Permutation error-correcting codes and their applications to public-key cryptography

A Thesis

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

by

Adarsh Srinivasan

Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

July, 2021

Supervisor: Ayan Mahalanobis

Certificate

This is to certify that this dissertation entitled **Permutation error-correcting codes and their applications to public-key cryptography** towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Adarsh Srinivasan at Indian Institute of Science Education and Research under the supervision of **Ayan Mahalanobis**, **Assistant Professor**, Department of Mathematics, during the academic year 2020-2021.

Ayan Mahalanobis

Committee:

Ayan Mahalanobis

Srinivas Kotyada

This thesis is dedicated to my late grandfather, KRS Iyengar, whose dedication, resilience and strength have always inspired me to reach for the stars

असतोमा सद्गमय ।

तमसोमा ज्योतिर्गमय ।

मृत्योर्मामृतं गमय ॥

ॐ शान्ति शान्ति शान्तिः ।।

From ignorance, lead me to truth
From darkness, lead me to light
From death, lead me to immortality
Let there be peace

Declaration

I hereby declare that the matter embodied in the report entitled **Permutation error-correcting codes and their applications to public-key cryptography** are the results of the work carried out by me at the Department of Mathematics, Indian Institute of Science Education and Research, Pune, under the supervision of **Ayan Mahalanobis** and the same has not been submitted elsewhere for any other degree.

Adarsh Srinivasan

Acknowledgments

This work could not have been completed without the constant encouragement, support and guidance from my mentors, family and friends. I have grown tremendously as a researcher over the past couple of years and I have Dr. Ayan Mahalanobis to thank for that. He encouraged me to think independently and ensured that I gave my best at all times. I would also like to express my gratitude to Prof. Srinivas Kotyada for evaluating this thesis and my mid year presentation. I would like to thank Upendra Kapshikar for the discussions, that helped me refine this work. IISER Pune provided me with an excellent environment to grow as person. It enabled me to think deeply on various mathematical and scientific concepts. Much of this thesis was done working remotely from home, due to the pandemic. My family has been a tremendous source of strength and support throughout this time. I consider myself fortunate to have an amazing group of friends who have always been there for me.

Abstract

In this thesis, we study some computational problems in permutation group theory and their applications to public-key cryptography. The primary goal of this thesis is to come up with a cryptosystem similar to the McEliece cryptosystem using permutation groups instead of vector spaces over finite fields. Like vector spaces, permutation groups too have been explored as a setting for error-correcting codes. These objects are called permutation codes. We propose a framework for such a cryptosystem and also come up with several classical attacks on it. We prove that the cryptosystem using transitive permutation groups is quantum-secure. We also explore using the permutation codes proposed by Bailey and Cameron in our cryptosystem. Although our cryptosystem using permutation codes that exist currently is insecure, we hope that this work would encourage research on coming up with new classes of permutation codes with efficient decoding algorithms.

Contents

Chapter 1

Introduction

Vector spaces over finite fields are used as error-correcting codes in information theory and have been extensively studied for over half a century since the seminal work of Shannon and Hamming [39, 24]. Decoding a general linear code is an **NP**-complete problem. This means that generic algorithms would only run in time exponential in the dimension of the code, which would make them unsuited for most applications. For practical applications, there exist specialised algorithms that are fast but only work for some specific families of codes. The decoding problem has been extensively studied, both in the general case and for families of codes such as Goppa codes and Reed Solomon codes [12]. Apart from a plethora of real world applications, like in CD drives, deep space communication and information transfer in general, linear error-correcting codes also have the potential to be useful in the field of cryptography. Code-based cryptosystems like the McEliece and Neiderreiter cryptosystems have been widely studied as candidates for post-quantum cryptosystems [31, 33]. Their security relies on the hardness of decoding a general linear code. Despite having much faster encryption and decryption speeds than most other public key cryptosystems such as RSA and ElGamal, they have not been used widely due to their very large key sizes. The advent of quantum computing has changed a lot of things, however. Both the RSA and ElGamal cryptosystems are susceptible to an attack using the technique of *Quantum Fourier Sampling* (QFS). This means that once functional quantum computers are built, our public key cryptosystems need to be replaced with post-quantum alternatives. Code-based cryptosystems appear to be capable of resisting this attack [18]. Hence, they have attracted a lot of attention recently.

Another possible setting for error-correcting codes is over permutations of finite sets. We can define a Hamming metric between permutations similar to how it is defined for vectors over finite fields, and hence, we can perform nearest neighbour decoding. This application was first studied by Blake, Cohen and Deza in the 1970's [25, 13], who studied lists of permutations (known as permutation arrays) as error correcting codes. Bailey and Cameron were the first to use permutation groups as codes and use the group structure to come up with decoding algorithms [20, 19, 8, 26]. Decoding algorithms for linear codes rely extensively on linear algebraic techniques. Permutation codes on the other hand, use techniques and tools from computational group theory. Permutation group algorithms is one of the most well studied aspects of computational group theory and provides a rich source of easy and hard problems. Like the corresponding problem in linear codes, decoding a general permutation code is **NP**-hard. Unlike for linear codes however, specialised decoders for families of permutation codes are not very well studied. One of the reasons for this might be the lack of applications where permutation codes are advantageous over their more widely used linear counterparts. To our knowledge, the only known decoding algorithms are those proposed by Bailey in his PhD thesis [20]. In this thesis, we explore whether permutation codes could be useful in public key cryptography.

1.1 Goals and original contribution

The primary goal of this thesis is to explore the use of permutation codes in public key cryptography. The security of public key cryptosystems are based on computational hardness assumptions. The assumption we use is that decoding in a general permutation code is hard. To start with, we prove in Chapter 2 that this problem is **NP**-hard. While a proof for this already exists in literature, we provide a new proof with some potential advantages. In Chapter 4, we define a public key cryptosystem similar to the McEliece cryptosystem using permutation codes. This cryptosystem uses a specific permutation code with an efficient decoding algorithm in its private key. The security of this cryptosystem would depend, among other assumptions, on the hardness of the general decoding problem for permutation codes. We come up with classical attacks on this cryptosystem and discuss its security in the face of these attacks. In Chapter 5, we prove that our proposed cryptosystem using some classes of permutation codes is resistant to QFS attacks. In fact, due to the lack of structure in permutation groups compared to linear codes, the QFS attack employed on the

McEliece cryptosystem might not even be possible on our cryptosystem. While we haven't been able to find a permutation code such that the cryptosystem using it is classically secure, we hope that this work will provide a motivation for research into finding new families of permutation codes with new decoding algorithms. The use of non-abelian permutation groups instead of linear codes could lead to significant reductions in key-size while retaining the very fast encryption and decryption speeds enjoyed by linear-code based cryptosystems. This is significant as one of the main complaints about the classic McEliece cryptosystem is its very large key size.

1.2 Notation and definitions used

In this thesis, we are dealing with finite permutation groups. We use the symbol Ω to denote the set $\{1, \ldots, n\}$. We use the notation S_n to denote the symmetric group on this set, the group whose elements are all the bijections from Ω to itself, with composition of functions being the group operation. Consider a group G acting faithfully on Ω[1]. Such a group can be identified with a subgroup of S_n and is called a *permutation group*. Throughout this thesis, we use right group actions and right cosets.

1.3 Structure of the thesis

In Chapter 2, we define two important computational problems from permutation group theory, the membership testing problem and the subgroup distance problem. We also introduce the fundamental data sets and objects used to manipulate permutation groups. In Chapter 3, we discuss Bailey and Cameron's work on using permutation groups as error correcting codes. The fundamental notions and definitions are defined. We study some decoding algorithms and discuss their performance. In Chapter 4, we define a public key cryptosystem similar to the McEliece cryptosystem and discuss its classical security. In Chapter 5, we review the fundamental problems and techniques in quantum computing, specifically the hidden subgroup and hidden shift problems and the technique of quantum Fourier sampling to solve them. We use the hidden shift problem to come up with an attack on our cryptosystem and

[1]The action of G on Ω is called faithful if, for all distinct $g, h \in G$, there exists $\omega \in \Omega$ such that $\omega \cdot g \neq \omega \cdot h$

show that our cryptosystem can indeed be made resistant to this attacks.

Chapter 2

Permutation group algorithms

In this chapter, we study some computational problems in permutation group theory. Permutation group algorithms are one of the most well studied and well developed aspects of computational group theory and provides a rich source of easy and hard problems. This makes it an excellent setting to develop public-key cryptosystems, which rely on computational hardness assumptions for security and on effective and fast algorithms for encryption and decryption. We focus on the problem of membership testing and introduce the fundamental data structures used to solve this problem. This problem can be solved using a non-trivial but very efficient algorithm. We later consider a generalisation of the membership testing problem, called the subgroup distance problem. Unlike the membership testing problem, this is an **NP**-hard problem and does not have a polynomial time solution.

Groups can be presented in multiple ways. Permutation groups are one of the oldest and most commonly used representations of groups. The input for a permutation group algorithm would be a list of generators S for the group $G = \langle S \rangle$. The representation of a permutation group using generators is highly efficient, since a very small number of permutations in S_n can describe a group of size up to $n!$. The symmetric group itself, for example can be generated using just two generators. In fact, many interesting groups, including all finite simple groups can be generated using just two generators. The succinctness of this representation however, means that even basic problems such as membership testing and finding the order of G requires nontrivial algorithms. For a resource on the computational problems and techniques in permutation group theory, we refer to Akos Seress' book on

permutation group algorithms [38].

Much of this chapter is a literature review of fundamental concepts in permutation group algorithms. In Theorem 2.2.1 however, we provide a new proof for the **NP**-hardness of the subgroup distance problem.

2.1 Bases and strong generating sets

Let G be a permutation group acting on $\Omega = \{1, 2, \ldots, n\}$ described by a set of generators. Consider the following problem of membership testing.

Problem 2.1.1 (Membership testing)**.** *Consider a permutation $\sigma \in S_n$ and a set of generators S for $G \leq S_n$. Is σ a member of G?*

If σ is a member of the group G, this means that it can be expressed as a product of elements in S. As G is non-abelian however, this word may even be exponential in length. Sims [43, 42] was the first to address this problem, introducing the notions of a base and a strong generating set. The idea is to obtain a new, slightly larger generating set for G with some additional structure so that every element of G can be expressed by a product of at most n generators.

Definition 2.1.1. *Consider a permutation group G acting on Ω. A* base *is a sequence of points $B = \{b_1, \ldots, b_m\}$ in Ω such that the only permutation in G to fix B pointwise is the identity.*

Given a base, we can define the following subgroup chain:

$$G = G^{[0]} \geq G^{[1]} \cdots \geq G^{[m]} = \{1\} \tag{2.1}$$

where $G^{[i]}$ is the pointwise stabilizer of $\{b_1, b_2, \ldots, b_i\}$ in G. We say that a base B is *irredundant* if $G^{[i]} \neq G^{[i+1]}$ for all i. We can now apply Lagrange's theorem repeatedly to obtain:

$$|G| = \prod_{i=0}^{m-1} \left|G^{[i]} : G^{[i+1]}\right| \tag{2.2}$$

It is easy to see, using the orbit stabilizer theorem, that the cosets of $G^{[i]} \mod G^{[i+1]}$ correspond to members of the orbit of b_i in $G^{[i-1]}$. Hence $\left|G^{[i]} : G^{[i+1]}\right| \leq n$. If the base is irredundant, $\left|G^{[i]} : G^{[i+1]}\right| \geq 2$. This means that $2^m \leq |G| \leq n^m$. Note the similarity with the concept of a basis of a vector space. In many ways, a base of a permutation group is similar to a basis of a vector space even though it is a much weaker concept. We now define a strong generating set:

Definition 2.1.2. *A strong generating set (SGS) for a group G relative to a base B is a generating set S for G such that $\left\langle S \cap G^{[i]} \right\rangle = G^{[i]}$ for $1 \leq i \leq m$.*

The orbits $b_i \cdot G^{[i-1]}$ are called the *fundamental orbits*, and the (right) *transversals* R_i are the set of (right) coset representatives for $G^{[i]} \mod G^{[i+1]}$, with the requirement that the representative for the coset $G^{[i+1]} \cdot 1 = G^{[i+1]}$ is the identity for all i. They can be computed by keeping track of the permutations of $G^{[i-1]}$ that take b_i to each point in the orbit. The orbits and the corresponding transversals can be computed using a type of breadth first search [38, Theorem 2.1.1].

Given a generating set for a permutation group, a base and a strong generating set can be constructed very efficiently using the *Schreier-Sims algorithm* [38, Chapter 4]. In computer algebra systems like GAP, a base, strong generating set, the stabilizer chains and transversals are stored as a recursive data structure called a stabilizer chain [21].

As an example, we consider the general linear group $GL(n, q)$. For the vector space $\mathbb{F}_q^n$, the general linear group associated with it, $\mathbb{F}_q^n$ consists of non singular $n \times n$ matrices over $\mathbb{F}_q$. As each matrix permutes the elements of the vector space and do not take non-zero vectors to $\mathbf{0}$, it can be considered as a permutation group of degree $q^n - 1$. We claim that a a set of vectors form a base for this permutation group if and only if they span $\mathbb{F}_q^n$.

