solution to network weights," *Neural Networks*, vol. 45, pp. 94–100, Sep. 2013.

[37] F. L. Tiplea and V. F. Dragoi "Generalized inverse based decoding," in *Proceedings of the IEEE International Symposium on Information Theory*, Espoo, Finland, pp. 2791–2796, Aug. 2022.

[38] F. Xin and H. George, "On the worst-case complexity of integer Gaussian elimination," in *Proceedings of the International Conference on Symbolic and Algebraic Computation*, Maui, HI, USA, pp. 28–31, July 1997.

[39] M. Esmaeili, "Application of linear block codes in cryptography," PhD Dissertation, Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada, 2019.

[40] R. W. Farebrother, "Linear least squares computations," *Statistics Textbooks and Monographs*, vol. 91, Taylor and Francis, London, UK, 1988.

[41] L. M, S. Picek, and R. Yorgova "On McEliece-type cryptosystems using self-dual codes with large minimum weight," *IEEE Access*, vol. 11, pp. 43511–43519, 2023.

[42] T. S. C. Lau, F. Ivanov, M. R. K. Ariffin, J.-J. Chin, and T. T. V. Yap, "New code-based cryptosystems via the IKKR framework," *Journal of Information Security and Applications*, vol. 76, art. 103530, 2023.

[43] H. Moufek, K. Guenda, and T. A. Gulliver, "A new variant of the McEliece cryptosystem based on QC-LDPC and QC-MDPC codes," *IEEE Communication Letters*, vol. 21, no. 4, pp. 714–717, 2017.

[44] M. Finiasz, "Parallel-CFS: Strengthening the CFS McEliece-based signature scheme," in *Selected Areas in Cryptography*, Lecture Notes in Computer Science, vol. 6544, Springer, Berlin, Germany, pp. 159–170, 2011.

[45] T. Pöppelmann, L. Ducas and T. Güneysu, "Enhanced lattice-based signatures on reconfigurable hardware," in *Cryptographic Hardware and Embedded Systems*,

Lecture Notes in Computer Science, vol. 8731, Springer, Berlin, Germany, pp. 353–370, 2014.

[46] J. Howe, T. Pöppelmann, M. O'Neill, E. O'Sullivan, and T. Güneysu, "Practical lattice-based digital signature schemes," *ACM Transaction on Embedded Computing Systems*, vol. 14, no. 3, pp. 1–24, 2015.

[47] D. Das, J. Hoffstein, J. Pipher, W. Whyte and Z. Zhang, "Modular lattice signatures, revisited," *Designs, Codes and Cryptography*, vol. 88, no. 3, pp. 505–532, 2020.

[48] National Institute of Standards and Technology, "SPHINICS+," *NIST*, Post-Quantum Cryptography, Round 3 Submissions, 2020.

[49] National Institute of Standards and Technology, "Crystal - Dilithium," *NIST*, Post-Quantum Cryptography, Round 3 Submissions, 2020.

[50] A.K. Pendey, A. Babati, B. Rajendran, S.D. Sudarsan and K.K. Soundra Pandian, "Cryptographic challenges and security in post quantum cryptography migration: A prospective approach," in *Proceedings of the International Conference on Public Key Infrastructure and its Applications*, Bangalore, India, 2023.

[51] D. Diemert, K. Gellert, T. Jager, and L. Lyu, "More efficient digital signatures with tight multi-user security," in *Public-Key Cryptography*, Lecture Notes in Computer Science, vol. 12711, Springer, Berlin, Germany, pp. 1–31, 2021.

[52] M. Bellare, C. Namprempre, and G. Neven, "Security proofs for identity-based identification and signature schemes," *Journal of Cryptology*, vol. 22, no. 1, pp. 1–61, 2009.

[53] S. Goldwasser, S. Micali, and R. Rivest, "A Digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 281–308, 1988.

[54] E. Prange, "The use of information sets in decoding cyclic codes," *IRE Transactions on Information Theory*, vol. 8, no. 5, pp. 5–9, 1962.

[55] P.J. Lee and E.F. Brickell, "An observation on the security of McEliece's public-key cryptosystem," in *Advances in Cryptology — EUROCRYPT'88*, Lecture Notes in Computer Science, vol. 330, Springer, Berlin, Germany, pp. 275–280, 1988.

[56] J. Stern, "A method for finding codewords of small weight," in *Coding Theory and Applications*, Lecture Notes in Computer Science, vol. 388, Springer, Berlin, Germany, pp. 106–113, 1989.

[57] C. Peters, "Information-set decoding for linear codes over F_q," in *Post-Quantum Cryptography, PQCrypto 2010*, Lecture Notes in Computer Science, vol. 6061, Springer, Berlin, Germany, pp. 25–28, 2010.

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

[59] J.S. Leon, "A probabilistic algorithm for computing minimum weights of large error-correcting codes," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 1354–1359, 1988.