Consider any set of row vectors that span $\mathbb{F}_q^n$. Any linear transformation that fixes each of these basis elements must be the identity transformation, because we can form a basis from a subset of these vectors. To prove the converse, Assume that there exists a base B for $\mathbb{F}_q^n$ consisting of some vectors which do not span the whole space. Hence, there exists the linear span of these vectors is a non-trivial subspace V of $\mathbb{F}_q^n$. We can now define a linear transformation that fixes every vector in V but is not the identity, contradicting the assumption that B is a base. We can consider the quotient space $\mathbb{F}_q^n/V$ and a non identity non-singular linear transformation S on it. $\mathbb{F}_q^n$ can be expressed as the $V \oplus \mathbb{F}_q^n/V$, with

$\mathbf{x_1}, \ldots, \mathbf{x_k}$ being a set of coset representatives. Consider any permutation σ of $\{1, \ldots, k\}$. We now define the following linear transformation. Any vector $\mathbf{x}$ can be expressed uniquely as $\mathbf{x'} + \mathbf{x_i}$ for some i and the transformation takes $\mathbf{x}$ to $\mathbf{x'} + \mathbf{x_{i \cdot \sigma}}$. It is easy to prove that this transformation is indeed linear and injective.

What we have proved so far is that a set of vectors form a base for $GL(q, n)$ if and only if they span the space. A base of the permutation group corresponds to a set of vectors that span the space. An irredundant base, on the other hand, corresponds to a basis of a vector space, which consists of vectors that not only span the space but are also linearly independent. One reason we chose this example is that it explicitly demonstrates the connections between these corresponding concepts in linear algebra and permutation group theory.

Once a base B and a strong generating set S for the group G have been obtained, we can use them to test membership:

Given a group G and a subgroup H of G, the right cosets of H form a partition of G. Let R be a right transversal for $G \mod H$. Hence every element of G can be written in the form hr, where $h \in H$ and $r \in R$. Now, given that we have generated a stabilizer chain for G, every $\sigma \in G$ can be written uniquely as $r_m r_{m-1} \ldots r_1$, such that $r_i \in R_i$. This decomposition can be done algorithmically using a procedure called sifting. If this procedure will fail and we can conclude that σ is not a member of G.

Given σ, we can compute its action on b_1. If $b_1 \cdot \sigma$ lies outside the orbit of b_1 in G, that must mean that σ is not a member of G and we can halt the procedure. Otherwise, we find the coset representative $r_1 \in R_1$ such that $b_1 \cdot r_1 = b_1 \cdot \sigma$. We then replace σ with σr_1^{-1} and check if its action on b_2 lies in that fundamental orbit and compute the corresponding coset representative r_2 if it does. If the procedure does not halt after m iterations, consider the permutation $\sigma r_1^{-1} r_2^{-1} \ldots r_m^{-1}$. By construction, this permutation stabilizes every point in B. As B is a base for G, if this permutation is not the identity, $\sigma \notin G$. If this permutation is the identity, we have obtained $r_1, r_2, \ldots r_m$ such that $r_i \in R_i$ and $\sigma = r_m r_{m-1} \ldots r_1$ which

implies that $\sigma \in G$. We now write down the pseudocode for this algorithm:

Algorithm 1: Sifting Procedure

Input: $\sigma \in S_n$, A base $B = \{b_1, \ldots, b_m\}$ and SGS S for $G \leq S_n$

Output: **true** if $\sigma \in G$. **false** if $\sigma \notin G$

```
for i = 1 to m do
    Compute the fundamental orbit O_i = b_i · G^[i−1] and the transversals R_i as a
     dictionary, with b_i · R_i[x] = x
for i = 1 to m do
    if b_i · σ ∈ O_i then
        σ ← σ R_i[x]^{-1}
    else
        return false
if σ = () then
    return true
else
    return false
```

The stabilizer chain of a group can be also used to compute its order. A consequence of Lagrange's theorem and the orbit stabilizer theorem is that $|G| = \prod_{i=0}^{m-1} \left|G^{[i]} : G^{[i+1]}\right| = \prod_{i=0}^{m} |R_i|$ and hence, the order of G can be readily computed.

An important consequence of the definition of a base is that a permutation in G is uniquely determined by its action on a base. This is because, if there exist two permutations x and y in G which identical action on the base points, the permutation $x^{-1}y$ must stabilize every point in the base and has to be the identity permutation by construction.

Problem 2.1.2. *Given a generator set for a group G, a base $B = \{b_1, \ldots, b_m\}$ and a sequence $\{x_1, \ldots, x_m\} \subseteq \Omega$, does there exist $\sigma \in G$ such that $b_i \cdot \sigma = x_i$, and if so compute σ?*

Given a base and the images of a permutation on the base points of a group, we can reconstruct the permutation modifying the sifting procedure slightly. Corresponding to the base B, we construct the fundamental orbits and the right transversals using the Schreier-Sims algorithm. Initially, we set σ to be the identity. We then find the permutation r_1 in

R_1 such that $b_1 \cdot r_1 = x_1$. If x_1 does not lie in the fundamental orbit of b_1, that means that no such σ exists and we can exit the procedure. We then replace $(x_1, \ldots, x_m)$ with $(x_1 \cdot r_1^{-1}, \ldots, x_1 \cdot r_m^{-1})$. In the i-th iteration, we check if $x_i \cdot r_1^{-1} r_2^{-1} \ldots r_{i-1}^{-1}$ lies in the orbit of G_{b_i}. If it does not, we exit the procedure and if it does we replace σ with σr_i, where r_i is the permutation in the transversal R_i that takes b_i to $x_i \cdot r_1^{-1} r_2^{-1} \ldots r_{i-1}^{-1}$.

Algorithm 2: Element reconstruction algorithm

Input: A base $B = \{b_1, \ldots, b_m\}$ for $G \leq S_n$ and $X = \{x_1, \ldots, x_m\}$

Output: $\sigma \in G$ such that $b_i \cdot \sigma = x_i$ for $i = 1$ **to** m. **false** if no such σ exists

```
σ ← ()
for i = 1 to m do
    Compute the fundamental orbit O_i = b_i · G^[i−1] and the transversals R_i as a
     dictionary, with b_i · R_i[x] = x
for i = 1 to m do
    if x_i ∈ O_i then
        σ ← σR_i[x]
    else
        return false
    for j = 1 to m do
        x_j ← x_j · R_i[x]^{-1}
return σ
```

This procedure is called the *element reconstruction algorithm*. As a permutation is uniquely determined by its action on the base, a permutation in a subgroup of S_n with a base of size k can be uniquely described using k symbols rather than n. This is another way in which a base is analogous to the basis of a vector space. A vector belonging to a k dimensional subspace of an n dimensional space can be described using just k symbols rather than n. In fact the connection is deeper. There exist groups for which all irredundant bases are of the same size. These groups are called IBIS groups (irredundant bases of invariant sizes). In these groups, the bases can be associated with a matroid structure, which is a generalisation of the concept of vector space bases [15].

2.2 The subgroup distance problem

In this section, we define a generalisation of the membership testing problem which has a very interesting connection to error correcting codes.

The natural generalisation of the group membership testing problem would be to find a permutation in G whose action on Ω agrees with the target permutation in as many positions as possible. Unlike membership testing however, there is no efficient algorithm to solve this problem, with the best known algorithms being exponential in the size of the input.

Problem 2.2.1 (Subgroup distance problem)**.** *Given a group $G \leq S_n$, a permutation σ and a natural number k, does there exist a permutation $g \in G$ whose action is identical to that of σ on at least k points out of n?*

For a given permutation σ and a group G, let g be the permutation in G whose action is identical to that of σ on the maximum number of points, say k. If k is equal to n, σ is a member of G. Otherwise, the closer k is to n, the closer σ can be said to being a member of G. The Hamming distance between g and the group is defined to be $d(\sigma, G) = n - k$. We will explore this concept further in the next chapter. The subgroup distance problem can be viewed as the problem of computing this distance. This problem and generalisations of it using different metrics on permutation groups were studied by Buchheim et al [14] who showed that it is **NP**-complete, which means that it cannot be solved in polynomial time unless $\mathbf{P} = \mathbf{NP}$. Arvind and Joglekar [7] also studied this problem and similar weight problems in permutation groups for several metrics. This problem can be solved in $O(2^n)$ time using the following deterministic procedure:

For every $S \subseteq \Omega$ of size at least k, we can use the element reconstruction algorithm to find $g \in G$ such that $x \cdot \sigma = x \cdot \Omega$ for every point x in S. Hence, we can solve this problem by iterating over all the 2^n subsets of Ω and find the largest $S \subseteq \Omega$ and $g \in G$ whose action on Ω is identical to that of σ on S.

2.2.1 Computational complexity

In this section, we investigate the computational complexity of the subgroup distance problem. We now show that it is **NP**-complete by proving the existence of a polynomial time

reduction from the Max-2-SAT problem, which is known to be **NP**-complete.

In computational complexity theory, we say that problem A is *at least as hard as* problem B if there exists an algorithm, called a reduction that runs in polynomial time and converts every instance of problem B to an instance of problem A. This means that an algorithm that solves problem A efficiently would also solve problem B efficiently. The complexity class **P** denotes the set of problems that can be efficiently (in polynomial time) solved. The complexity class **NP** denotes those whose solutions can be efficiently verified. An **NP**-hard problem is a problem in which is at least as hard as every other problem in **NP**. Assuming that $\mathbf{P} \neq \mathbf{NP}$, as is widely believed, this means that polynomial time solutions to **NP**-hard problems are not possible.

The problem of boolean satisfiability (SAT) was independently proved by Cook and Levin to be at least as hard as every other problem in NP. Most other problems are generally proved to be **NP**-hard by coming up with a reduction from another problem which is known to be **NP**-hard, eventually tracing back to SAT. For more details on NP completeness and reductions, we refer to [6].

Boolean variables are variables which can be either TRUE or FALSE. Boolean expressions are built from boolean variables using three logical operations– AND ($\wedge$), also called conjunction, OR ($\vee$), also called a disjunction and negation, or NOT ($\bar{\cdot}$). The AND and OR operations take two variables as input and give out one as output while the negation operation takes in one as input and gives out one as output. Given two boolean variables x and y, $x \wedge y$ is TRUE if and only if x and y are both TRUE, and $x \vee y$ is TRUE if and only if either x or y TRUE. The negation of a variable, $\bar{x}$ is TRUE if and only if x is FALSE. An example of a boolean expression would be $(x_1 \vee x_2) \wedge \bar{x_3}$. A boolean expression is set to be *satisfiable* if there exists an assignment to the variables such that the truth value of the whole expressions is TRUE. For more details, we refer to [22]. It is possible to simplify every boolean expression into the following form, called the conjunctive normal form. The new expression may, however be exponentially longer.

Definition 2.2.1 (Conjunctive normal form)**.** *A formula is said to be in k-CNF form, if it is a conjunction of one or more clauses, where each clause is a disjunction of exactly k variables.*

The expression $\Psi = (x_1 \vee \bar{x_2} \vee x_5) \wedge (x_1 \vee \bar{x_3} \vee x_4) \wedge (\bar{x_1} \vee \bar{x}_4 \vee x_5)$ is an example of an

expression in 3-CNF form.

Problem 2.2.2 (k-SAT)**.** *Given a boolean expression Ψ in k-CNF form, with n variables and m clauses, does there exist an assignment to these variables such that Ψ is satisfied?*

The k-SAT problem is also NP-hard for $k \geq 3$ [22, LO2, Appendix A9]. The 3-SAT problem is a fundamental problem in theoretical computer science, and many important problems in graph theory and combinatorial optimisation have been proven to be NP-hard by obtaining reductions from it. The problem of 2-SAT is solvable in polynomial time. The problem of determining an assignment to an instance of 2-SAT that *maximises* the number of satisfying clauses however, is NP-hard [22, LO5, Appendix A9].

Problem 2.2.3 (MAX-2-SAT)**.** *Given a 2-CNF formula Ψ with n variables and m clauses and a natural number $k \geq 0$. Does there exist an assignment to the variables that satisfies at least k of the clauses?*

We now show the existence of a reduction from MAX-2-SAT to the subgroup distance problem. This reduction is an algorithm that takes in an instance of MAX-2-SAT and outputs and instance of the subgroup distance problem. We note that the permutation group that this reduction outputs are always abelian and have exponent 2. This means that the problem remains **NP**-hard even in this restricted class of inputs.

Now, we prove that the subgroup distance problem is NP-hard by constructing a reduction from MAX-2-SAT.

Theorem 2.2.1. *There exists a reduction f that runs in polynomial time that maps every instance of* MAX-2-SAT *on n variables to an instance (G, σ, k') of the subgroup distance problem with the property that $|G| = 2^n$.*

Proof. Let (Ψ, k) be an instance of MAX-2SAT with m clauses $C_1, C_2, \ldots, C_m$ with n variables $u_1, u_2, \ldots u_n$. We create an instance (G, σ, k') of the subgroup distance problem with G acting on $6m$ points, $|G| = 2^n$ and $k' = 4k$.

Corresponding to each clause C_j, we define the clause gadget $X_j = \{x_{j,1}, \ldots, x_{j,6}\}$ consisting of 6 points, and define the set Ω to be the union of all these clause gadgets. For every variable u_i, we define a corresponding generator π_i which acts on Ω as follows: A variable

can appear in a clause either in the first position or the second position. If u_i appears in the clause C_j in the first position, it swaps $x_{j,1}$ with $x_{j,2}$ and $x_{j,3}$ with $x_{j,4}$. u_i appears in the clause C_j in the second position, it swaps $x_{j,1}$ with $x_{j,2}$ and $x_{j,5}$ with $x_{j,6}$. If the variable does not appear in the clause at all, it fixes all the points in the clause gadget.

Some of the clauses in Ψ may contain some negations. The target permutation σ will depend on which clauses and the positions in these clauses these negations occur in. Consider a clause C_j. If C_j is of the form $x \vee y$, σ acts on C_j as $(x_{j,1}, x_{j,2})(x_{j,3}, x_{j,4})(x_{j,5}, x_{j,6})$. If C_j is of the form $\bar{x} \vee \bar{y}$, σ acts on X_j as $(x_{j,1}, x_{j,2})$. If C_j is of the form $\bar{x} \vee y$, σ acts on X_j as $(x_{j,5}, x_{j,6})$. If C_j is of the form $x \vee \bar{y}$, σ acts on X_j as $(x_{j,3}, x_{j,4})$.

It remains to show that there exists an assignment for Ψ satisfying k clauses if and only if there exists a permutation $g \in G$ whose action on Ω is same as that of σ on at $4k$ points. For an assignment $A : \{u_1, \ldots, u_n\} \to \{\textsc{True}, \textsc{False}\}$, Consider the permutation

$$g := \prod_{i:A(u_i)=\textsc{True}} \pi_i$$

Consider a clause gadget X_j corresponding to a clause C_j containing two variables u_{i_1} and u_{i_2}. There are four possible assignments to the variables in the clause. Each clause specifies an assignment to the variables that will not satisfy it, and the other three will. Observe that by construction, $\pi_{i_1}\pi_{i_2}$ agrees with σ on 4 points of X_j if C_j is satisfied and it agrees on no points if C_j is not satisfied. All the other generators fix all points in X_j. Hence, if k clauses are satisfied, τ agrees with σ on exactly $4k$ points.

Now, assume that there exists $g \in G$ whose action agrees with that of σ on k' points. As g is a member of G and G is abelian with every element being an involution, it can be uniquely written as a product of the defined generators. We define the assignment $A : \{u_1, \ldots u_n\} \to \{\textsc{True}, \textsc{False}\}$ by setting $A(u_i)$ to be $\textsc{True}$ if and only if π_i appears in the composition of g. As before, for any clause, g agrees with σ on 4 points of X_j if C_j is satisfied and it agrees on no points if C_j is not satisfied. Hence, if g agrees with σ on $4k$ points, A satisfies k clauses of Ψ. □

As Max-2-SAT is an **NP**-hard problem, the existence of such a reduction shows that the subgroup distance problem is also **NP**-hard, which means that an efficient algorithm to solve it very unlikely. Our reduction is slightly different from the reduction demonstrated by

Buchheim et al [14]. In that paper, the authors kept the target permutation σ to be fixed, while here, this permutation varies depending on the negations in the input Max-2-SAT instance. The advantage of our reduction is that it yields a permutation group of size 2^n, whereas their reduction yields a permutation group of size 2^{2n}. This fact may come in useful if we are to examine the hardness of this problem from a more fine-grained perspective.

It is important to note that this reduction only proves a *worst case* complexity result. It says nothing about the average case complexity of this problem. In cryptography, we are looking for problems that are hard on average, not just in the worst case. Currently, only lattice-based schemes have such a theoretical backing [1, 2]. Based on similar assumptions for analogous problems in linear coding theory however, we assume that this problem is hard to solve on average too.

2.3 Comparison to problems in linear algebra and coding theory

In this chapter, we have pointed out several parallels between the concepts of a base in permutation group theory and that of a basis in linear algebra. These parallels were combinatorial in nature. in this section, we compare these concepts and problems from an algorithmic viewpoint. We focus on vector spaces over finite fields, These objects are studied in information and coding theory as *linear error correcting codes*. We begin with defining the membership testing in linear algebra. For $m \leq n$, an m dimensional subspace of $\mathbb{F}_q^n$ is called an $[n, m]$ linear code. Consider a basis B for the vector space. The *generator matrix* for a linear code is a matrix with m rows and n columns whose rows are the vectors in B.

Problem 2.3.1 (Membership testing for linear codes)**.** *Consider a a $m \times n$ matrix G over the finite field $\mathbb{F}_q$ and a target row vector t in $\mathbb{F}_q^n$. Does t belong to the vector subspace of $\mathbb{F}_q^n$ spanned by the rows of G? In other words, given the matrix G and the target vector t, does there exists $x \in \mathbb{F}_q^m$ such that $xG = t$?*

We compare the procedure of Gaussian elimination to the sifting procedure in permutation groups. Given a vector space spanned by the vectors $v_1, v_2, \ldots, v_m$, any vector in the vector space spanned by them can be uniquely expressed in the form $\sum_i x_i v_i$. Similarly, any

permutation in the group G can be expressed in the form $r_m r_{m-1} \dots r_1$, such that $r_i \in R_i$, the transversals. The sifting procedure in permutation groups finds r_i's in permutation groups similar to how Gaussian elimination finds the coefficients x_i's in vector spaces. The i-th step of Gaussian elimination finds the coefficient x_i and modifies the target vector and the matrix using some elementary column transformations. It replaces the target vector with a vector in the subspace spanned by $v_{i+1}, \dots, v_m$. Similarly, the i-th step of the sifting procedure finds the coefficient r_i and replaces the target permutation with one in $G^{[i+1]}$.

An alternative method of understanding this problem would be using the concept of a *parity check matrix*. We previously defined a linear code as the subspace of $\mathbb{F}_q^n$ spanned by the rows of a generator matrix. Alternatively, we can understand it as a solution space of a set of $n-m$ linear equations with n variables over $\mathbb{F}_q$. These equations form the parity check matrix of a code. For an $[n, m]$ linear code, the parity check matrix H is defined as a matrix with n rows and $n - m$ columns such that $\mathbf{x}H^T = \mathbf{0}$ for all vectors $\mathbf{x}$ in the linear code. Given the parity check matrix for a linear code, we can test membership of a vector simply by post-multiplying the vector by its transpose. Given the generator matrix of a code, its parity check matrix can be obtained using Gaussian elimination and vice versa.

We now attempt to construct a parallel to this concept for permutation groups. A parity check matrix defines a homomorphism from $\mathbb{F}_q^n$ to $\mathbb{F}_q^{n-k}$, with the linear code being its kernel. Consider a permutation group $G \leq S_n$. For it to have a 'parity check', it must be a kernel of some homomorphism from S_n. This would however mean that is is a normal subgroup of S_n. As the only normal subgroups of the symmetric group are the alternating groups (with two exceptional examples for S_4) [17, section 8.1], this corresponding concept isn't very useful for permutation groups.

Given the generators of a subgroup G of S_n and a target permutation σ, the subgroup distance problem asks if there exists a permutation in G whose action is identical to that of σ on at least k points. This is very similar to the very famous *closest vector problem* in linear coding theory. Given the basis of a k dimensional subspace C of $\mathbb{F}_q^n$ and a target vector t, the closest vector problem asks if there exits a vector in C which is equal to t on at least k positions.

Problem 2.3.2 (Closest Vector Problem)**.** *Consider a $m \times n$ matrix G over $\mathbb{F}_q$ and a target row vector $\mathbf{t}$ in $\mathbb{F}_q^n$, which does not belong to the vector space spanned by the rows of G. Consider the set of n linear equations $\mathbf{x}G = \mathbf{t}$. Does there exist $\mathbf{x} \in \mathbb{F}_q^m$ such that at least k*

of these linear equations are satisfied.

The closest vector problem is also an **NP**-hard problem and the security of the McEliece and Neiderreiter public key cryptosystems depend crucially on the assumption of its hardness [10]. We can also reduce instances of the closest vector problem in $[n, m]$ linear codes to those of the subgroup distance problem using exponent 2 subgroups of S_{2n} of size 2^m, using m generators, which would be another proof for the **NP**-hardness of the subgroup distance problem. Abelian permutation groups of exponent 2 are isomorphic to vector spaces over $\mathbb{F}_2$, which makes the subgroup distance problem a strict generalisation of the closest vector problem. This would lead to an alternate proof for **NP**-hardness of the subgroup distance problem.

In this chapter, we have discussed two important computational problems in permutation group theory, one of which is an easy problem and the other hard. We have also defined several important combinatorial and algorithmic concepts and discussed parallels with some well studied problems and concepts in coding theory. In the next chapter, we will make this comparison concrete by defining a Hamming metric on the symmetric group and discussing the use of permutation groups as error-correcting codes. In Chapter 4, we will develop a framework to use permutation codes in public-key cryptography protocols similar to the McEliece and Neiderreiter cryptosystems.

Chapter 3

Permutation Codes

In this chapter, we explore the use of permutation groups as error-correcting codes. We begin by defining a Hamming metric on permutations, similar to the one defined for vector spaces over finite fields. Consider the set $\Omega = \{1, 2, \ldots, n\}$. For a permutation $g \in S_n$, we define its support $Supp(g)$ and set of fixed points $Fix(g)$ to be those points it moves and those points it fixes respectively. A permutation, which is a bijective function from Ω to Ω can be represented in two ways. The first is by listing down all its images. This is called the list form of a permutation. For example, $[2, 3, 1]$ is a permutation g acting on $\{1, 2, 3\}$ such that $1 \cdot g = 2, 2 \cdot g = 3, 3 \cdot g = 1$. The other way is as a product of disjoint cycles. The list form of a permutation is more useful in defining a Hamming metric.

Definition 3.0.1. *For elements $g, h \in S_n$, the Hamming distance between them, $d(g, h)$ is defined to be the size of the set $\{x \in \Omega | x \cdot g \neq x \cdot h\}$.*

In other words, the Hamming distance between two permutations is the the number of positions on which they differ in when written down in list form. The Hamming distance between two permutations g and h can be proved to be equal to $|Supp(gh^{-1})| = n - |Fix(gh^{-1})|$. A *permutation code* is a collection of permutations in S_n endowed with a Hamming metric. For a subset C of S_n, the minimum distance of the subset is defined to be $\min_{g,h \in C} d(g, h)$. In this thesis, we assume that the collection of permutations form a subgroup G of S_n. In that case, the minimum distance can be shown to be:

$$m(G) = \min_{g \in G, g \neq 1} |Supp(g)| = n - \max_{g \in G, g \neq 1} |Fix(g)| \tag{3.1}$$

This is also known as the *minimal degree* of the permutation group. The quantity $r = \left\lfloor \frac{m(G)-1}{2} \right\rfloor$ is called the *correction capacity* of the code.

We can now define the distance between a permutation σ and a group G to be the distance between σ and the permutation in G closest to it.

$$d(G, \sigma) = \min_{g \in G} d(g, \sigma)$$

The subgroup distance problem discussed in the previous chapter can hence be rephrased as follows: Given the generators of a permutation group $G \leq S_n$, a permutation $\sigma \in S_n$ and an integer $k > 0$, is $d(\sigma, G) \leq k$? We can also generalise this problem to consider inputs σ to be lists of length n with symbols from Ω which do not necessarily form a permutation.

A nearest neighbour of σ in G is defined to be a permutation g in G such that $d(g, \sigma) = d(G, \sigma)$. This permutation exists because $d(g, \sigma) = \min_{g \in G} d(g, \sigma)$. If the distance between σ and G is greater than the correction capacity of the group, this nearest neighbour need not be unique. If $d(\sigma, G) \leq r$ however, σ's nearest neighbour in G is unique.

Hence, G can be used as an error-correcting code in the following manner. The information we want to transmit is encoded as a permutation in G and transmitted as a list through a noisy channel. We assume that when the message is received, r or fewer errors have been induced. Such a channel is called a *Hamming channel*. The received message is a list w of length n with symbols from Ω (which need not necessarily form a permutation). Because the correction capacity of G is r, the nearest neighbour of w in G is uniquely defined to be g. The decoder can now solve the subgroup distance problem with the word w and the group G as input to recover the original message g. However, while we have proved that decoding is possible in theory, because the subgroup distance problem is NP-complete, generic algorithms cannot be used to decode permutation codes efficiently. Rather, we need specific algorithms tailor-made for different families of permutation codes.

The use of sets of permutations in coding theory (also referred to as permutation arrays) has been studied since the 1970's. Blake, Deza and Cohen [25, 13] were the first to discuss using permutations this way. They have had some applications, for example in powerline communications. However, they have never reached the level of attention that linear codes have gotten. The idea of using groups of permutations in coding theory was studied by Bailey and Cameron in Bailey's PhD thesis [20]. In his thesis, he describes a variety of permutation

codes and presents a decoding algorithm which works for arbitrary permutation groups using a combinatorial structure called *uncovering-by-bases* which is similar to the technique of permutation decoding developed by MacWilliams and Sloane [19, 8]. They were the first to exploit the algebraic structure of groups to come up with decoding algorithms for several families of groups.

Most of this chapter is a review of Bailey's work on permutation codes. We have however, performed some extra computational analysis of the alternative decoding algorithm he described in Section 3.3.1

3.1 A decoding algorithm using uncovering-by-bases

In this section, we review Bailey's work on decoding algorithms for permutation codes. Consider a permutation group $G \leq S_n$ with correction capacity r. Bailey's decoding algorithm uses the following combinatorial structure. An uncovering-by-bases (UBB) is a collection of bases for G. To recall, a base is a set of points in Ω with the property that the only element of G which stabilises all the points is the identity. An r-subset of Ω is a subset of Ω with size r.

Definition 3.1.1. *An Uncovering-by-bases (UBB) to fix r errors in a permutation group acting on Ω is a set of bases for the group such that for each r-subset of Ω, there is at least one base in the UBB that is disjoint from the r subset.*

It is known that every permutation group has an uncovering-by-bases [19, Proposition 7]. However, they have been constructed only for a few permutation groups. There isn't any known procedure to construct a small UBB for an arbitrary permutation group. Once an uncovering-by-bases has been obtained we can use it in the following decoding algorithm-

Consider a permutation $g \in G$ in list form and the list w obtained after r or fewer errors are introduced to g. Note that w can have repeated entries and hence need not be a permutation. The errors have been introduced in a set of positions R with size less than or equal to r (not known to the decoder a priory). Let $\mathcal{U}$ be a UBB for G. It is a consequence of the definition of a UBB that there exists a base B in $\mathcal{U}$ which is completely disjoint from R. Hence, in the positions indexed by B, the lists w and g are identical. As B is a base, we can

reconstruct the entire permutation B from just its action on the points in B using the element reconstruction algorithm. While the decoder doesn't know which base in $\mathcal{U}$ is the appropriate base B whose positions do not have an error induced, they can just repeat this procedure for every base in $\mathcal{U}$. We now write down the pseudocode for this algorithm. Consider a list w, a base B and a generator set S. The function ElementReconstruction(w, S, B) constructs an element g of the group generated by S such that $i \cdot g = w[i]$ for all $i \in S$ (Algorithm 2).

Algorithm 3: UBB decoding algorithm

Input: A set of generators S for $G \leq S_n$ with correction capacity r, A UBB $\mathcal{U}$ for G, a list w of symbols in Ω

Output: $g \in G$ such that $d(\sigma, w) \leq r$. **false** if no such g exists

```
for B ∈ U do
    if w has no repeated symbols in positions indexed by B then
        g' = ElementReconstruction(w, S, B)
        if g' ≠ false and d(g, g') ≤ r then
            return g'
return false
```

This algorithm is similar to the method of *permutation decoding* for linear codes proposed by MacWilliams and Sloane [30]. This involves using a subset of the automorphism group of the code which would move any errors out of the information positions.

3.2 Some constructions of uncoverings-by-bases

We now consider some examples of uncovering-by-bases in Bailey's thesis.

3.2.1 Symmetric group acting on 2-sets

Let Ω be the set of all 2-subsets of $\{1, 2, \ldots, m\}$. Consider the symmetric group S_n acting on Ω with the setwise action: $g \cdot \{i, j\} = \{g \cdot i, g \cdot j\}$. Hence, G is a subgroup of S_n in this action. The minimal degree of G is $2(m-2)$ and its correction capacity is $m-3$ [19, Proposition 21].

The set of all 2-subsets of $\{1, 2, \ldots, m\}$ can also be viewed as the edge set of the complete graph K_m. This means that we can use graph theoretical results such as the following to construct bases for G, which we will use to construct a UBB.

A spanning subgraph of K_m which has at most one isolated vertex and no isolated edges forms a base for G. A Hamiltonian circuit of K_m is a circuit in K_m containing each vertex exactly once. From a Hamiltonian cycle of K_m, we can obtain such a spanning graph (called a V-graph) by deleting every third edge as in Fig. 3.1. A Hamiltonian decomposition of a graph is a partition of the edge set of the graph into disjoint Hamiltonian circuits and at most one perfect matching. In the 1890's, Walecki [5] showed that K_m can be decomposed into $(m-1)/2$ Hamiltonian cycles when m is odd and that if m is even, it can be decomposed into $(m-2)/2$ Hamiltonian cycles and one perfect matching. Using Walecki's decomposition, we can construct a UBB for G. The UBB is the set of all V-graphs which can be obtained from the Hamiltonian decomposition of K_m and it can be proved that any r-subset of the edges of K_m avoids at least one of these V-graphs. This UBB is of size $O(m)$, and each base is of size $O(m)$. Hence, we can use this UBB in the decoding algorithm would have time complexity of $O(m^4) = O(n^2)$.

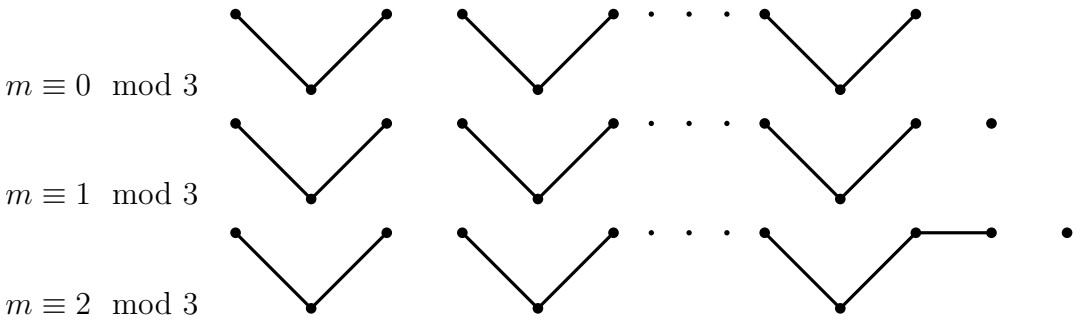

Figure 3.1: V-graphs for different cases of $m \mod 3$

3.2.2 Wreath product of regular groups

Let H be a group acting on a set Δ. It is called a regular group if, for every $x, y \in \Delta$, there exists exactly one $h \in H$ such that $x \cdot h = y$. The permutation code G is the wreath product of H, a regular permutation group acting on m points with the symmetric group S_n acting imprimitively on mn points.

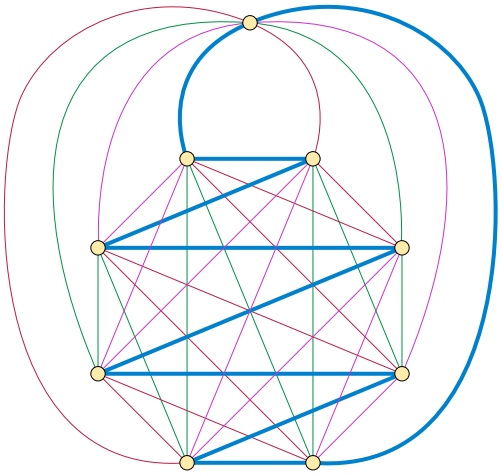

Figure 3.2: Walecki Decomposition of K_9

The group G consists of all tuples of the form $(h_1, h_2, \ldots, h_n, \sigma)$, where $h_i \in H$ and $\sigma \in S_n$. We use the symbol '1' to denote the identity element of H and the symbol 'e' to denote the identity element of S_n. We can consider the set $\Omega = [m] \times [n]$ consisting of n copies of the set $[m]$ as n columns each with m rows. The action of $g = (h_1, h_2, \ldots, h_n, \sigma)$ on (i, j) is $(i \cdot h_{i\cdot\sigma}, j \cdot \sigma)$. The group S_n acts on the columns and the group H^n acts on the rows. As a regular group acting on m points has size m, $|G| = m^n n!$.

Theorem 3.2.1. *The minimal degree of G is m and its error correction capacity is $\left\lfloor \frac{m-1}{2} \right\rfloor$.*

Proof. Consider the group element $(h, 1, \ldots, 1, e)$ for some $h \in H$. This permutation moves all the points in the first column and fixes all the other points. Hence the minimal degree of H is less than or equal to m. We now show that any non identity permutation in G moves at least m points. Consider a permutation $(h_1, h_2, \ldots, h_n, \sigma)$. If σ is not the identity, it must move all the points in one of the columns to another column and hence moves at least m points. If σ is the identity permutation, at least one of the h_i's must be a non identity element of H which will move all m points in that column. □

Theorem 3.2.2. *Any subset of Ω forms an irredundant base for G if and only if it has exactly one point from each column of Ω.*

Proof. Consider any set S which does not contain any points from the first column. The permutation $(h, 1, \ldots, 1, e)$ fixes every point in S. Hence, S cannot be a base for G.

To prove the converse, consider the following set $\{(x_1, 1), (x_2, 2), \ldots, (x_n, n)\}$ with each x_i belonging to the i-th column. Suppose here exists $g = (h_1, h_2, \ldots, h_n, \sigma)$ that fixes each of these points. As $(x_i, i) \cdot g$ must lie in the i-th column for each i, this implies that σ must be the identity permutation. Hence, the action of g on x_i is $x_i \cdot h_i$. As the action of H on $[m]$, is regular, this implies that each h_i is the identity.

We have proved that B is a base for G if and only if B contains at least one point from each column of Ω. If B has more than one point from some of the columns of Ω, we can obtain a subset of B which contains exactly one point from each column of Ω and is a base for G. This means that B must be redundant. Hence, we have proved that B is a irredundant base for G if and only if it has exactly one point from each column of Ω. □

Using this theorem, we can now construct a UBB for G. Each row for of Ω forms a base and a collection of $r + 1$ rows would form a UBB for G, where $r = \left\lfloor \frac{m-1}{2} \right\rfloor$. This is because any subset of size r will intersect with at most r of those bases described in the UBB and hence, disjoint from at least one of those bases.

Having obtained a UBB, we can use it in the decoding algorithm to obtain one which runs in $O(m^2n^2)$.

3.3 An alternative decoding algorithm for wreath product groups

In this section, we present a simple decoding algorithm introduced by Bailey, which uses majority logic principles for the wreath product groups and compare its performance to the original UBB algorithm. For simplicity, we assume the group H to be the cyclic group C_m, although this algorithm can be extended to any regular groups.

Consider a permutation g in G, with less than r errors induced to form a word w. The permutation g acts on Ω, which as we noted before, can be viewed of as n copies of the set $[m]$ as n columns each with m rows. Firstly, we rewrite each member of Ω and the letters in the received word in the form (i, j), with $i \in [m]$ and $j \in [n]$. The columns of Ω form a complete block structure for G. As less than $r = \left\lfloor \frac{m-1}{2} \right\rfloor$ errors were introduced,

the majority of positions in each block will contain the correct symbol. Let g be of the form $(h_1, \ldots, h_n, \sigma)$ where each h_i is a cyclic shift. In the j-th column, we consider the i-th position which contains the tuple (p, q), and calculate the number $p - i$, which corresponds to the cyclic shift in that block. We set the value of h_j to be the most frequently occurring cyclic shift and we set $j \cdot \sigma$ to be the most frequently occurring value of q in that block. When we repeat this procedure for each block, we can obtain the cyclic shifts corresponding to each h_j and obtain σ by computing its action on each $j \in [n]$. We now convert g from the form $(h_1, \ldots, h_n, \sigma)$ to that of a permutation acting on Ω.

We now demonstrate this using GAP. We consider the group $G = C_5 \wr S_{10}$, and pick a random element of G and convert it in the required form.

```
gap> H:=Group((1,2,3,4,5));
Group([ (1,2,3,4,5) ])
gap> S:=SymmetricGroup(10);
Sym( [ 1 .. 10 ] )
gap> G:=WreathProductImprimitiveAction(H,S);
<permutation group of size 35437500000000 with 12 generators>
gap> MaximalBlocks(G,[1..50]);
[ [ 1, 2, 3, 4, 5 ], [ 6, 7, 8, 9, 10 ], [ 11, 12, 13, 14, 15 ],
[ 16, 17, 18, 19, 20 ], [ 21, 22, 23, 24, 25 ], [ 26, 27, 28, 29, 30 ],
[ 31, 32, 33, 34, 35 ], [ 36, 37, 38, 39, 40 ], [ 41, 42, 43, 44, 45 ],
[ 46, 47, 48, 49, 50 ] ]
gap> g:=Random(G);
(1,40,49,14)(2,36,50,15)(3,37,46,11)(4,38,47,12)(5,39,48,13)(6,10,9,8,7)
(16,21,29,41,33)(17,22,30,42,34)(18,23,26,43,35)(19,24,27,44,31)(20,25,28,45,32)
gap> ListPerm(g,50);
[ 40, 36, 37, 38, 39, 10, 6, 7, 8, 9, 3, 4, 5, 1, 2, 21, 22, 23, 24, 25, 29, 30,
26, 27, 28, 43, 44, 45, 41, 42, 19, 20, 16, 17, 18, 50, 46, 47, 48, 49, 33, 34,
35, 31, 32, 11, 12, 13, 14, 15 ]
gap> L:=[];
[  ]
gap> for i in ListPerm(g,50) do
> q:=QuoInt(i,5);
> r:=RemInt(i,5);
```

```
> if r<>0 then
> Add(L,[r,q]);
> else
> Add(L,[r,q-1]);
> fi;
> od;
gap> L;
[ [ 0, 7 ], [ 1, 7 ], [ 2, 7 ], [ 3, 7 ], [ 4, 7 ], [ 0, 1 ], [ 1, 1 ], [ 2, 1 ],
[ 3, 1 ], [ 4, 1 ], [ 3, 0 ], [ 4, 0 ], [ 0, 0 ], [ 1, 0 ], [ 2, 0 ], [ 1, 4 ],
[ 2, 4 ], [ 3, 4 ], [ 4, 4 ], [ 0, 4 ], [ 4, 5 ], [ 0, 5 ], [ 1, 5 ], [ 2, 5 ],
[ 3, 5 ], [ 3, 8 ], [ 4, 8 ], [ 0, 8 ], [ 1, 8 ], [ 2, 8 ], [ 4, 3 ], [ 0, 3 ],
[ 1, 3 ], [ 2, 3 ], [ 3, 3 ], [ 0, 9 ], [ 1, 9 ], [ 2, 9 ], [ 3, 9 ], [ 4, 9 ],
[ 3, 6 ], [ 4, 6 ], [ 0, 6 ], [ 1, 6 ], [ 2, 6 ], [ 1, 2 ], [ 2, 2 ], [ 3, 2 ],
[ 4, 2 ], [ 0, 2 ] ]
```

As we can see, in each block, the value of the second coordinate is the same. Now, let's see what the list L would look like if we introduce two errors to the permutation in list form. For example, let us switch the second and last symbols to obtain:

```
[ 40, 15, 37, 38, 39, 10, 6, 7, 8, 9, 3, 4, 5, 1, 2, 21, 22, 23, 24, 25, 29, 30,
26, 27, 28, 43, 44, 45, 41, 42, 19, 20, 16, 17, 18, 50, 46, 47, 48, 49, 33, 34,
35, 31, 32, 11, 12, 13, 14, 36 ]
```

After performing the conversion, this list becomes:

```
[ [ 0, 7 ], [ 0, 2 ], [ 2, 7 ], [ 3, 7 ], [ 4, 7 ], [ 0, 1 ], [ 1, 1 ], [ 2, 1 ],
[ 3, 1 ], [ 4, 1 ], [ 3, 0 ], [ 4, 0 ], [ 0, 0 ], [ 1, 0 ], [ 2, 0 ], [ 1, 4 ],
[ 2, 4 ], [ 3, 4 ], [ 4, 4 ], [ 0, 4 ], [ 4, 5 ], [ 0, 5 ], [ 1, 5 ], [ 2, 5 ],
[ 3, 5 ], [ 3, 8 ], [ 4, 8 ], [ 0, 8 ], [ 1, 8 ], [ 2, 8 ], [ 4, 3 ], [ 0, 3 ],
[ 1, 3 ], [ 2, 3 ], [ 3, 3 ], [ 0, 9 ], [ 1, 9 ], [ 2, 9 ], [ 3, 9 ], [ 4, 9 ],
[ 3, 6 ], [ 4, 6 ], [ 0, 6 ], [ 1, 6 ], [ 2, 6 ], [ 1, 2 ], [ 2, 2 ], [ 3, 2 ],
[ 4, 2 ], [ 1, 7 ] ]
```

The number 7 appears 4 times in the first block and the number 2 appears 4 times in the

final block. The algorithm then replaces both these symbols with the majority. Now, we replace the symbols in the first coordinate with the cyclic shift:

```
gap> for i in [1..50] do
> r:=RemInt(i,5);
> L[i][1]:=L[i][1]-r mod 5;
> od;
gap> L;
[ [ 0, 7 ], [ 4, 2 ], [ 0, 7 ], [ 0, 7 ], [ 0, 7 ], [ 0, 1 ], [ 0, 1 ], [ 0, 1 ],
[ 0, 1 ], [ 0, 1 ], [ 3, 0 ], [ 3, 0 ], [ 3, 0 ], [ 3, 0 ], [ 3, 0 ], [ 1, 4 ],
[ 1, 4 ], [ 1, 4 ], [ 1, 4 ], [ 1, 4 ], [ 4, 5 ], [ 4, 5 ], [ 4, 5 ], [ 4, 5 ],
[ 4, 5 ], [ 3, 8 ], [ 3, 8 ], [ 3, 8 ], [ 3, 8 ], [ 3, 8 ], [ 4, 3 ], [ 4, 3 ],
[ 4, 3 ], [ 4, 3 ], [ 4, 3 ], [ 0, 9 ], [ 0, 9 ], [ 0, 9 ], [ 0, 9 ], [ 0, 9 ],
[ 3, 6 ], [ 3, 6 ], [ 3, 6 ], [ 3, 6 ], [ 3, 6 ], [ 1, 2 ], [ 1, 2 ], [ 1, 2 ],
[ 1, 2 ], [ 1, 7 ] ]
```

In the first block, the number 0 appears 4 times and the number 4 appears once. Hence, we replace the number 4 by 0. In all the other blocks, there is no rewriting necessary. We have now recovered the original permutation.

We now estimate the complexity of this algorithm. The first part of the algorithm involves splitting the word into blocks and rewriting each symbol. This could be done in constant time for each position. Computing the value of the block label and the cyclic shift for each position involves some integer arithmetic which would also take a constant amount of time for each position. Hence this would take $O(mn)$ time.

The second part of the algorithm involves finding the most frequently occurring value of the block value and cyclic shift in each block. Let's consider the part where we find the most frequently occurring value of the cyclic shift. We maintain an auxiliary list of size m, with each position corresponding to the frequency of that cyclic shift. We iterate through the block, compute the cyclic shift for each location of the block and update its frequency in the auxiliary list. This procedure takes $O(m)$ time. We then go through the auxiliary list to find the most frequently occurring cyclic shift. We do a similar procedure for finding the action of σ on the block using an auxiliary list of size n and would hence take $O(n)$ time. This procedure is repeated for each block.

The final part of the algorithm, which involves converting the reconstructed word back to a permutation form can be done with $O(mn)$ arithmetic operations. Hence, the algorithm takes $O(mn)$ time if $m > n$ and $O(n^2)$ time if $n > m$. This is faster than the UBB algorithm for the same group, which would take $O(m^2n^2)$ time.

3.3.1 Decoding more than r errors

In fact, the algorithm described in the previous section can correct more than r errors, as long as the majority of elements in each block are correct. Hence there are some error patterns with even up to nr errors which can be decoded using this algorithm. Bailey obtained the following expression for the number of error patterns of a certain length which can be corrected by this algorithm (an error pattern is a certain set of positions on which the errors are induced).

Let k be a positive integer. Let $P_{n,r}(k)$ be the set of all partitions of k into at most n parts with each part having size at most r . We write down such a partition in the form $\pi = \{(i, f_i) | \sum_i i f_i = k\}$, where f_i is the number of times the number i appears in the partition. For a partition π, we define the quantity c_i to be $c_i = \sum_{j=1}^{i-1} f_i$, which is the number of parts in π of size strictly less than i.

Theorem 3.3.1. *For an integer $k \leq nr$, the following number of patterns of k errors can be corrected:*

$$E_{n,r}(k) = \sum_{\pi \in P_{n,r}(k)} \prod_{i=1}^{r} \binom{n - c_i}{f_i} \binom{m}{i}^{f_i}$$

Proof. For a proof, we refer to Bailey's thesis [20, Section 6.7]. It is a farily simple counting argument using these definitions described above. □

We ran some simulations to find the proportion of error patterns which can be corrected for the case $m = 5$, in which case the minimal degree is 5 and has correction capacity 2. In practice, however, a much larger number of errors can be corrected, with a high probability of success.

Using a computer program, we calculate the value of $E_{n,r}(k)$ for different values of k, and calculate the value of $\frac{E_{n,r}(k)}{mn}$ to obtain the probability that the alternate decoding algorithm

succeeds. We plot this for $n = 100$:

We can deduce from the data the algorithm can decode 95 percent of error patterns of length up to 19, 90 percent of error patterns of length up to 24, 80 percent of error patterns of length up to 31, and 50 percent of error patterns of length up to 46.

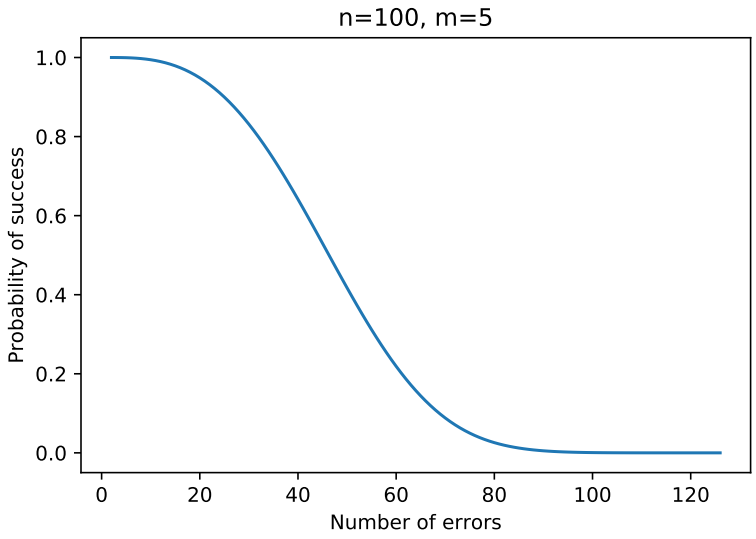

Figure 3.3: Plot of fraction of errors that can be corrected for $n = 100$, $m = 5$

Interestingly, with the probability of success fixed, the number of errors which can be corrected increases with n. This is despite the minimal degree (and hence the correction capacity) remaining the same. We plot the number of errors which can be plotted with 95 percent probability of success, with respect to n, both in terms of absolute number of errors which can be corrected and in terms of the *error rate*, which is the ratio of the error induced to the degree of the permutation group. Unlike the correction capacity which increases with n, the error rate seems to decrease slightly as n increases.

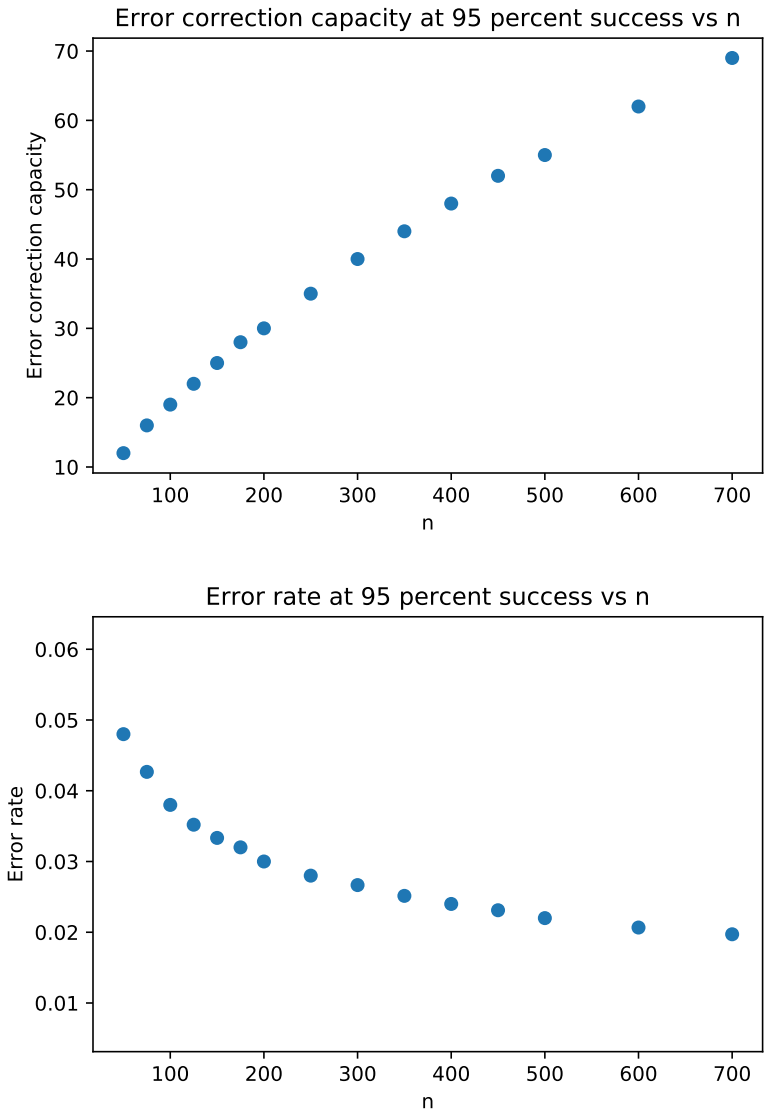

Figure 3.4: Plot of error correction capacity at 95% probability of success

Chapter 4

Permutation codes and public-key cryptography

The McEliece cryptosystem is a very popular cryptosystem using linear codes, first proposed by Robert McEliece [31] in 1978. Due to its large key sizes, it never gained a lot of popularity at the time. Recently however, with the rise of quantum computing, it has gained a lot of attention along with its counterpart proposed by Neiderreiter [33] as a *post-quantum* alternative to currently used public key cryptosystems. Fundamentally, the McEliece cryptosystem uses two linear codes, one kept private and another made public with the two codes being equivalent in the following way:

Definition 4.0.1. *Consider two $[n, k]$ linear codes C_1 and C_2 generated by the generator matrices G_1 and G_2. These codes are considered equivalent if, there exists a non singular $k \times k$ matrix S and an $n \times n$ permutation matrix P such that $G_2 = SG_1P$*

Consider a linear code C generated by a matrix G. Pre-multiplying G by a non-singular matrix will not change the code it generates, as this is basically a change of basis for the code C. Post-multiplying G by a permutation matrix however, creates a new code, one whose codewords are obtained by permuting the positions of the codewords in C by the permutation corresponding to the permutation matrix. In other words, there is a *distance preserving map*, or isometry between two equivalent linear codes. The principle behind the McEliece cryptosystem is that we have a map between an easy instance and a hard instance of the closest vector problem in linear codes using this distance preserving map. The private

key matrix, is chosen from a family of linear codes with a known fast decoding algorithm, but the public key matrix generates a linear code is assumed to not have a fast decoding algorithm. The map between these easy and hard instance is known only to the owner of the private key. This provides the one-way property which is essential to all public-key cryptosystems.

In the previous chapter, we studied using permutation groups as error-correcting codes. In this chapter, we investigate the following questions:

- Can we develop a public key cryptosystem similar to the McEliece cryptosystem using permutation codes instead of linear codes?
- Is there any potential advantage of using permutation codes instead of linear codes from the standpoint of cryptography?

The first step in this direction would be to investigate isometries of the symmetric group with the Hamming metric. Let us say we know an isometry ϕ of the Hamming space. We can have the private key as a permutation code H (which is a subgroup of S_n) with an efficient decoding algorithm and the public key as the permutation code $\widehat{H} = \phi(H)$ for which the decoding algorithm will not work. The isometries in the symmetric group were studied and classified by Farahat [27] who proved that they are of one of the following types:

- $x \mapsto gxh$ for g and h being fixed members of S_n
- $x \mapsto gx^{-1}h$ for g and h being fixed members of S_n

However the isometries that interest us must also be homomorphisms. This is mainly due to practical reasons. We want both the codes H and $\widehat{H}$ to be subgroups of S_n[1]. This is because a permutation group can be represented efficiently using a generator set. A permutation array without a group structure however, cannot be represented efficiently this way. Hence, as we want both H and $\widehat{H}$ to be groups, we are interested in isometries of S_n that map subgroups to subgroups, which have to be homomorphisms. Hence, it is clear that the only possible isometries of S_n which we can use are the conjugation maps, which are maps of the form $x \mapsto g^{-1}xg$ for a fixed element g.

[1]Permutation codes can also just be subsets of S_n without a group structure, although we didn't study these types of codes in detail

Now, we combine these principles to create a public-key cryptosystem. Our ingredients are a permutation code with a fast decoding algorithm and a conjugation map. The following cryptosystem is naturally uses these ingredients in a way that mirrors the McEliece cryptosystem very closely. This chapter consists entirely of our original contributions.

4.1 A McEliece cryptosystem using permutation codes

In this section, we present a public-key cryptosystem using permutation codes that is similar to the McEliece cryptosystem.

Private key: We choose a subgroup H of S_n with a fast decoding algorithm and a permutation g from S_n randomly. The private key is a set of randomly chosen generators $\{h_1, \ldots, h_s\}$, for H and g.

Public key: The public key is the sequence of elements $\{\hat{h}_i\} = \{g^{-1}h_i g\}$, which generate the conjugate group $\widehat{H} = g^{-1}Hg$ and a base B for $\widehat{H}$. Note that $\widehat{H}$ does not have a fast decoding algorithm.

Encryption: The plaintext is an $\hat{h}$ of $\widehat{H}$. We then introduce r errors to $\hat{h}$ randomly to create the ciphertext c.

Decryption: Once we receive c, we compute gcg^{-1}. We then use the fast decoding algorithm to obtain h, which is its nearest neighbour in H. We then compute $\hat{h} = g^{-1}hg$.

The conjugation map is an isometry of the Hamming space. That is why this cryptosystem works. The distance between h and H is equal to the distance between $\hat{h}$ and $\widehat{H}$. In the McEliece cryptosystem, post-multiplication by a permutation matrix is the Hamming isometry used. The secret scrambler however, which just scrambles the basis of a linear code to create a new basis. It doesn't change the linear code itself. Here, the analogue to that would be the fact that we choose a random set of generators for the conjugate group as the public key.

There is a possibility of using other isometries other than the conjuation map. However, the errors have to be induced more carefully. For example, suppose we have a group $G \leq S_n$ and a map from G to G which is an isometry locally. Our permutation code is a subgroup $H \leq G \leq S_n$. Hence, if we induce the errors in H so that the ciphertext c lies in the group G, the cryptosystem using this isometry will work too. As of now however, we do not have an example of a situation where doing this would be beneficial.

4.2 The security of our cryptosystem

4.2.1 Security assumptions

There are two security assumptions:

AS1 Solving the general decoding problem in $\widehat{H}$ is hard. This means that $\widehat{H}$ has no good decoding algorithm.

AS2 Inability to construct a subgroup H' of S_n with generators $\{h'_1, \ldots, h'_s\}$ and $g \in S_n$, such that $g^{-1}H'g = \widehat{H}$ and there is an efficient decoding algorithm in H' that can correct up to r errors.

We now discuss possible classical attacks on our cryptosystem and it's security in the face of these attacks. Like the McEliece cryptosystem, attacks on this cryptosystem can broadly be classified into two types– unstructured and structural attacks. Unstructured attacks attempt to solve the nearest neighbour problem in $\widehat{H}$ directly without attempting to obtain the private key. A structural attack would attempt to obtain the private key from the available public information.

The group H has an efficient decoding algorithm. For the cryptosystem to be secure, the group $\widehat{H}$ must not have an efficient decoding algorithm. Specifically, if the algorithm used to decode in H can be modified to work for the conjugate group $\widehat{H}$, the cryptosystem will not be secure. This would be another type of attack which would be specific to the cryptosystem using a particular permutation code. Hence, the security of the cryptosystem depends on the structure of the permutation code H used and its decoding algorithm, and

not just on parameters like size, correction capacity, etc. This is especially true in the context of structural attacks. Just like the case with the McEliece cryptosystem, we cannot provide any theoretical basis to the claim that the cryptosystem using a certain permutation code is secure. Rather, we construct a cryptosystem and then attempt to come up with attacks to demonstrate its security. Due to the similarity of our cryptosystems with linear code based cryptosystems, which are very well studied and believed to be secure, one good place to start would be to try and extend the well known attacks on the McEliece cryptosystem to our cryptosystem. One of the attacks we consider in this chapter is the well studied and powerful information set decoding (ISD) attacks on the McEliece cryptosystem. For the case of permutation codes, we have designed an algorithm which is very similar in spirit to the ISD attack and we study its effectiveness. In the next chapter, we consider quantum attacks on our cryptosystem. The only known way to attack the McEliece cryptosystem by taking advantage of quantum computing is by attempting to solve the *scrambler permutation problem* using the technique of quantum Fourier sampling. In the next chapter (Section 5.3), we present a similar attack on our cryptosystem after reviewing some literature on quantum computing. This attack attempts to uncover the secret conjugator g from the public information.

We now present some generic classical attacks on our cryptosystem. These attacks work regardless of the choice of the permutation code. Hence, their effectiveness only depend on public parameters like the correction capacity and size of the code. Later, we will demonstrate the existence of a code with parameters that make the cryptosystem secure against these attacks (Section 4.3).

4.2.2 Brute Force Attacks

One obvious method to solve the nearest neighbour problem in $\widehat{H}$ is to enumerate all elements of $\widehat{H}$ and calculate their hamming distance from the ciphertext. This means that $\widehat{H}$ must be large. Its exact size requirement would depend on the level of security we are looking for.

Another way is to enumerate all possible permutations within a distance of r from the codeword c and check if they belong to $\widehat{H}$ using the sifting procedure. The complexity of this attack would be $\binom{n}{r}(n-1)^r$, where n and r are the degree and correction capacity of the group respectively.

4.2.3 Information Set Decoding

One of the more successful attacks on the McEliece cryptosystem uses the technique of information set decoding. The original idea was proposed by Prange [36] in 1962 and over the years many improvements on this basic attack have been proposed [35, 11]. For an $[n, k]$ linear code, it involves picking k coordinates at random and trying to reconstruct the codeword from those positions. It would be successful if the corresponding columns in the generator matrix form an invertible sub-matrix and no errors were induced in those coordinates. We can then recover the codeword from the ciphertext by solving a system of linear equations. We now recreate a version of ISD for permutation codes:

Algorithm 4: Information Set Decoding

Input: generators of $\widehat{H}$, A ciphertext c

```
Procedure ISD():
    Choose a base B and SGS for Ĥ at random
    if There are no repeated symbols in positions indexed by B then
        Use element reconstruction algorithm to try and find an element h ∈ Ĥ
         agreeing with c on these positions.
        if h exists and d_H(h, c) ≤ r then
            return h
    return False
Procedure repeated till it succeeds
```

This algorithm is similar to the uncovering-by-bases decoding algorithm proposed by Bailey, except that here, we do not know what the UBB is and hence, the running time isn't bounded by the size of the UBB. The correctness of this algorithm follows from the proof in Bailey's thesis [19, Proposition 7] in which he shows that given any r-subset of $\{1, \ldots, n\}$, there exists a base for H disjoint from it. Each iteration is a success if none of the positions of c indexed by the chosen base have an error, which occurs with a finite non-zero probability because such a base exists. The performance of the ISD algorithm is identical for the groups H and $\widehat{H}$. This is because B is a base for H if and only if $g^{-1}B$ is a base for $\widehat{H}$, and the algorithm picks a base at random (g is the secret conjugator)

Complexity: The algorithm uses a random procedure to choose a base B for the group at random. At the same time, a set R of size r is chosen uniformly at random and errors are induced in those positions. Before making concrete complexity estimates, we would like to make some observations. First, it is clear that if there are more error positions, the probability that the information set doesn't have any errors is lower and hence the complexity of the attack increases with r. Also, if the size of the base chosen is small, the probability that the positions indexed by this base doesn't have any error positions is high. Hence the complexity of the attack increases with k, the expectation of the length of the base.

Let E_B denote the event that the algorithm chooses subset B as a base for the group. Conditioned on the event E_B, an iteration of the algorithm succeeds if the sets R and B are disjoint. Because the set R is chosen uniformly at random from $\{1, \ldots, n\}$,

$$Pr[\text{Success}|E_B] \geq \frac{\binom{n-|B|}{r}}{\binom{n}{r}}$$

As this probability depends only on the size of B and not on the set B itself, we can obtain an expression for the probability of an iteration of the algorithm succeeding:

$$Pr[\text{Success}] \geq \sum_k Pr[\text{Success}||B| = k]Pr[|B| = k] = \sum_k \frac{\binom{n-k}{r}}{\binom{n}{r}} Pr[|B| = k]$$

Let $k_{\max}$ be the size of the irredundant base of largest size of H. As the $Pr[\text{Success}|E_B]$ decreases as the size of the base increases, we can obtain the following lower bound on the probability of the algorithm succeeding:

$$Pr[\text{Success}] \geq \frac{\binom{n-k_{\max}}{r}}{\binom{n}{r}}$$

This is a Monte Carlo algorithm, which decodes correctly with some probability $Pr[\text{Success}]$, and otherwise returns false. If we repeat the algorithm till it succeeds, we obtain a Las Vegas algorithm that always decodes correctly but has a running time which is a random variable with expectation $1/Pr[\text{Success}]$. It is also important to note that as each iteration is independent, this attack can be very effectively parallelized. Using the following well known formula for binomials,

$$\binom{n}{r} = \Theta\left(2^{H_2(\frac{r}{n})n}\right)$$

we can show that this algorithm has expected complexity:

$$\Theta\left(2^{\alpha(k_{\max},r)n}\right)$$

where $H_2(p) = -p\log_2 p - (1-p)\log_2(1-p)$ is the binary entropy function and $\alpha(k,r) = H_2\left(\frac{r}{n}\right) - \left(1-\frac{k}{n}\right) H_2\left(\frac{r}{n-k}\right)$. The running time increases with both error rate and information rate.

The security level of a cryptosystem is measured using the amount of computational resources needed to break it. It is often expressed in terms of 'bits', where a cryptosystem that is b-bit secure requires 2^b operations to be broken. For asymmetric cryptosystems, this security level is obtained using the running time of the best known attacks on them. Using the ISD, attack, we can now obtain the following necessary condition for our cryptosystem to have b-bit security:

$$\alpha(k_{\max}, r)n \geq b \tag{4.1}$$

Hence, if we want a cryptosystem with a b-bit security requirement, we need to choose a permutation code with the appropriate parameters. Most commercial applications today require a security level of 128 or even 256 bits in some cases.

So far, we have described a framework to develop cryptosystems using permutation codes and outlined generic attacks on those cryptosystems, whose effectiveness depend on the parameters of the chosen permutation code H. Next, we attempt to use the permutation codes described in Chapter 3 in our cryptosystems.

4.3 Our cryptosystem using wreath product groups

The security and the performance of our cryptosystem depends on the choice of the group H and its decoding algorithm. In this section, we explore using the wreath product groups with the alternate decoding algorithm discussed in Section 3.3 for this purpose. We show that, using the appropriate parameters, it can be made resistant to information set decoding attacks. The decoding algorithm can be however, modified to work in the conjugate group $\widehat{H}$ as well, which would make the cryptosystem using it insecure. Although this cryptosystem is insecure, it does provide a concrete example of a permutation code with an efficient decoding algorithm which cannot be efficiently decoded using the ISD algorithm.

Let H be the wreath product group $C_m \wr S_n$ (more generally, we can take H to be a subgroup of $C_m \wr S_n$ with suboptimal decoding). We consider the case of $m = 5$ for different values of n, for different probabilities of correct decoding. Because the actual minimal degree of this group is 5, there will be cases in which the receiver of the encrypted word cannot decode correctly and will receive a wrong word. In these cases, the receiver will be able to tell that the decoding is wrong (possibly by using an appended hash function) and ask the sender the resend the message. Consider the case for which the probability of correct decoding is 0.9 and 0.95. We find the largest such value of r for which r errors are introduced and can be corrected for at least 0.9 or 0.95 fraction of the cases for different values of n, and estimate the amount of computational resources required for an ISD based attack to break the cryptosystem. As we can see, the computational resources needed to break this cryptosystem is non trivial and increases with n, although it is still not close to commercial levels of security. We also repeat this exercise for the case of $m = 7$. For $m = 7$, we can attain higher security levels for smaller values of n compared with $m = 5$. These results mean that, by increasing m and n to an appropriate amount, we can reach commercial levels of security such as 128 or 256 bits. Our analysis of larger values of m and n however, are limited by our computational resources, as our computer program for calculating the proportion of error patterns that can be corrected involves iterating over integer partitions, which is computationally intensive.

4.3.1 Attack using block systems

For a permutation group G acting on Ω, a complete block system $\mathcal{B}$ is a partition of Ω into disjoint sets $B_1, \ldots, B_k$ called blocks, such that if two points x and y are in the same block, then for all permutations $g \in G$, $x \cdot g$ and $y \cdot g$ are also in the same block. It can be shown that every block in $\mathcal{B}$ have the same cardinality if G is transitive.

The first property of block systems that we shall prove is that conjugation preserves the block structure of a permutation group.

Theorem 4.3.1. *Consider a permutation group G with a block system $\mathcal{B} = \{B_1, \ldots, B_k\}$. The conjugate permutation group $G' = \sigma^{-1} G \sigma$ has the block system $\mathcal{B}' = \{B_1 \cdot \sigma, \ldots, B_k \cdot \sigma\}$.*

Proof. Every permutation in G' is of the form $\sigma^{-1} g \sigma$ where g is a member of G. Consider

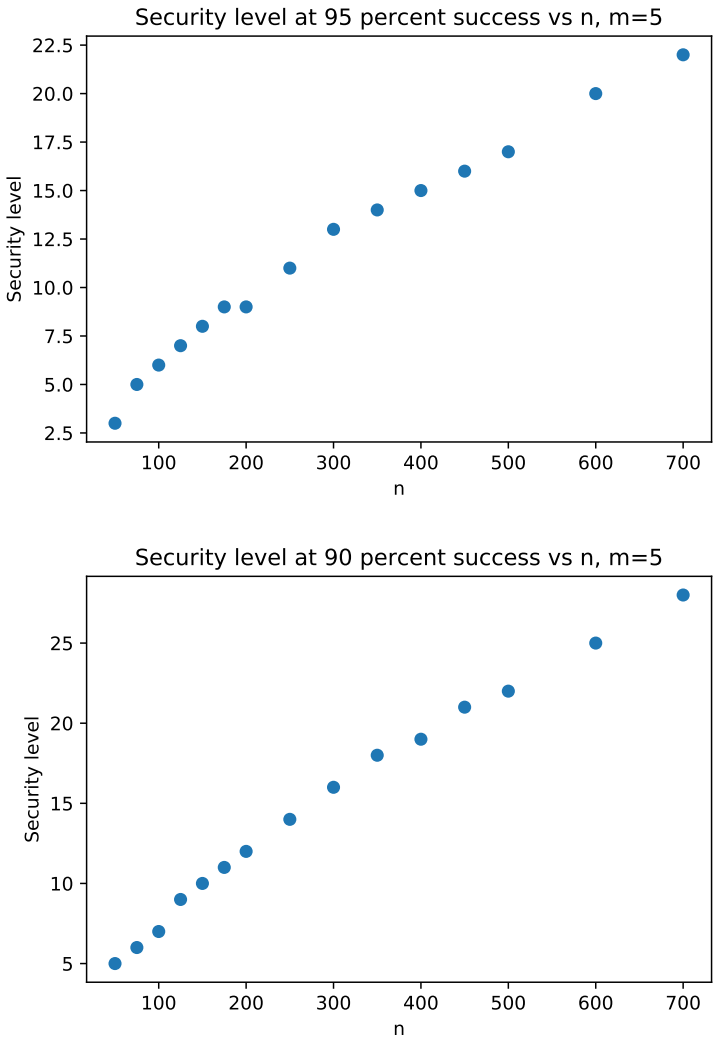

Figure 4.1: Security levels for $m = 5$

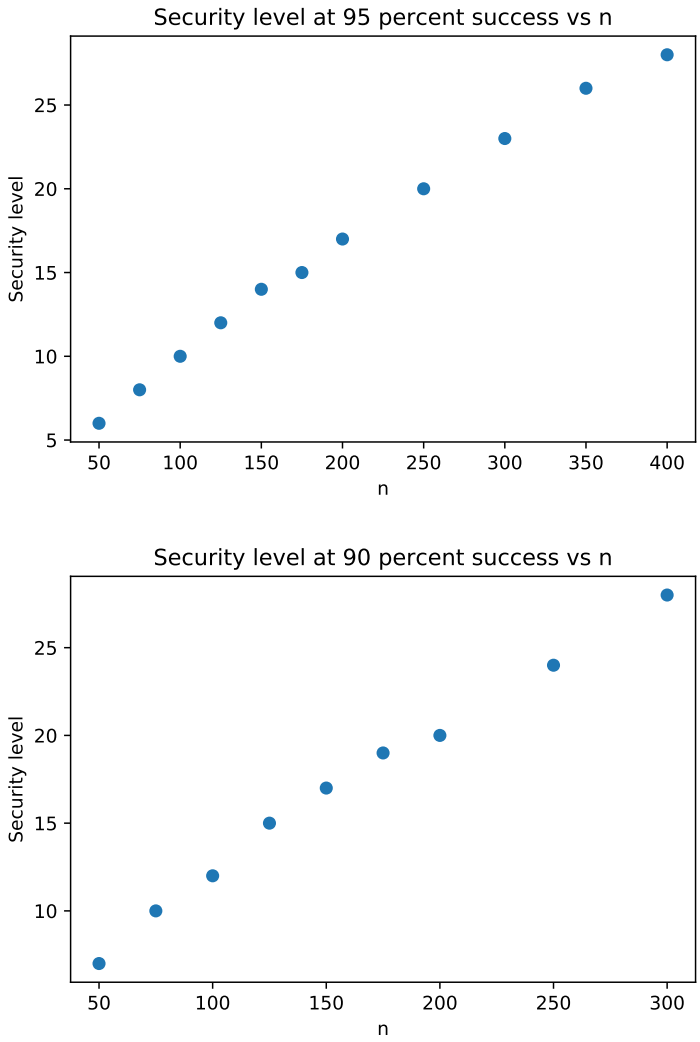

Figure 4.2: Security levels for $m = 7$

any two points x and y in the set $B_i \cdot \sigma$.

$$x \cdot \sigma^{-1} g \sigma = x' \cdot g\sigma \text{ and } y \cdot \sigma^{-1} g \sigma = y' \cdot g\sigma$$

As x' and y' are in the same block B_i, $x' \cdot g$ and $y' \cdot g$ are in the same block of $\mathcal{B}$, which means that $x \cdot \sigma^{-1} g \sigma$ and $y \cdot \sigma^{-1} g \sigma$ are in the same clock of $\mathcal{B}'$ □

For the wreath product group $C_m \wr S_n$ acting on a set of mn points it is easy to see that the columns form a maximal block system, with the group C_m acting on the blocks and the group S_n permuting them. Hence, the public key group $\sigma^{-1} G \sigma$ has a corresponding block structure. First, we observe that the alternative decoding algorithm to be modified for a more generic type of group, with a block system such that the group acts regularly on each block.

Consider any permutation group acting on a set Ω with a block system consisting of n blocks of size m each. It is well known that this permutation group can be described as a subgroup of the wreath product group $S_m \wr S_n$. Every permutation is described by how it acts on each block and by how it permutes the blocks among themselves. In this case the permutation group also acts regularly on each block.

Let the conjugate of G be G'. We compute a maximal block system for G' (or, if we use a subgroup of G, we can look for a block system with blocks with the required size). We then compute the generators for a subgroup of G' which stabilises this block system. This subgroup will act regularly on each block. We call these regular subgroups $H_1, H_2, \ldots, H_n$. An element of a regular subgroup is uniquely determined just by its action on one point. This element can also be computed using the Schreier-Sims algorithm.

Hence, for each position in Ω, we compute the block labels and relabel the received word accordingly. For each position, we then compute the element of the corresponding regular group and the value of the block permutation. We then compute the most frequently occurring value of these elements in each block to obtain a group element of the form $(h_1, h_2, \ldots, h_n, \sigma)$ where $h_i \in H_i$ and $\sigma \in S_n$ and convert it back to permutation form.

Hence, the same decoding algorithm can be deployed on the public key group, breaking the cryptosystem.

4.4 Our cryptosystem using the symmetric group acting on 2-subsets

We now examine the use of a group whose decoding algorithm will not work on its conjugates. Because of its parameters however, it can be trivially broken using ISD attacks.

Let G be the permutation image symmetric group S_m acting on the Ω which is defined to consist of all the two sets of $\{1, \ldots, m\}$. Hence, $n = \frac{m(m-1)}{2}$ and we choose H to be a subgroup of G . The conjugate $\widehat{H} = g^{-1}Hg$ is chosen such that $\widehat{H}$ is not a subgroup of G. We use the decoding algorithm using the UBB described in Section 3.2.1. As H is a subgroup of G, the same decoding algorithm can be used, although the decoding might be suboptimal because the minimal degree of H might be greater than that of G. Unlike the case of the alternative decoding algorithm for the wreath product groups however, we cannot easily construct a UBB for the conjugate group $\widehat{H}$ using the same techniques used to construct one for H. To recall the UBB is constructed using the hamiltonian cycles of a complete graph. The group H, as a subgroup of S_n, preserves the graph structure of K_m. This is what that means. Consider two 2-sets (i, j) and (i, k) in Ω. These two sets have one point in common. The 2-sets $(i, j) \cdot h$ and $(i, k) \cdot h$ must also have a point in common for all $h \in H$. No other conjugates of H in S_n have this graph preserving property however, and without that, it is unclear how to construct a UBB for $\widehat{H}$ without knowledge of the connjugator g.

The next thing we attempt to do is to attack the second security assumption. That is, we attempt to find a conjugator g and a subgroup H' of S_n that has an efficient decoding algorithm. We assume that these subgroups are all subgroups of G, as we can employ the UBB algorithm for them. While we do manage to come up with an attack which is better than a brute force search for g, the group H can still be chosen so that the cryptosystem is resistant to this attack.

4.4.1 Conjugator Search Attack

To recall, we are given a sequence of generators $\{\hat{h}_i\}$ for the group $\widehat{H}$. We attempt to construct a set of generators for a subgroup H' of G and $g \in S_n$ such that $g\widehat{H}g^{-1} = H'$. We

provide a sketch of the attack:

Consider any element of S_m. Its cycle type in the permutation image acting on 2-subsets is entirely determined by its cycle type acting naturally. It is possible however, for two different cycle types acting naturally to lead to the same cycle type acting on 2-sets. Two elements which are conjugate in S_m are also conjugate in S_n, but two elements can be conjugate in S_n but not in S_m.

Consider the elements h_1 and $\hat{h}_1$. The crucial thing to note is that it is enough to find the cycle type of the pullback $\hat{h}_1$ in S_m. This is because any element of the same cycle type of h_1 is conjugate to h_1 in S_m and hence also in S_n, and we can conjugate all the other generators also by the same element to obtain a different subgroup H' of S_m which would also have an efficient decoding algorithm.

Let us say we find an element of S_m with the same cycle type as h_1 (cycle type in S_n). The conjugator is unique up to a coset of the centralizer of $\hat{h}_1$ in S_n. Among all the elements in this coset, we need to find a conjugator that takes each $\hat{h}_i$ into S_m. The only way to do that (to our knowledge) would be by an exhaustive search. The bottleneck in the algorithm is this step. Finding $g \in S_n$ that conjugates an element $a \in S_n$ to an element $b \in S_n$ is not considered a difficult problem in practice and neither is computing the centralizer of an element as there are very effective backtracking methods to solve these problems [38, Section 9.1.2].

Complexity: The algorithm has to exhaustively search through the centralizer of one of the generators of $\widehat{H}$, and as we can choose the generator we start with, we can pick the generator with the smallest centralizer. Hence, for the cryptosystem to be resistant to conjugator search attack, we would need to pick the subgroup H of S_m such that its generators have large centralizers in S_n.

4.4.2 ISD attacks

The size of a base for H is less than $\log_2(m!) \leq m \log_2 m$. Hence, the number of bit operations required to break the cryptosystem using information set decoding is less than:

$$\left(H_2 \left(\frac{m-3}{m(m-1)/2} \right) - \left(1 - \frac{m \log_2 m}{m(m-1)/2} \right) H_2 \left(\frac{m-3}{m(m-1)/2 - m \log_2 m} \right) \right) \frac{m(m-1)}{2}$$

We plotted this as a function of m (Fig. 4.3) and we can see that the security level of even 80 bits isn't attainable for reasonable values of m.

The key reason why the cryptosystem using this code is insecure is the *parameters* of the code. Both the rate of the code and the correction capacity are too low. What we are aiming for is a code where the quantities k/n (on average) and r/n are asymptotically constant, like the binary Goppa codes. This requirement is satisfied by the wreath product groups however, using the alternative probabilistic decoding algorithm.

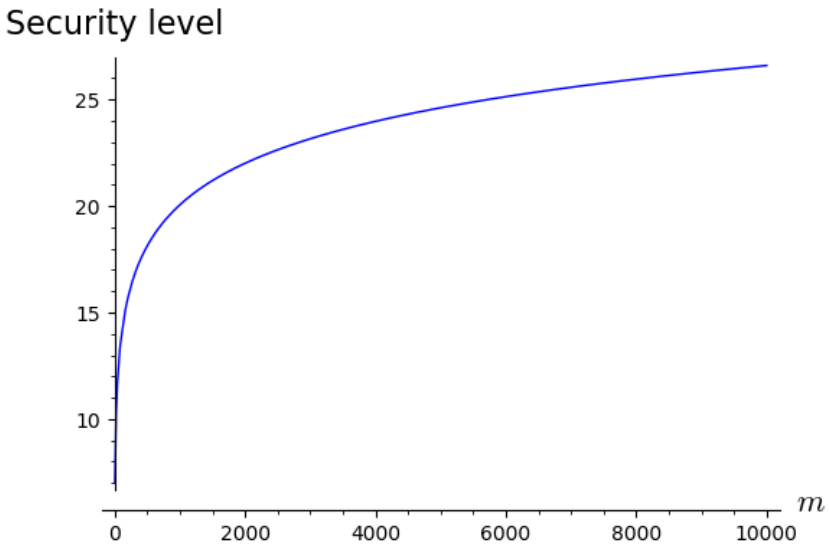

Figure 4.3: Security of cryptosystem using 2-sets

4.5 Linear codes as permutation codes

Linear codes are closely related to permutation codes. A linear code of block size n over the field $\mathbb{F}_q$ is an additive subgroup of $\mathbb{F}_q^n$ and a permutation code of block size n is a subgroup of S_n. A linear code C with minimum distance t which is a subgroup of $\mathbb{F}_q^n$ can be embedded naturally into $S_q^n < S_{qn}$ as a permutation code with minimal degree qt. This embedding is a distance expanding map. That is, a vector with weight w maps to a permutation with weight qw. For the case of binary linear codes, the i-th basis vector e_i of $\mathbb{F}_2^n$ is mapped to the permutation $(2i-1, 2i)$, and we can extend this to a homomorphism from $\mathbb{F}_2^n$ to S_2^n. For the q-ary case, the additive group $\mathbb{F}_q$ has a regular permutation homomorphism ϕ into

S_q. The vector qe_1, where $q \in \mathbb{F}_q$ is mapped to $\phi(q)$. Similarly, the vector qe_i is mapped to $c^{-qi}\phi(q)c^{qi}$, where $c = (1, 2, \ldots, n)$ is the permutation formed by a single cyclic shift. e_1 maps to a permutation acting on the first q points, e_2 maps to a permutation acting on the next q points, etc. This would extend to a homomorphism from $\mathbb{F}_q$ into S_q^n.

Hence, classical coding theory provides us with a rich source of permutation codes. However, for the purposes of our cryptosystems, permutation codes do not suffice. Rather, we need permutation codes with efficient decoding algorithms. The natural question to ask is- Do the classical linear code decoders effectively translate to the permutation code setting?

Consider a linear code C and $c \in C$. Once we introduce r errors to c to form a word w, the error pattern is $w - c$. For a permutation code G, once we introduce r errors to $g \in G$ to form a word w, the error pattern is wg^{-1}. We abuse notation slightly as this need not be a permutation.

Let us assume the code has an efficient decoding algorithm $\mathcal{A}$. We create a permutation code G from this, and introduce an error pattern e to g. Now, if e is also a member of S_2^n, then, we can employ the inverse of the described homomorphism to map this back to $\mathbb{F}_2^n$, and employ $\mathcal{A}$ to decode. However, the error pattern need not be in S_2^n, it just needs to have support of size less than $2t$. Hence the linear code decoding algorithms cannot always be used for the corresponding permutation code, and designing effective algorithms in that setting seems to be a hard problem. From the standpoint of cryptography, this problem can be worked around. This is because the person who sends the message introduces the errors and the error pattern is in their hands unlike in the usual coding theory applications! Hence, in our protocol, we can use a linear code as a permutation code, and when the message is sent with an error, there is an additional restriction that the error pattern belongs to S_2^n. Hence, our framework is a generalisation of traditional code-based cryptosystems.

Most linear code decoders rely extensively on syndrome decoding and the parity check matrix, which makes it difficult to extend them to permutation codes. A corresponding concept of a parity check matrix for permutation groups does not seem to exist (Section 2.3), which is a major inconvenience to designing potential permutation code decoders, even for permutation codes designed directly from linear codes.

4.6 Why use permutation codes?

So far, we have described a framework for using permutation codes in public key cryptography and explored this framework using two particular permutation codes. The cryptosystem using wreath product groups can be made resistant to ISD attacks but its decoding algorithm can be adapted for use in any of its conjugates too, which makes it unsuitable. For the symmetric group acting on 2-sets on the other hand, the decoding algorithm cannot be be modified for use in its conjugates, nor can the conjugator be uncovered easily from H and $\widehat{H}$. This is because the decoding algorithm depends on some very specific graph theoretical properties of the complete graph, and the conjugates of H cannot be modelled as the symmetric group acting on a complete graph. The parameters of this code however, mean that any cryptosystem using this is susceptible to ISD attacks. The question is, can we come up with a permutation code with parameters that make it resistant to ISD attacks and a decoding algorithm that cannot be modified for its conjugates? Our cryptosystem using such a permutation code would be secure against both the kind of attacks demonstrated earlier.

The primary quality which differentiates permutation groups from vector spaces is their non-commutativity, and it would be an interesting question to see if this would lead to any significant improvements in key size, speed, etc over traditional code based cryptosystems. For example, a rank k subspace of $\mathbb{F}_q^n$ needs k basis vectors to describe, each of length n. On the contrary, very large permutation groups can be generated by a very small number of generators. The symmetric group itself for example, can be generated using just two of its elements. This is a characteristic shared by many non-abelian permutation groups. Hence, a cryptosystem using permutation codes can potentially achieve a quadratic reduction in key size compared to its linear code counterparts for the same level of security! A linear code based cryptosystem using a code of length n over $\mathbb{F}_q$ would require a key of $O(n^2)$ as one of the components of the public key is a $k \times n$ matrix. By contrast, consider a permutation group of comparable size which is a subgroup of S_n which can be generated using just two generators. The space needed to store these generators would be just $O(n)$. One of the common complaints against the McEliece cryptosystem is its large key size so this would be a direction of research worth pursuing.

Chapter 5

The quantum security of our cryptosystem

The security of all currently known public key cryptosystems rely on some computational hardness assumptions. The security of the widely deployed RSA cryptosystem depends on the hardness of the integer factorisation problem which can be reduced to the order finding problem:

Problem 5.0.1. *Given a number N, f is a function from $\mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/N\mathbb{Z}$ given by:*

$$f(x) = a^x \text{ for a constant } a \in \mathbb{Z}/N\mathbb{Z}^\times$$

Find the period of f, which is the smallest natural number p such that $f(x+p) = f(x)$ for all $n \in \mathbb{Z}/N\mathbb{Z}$

Classically, this problem has been widely studied and is known to be hard, with no algorithm that runs in $poly(\log N)$ time. With a quantum computer however, this problem can be efficiently solved. In 1995, Peter Shor [41, 40] proposed a quantum algorithm that would solve these problems in polynomial time in input size. A similar algorithm can solve the discrete logarithm problem in finite fields and elliptic curves efficiently and can be used to break the ElGamal cryptosystem and Diffie Hellman key exchange protocols.

This means that if large-scale quantum computers were ever built, they will be able to break many of the public-key cryptosystems currently in use. Hence, there is substantial

interest in the cryptography community on building *post-quantum cryptosystems* that are secure against both classical and quantum attacks. The NIST has already started a process to update their standards to include post-quantum cryptosystems [3, 4]. The first step in this direction would be to identify computational problems that remain hard even for quantum computers and design trapdoor functions using them.

It turns out that nearly all the problems for which quantum computers provide an exponential speed-up can be cast into the framework of the *Hidden Subgroup Problem* (HSP). Given a finite group G and access to a function separating the cosets of a subgroup H of G, the problem requires us to compute a set of generators for H. The reason behind this speed-up is that it is possible to compute the Fourier transform of a function over a finite group efficiently in a quantum computer, while the analogous transformation can be done only in exponential time by a classical computer. Both the order finding problem and the discrete logarithm problem can be cast as hidden subgroup problems over abelan groups, for which the Fourier transform can be computed efficiently. This allows us to recover the hidden subgroup using only a polynomial number of queries and preprocessing time. The non-abelian hidden subgroup problem however, appears to remain hard for quantum computers and this provides us a promising new direction to design new, *post-quantum* cryptosystems.

In this chapter, we look at the basics of quantum computing and its relevance to public key cryptography. We start with defining the fundamental notions and concepts used in quantum computing. Next, we look at the hidden subgroup problem and the quantum Fourier sampling algorithm used to solve it. We then analyse the security of our cryptosystem in the face of quantum Fourier sampling attacks and show that it is quantum secure for regular permutation codes (Theorem 5.3.2).

5.1 Quantum data and measurement

Quantum computers are devices which manipulate and perform computations on quantum information. The fundamental unit of quantum information is a quantum bit, or *qubit*. Unlike classical bits, which can usually take one out of two values, a qubit can also be a *superimposition* of two values. Formally, a qubit is an L_2 normalized vector in a complex 2 dimensional vector space.

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{5.1}$$

where α and β are complex numbers and $|\alpha|^2 + |\beta|^2 = 1$. As is standard in quantum computation, we use $|\cdot\rangle$ to represent vectors. Using multiple qubits, we can also define quantum states storing more abstract representations of data. For example, consider a group G. We can define a superposition of group elements as:

$$|\psi\rangle = \sum_{g \in G} \alpha_g |g\rangle \text{ such that } \sum_{g \in G} |\alpha_g|^2 \tag{5.2}$$

This state is a superposition of the elements of G. We implicitly assume that there is a canonical way to represent elements of G as bit strings. The basis $\{|g\rangle, g \in G\}$ is referred to as the *computational basis*. For more details on the notation and qubits, we refer to [34].

For data to be extracted from the quantum states, they need to be measured. Measurement in quantum mechanics is a non-deterministic process with respect to a chosen basis for $\mathbb{C}[G]$. For an orthonormal basis $\mathcal{B} = \{b_1, \ldots b_n\}$ for an n dimensional space. Any state $|\psi\rangle$ can be expressed in terms of this basis as $|\psi\rangle = \sum_i \beta_i |b_i\rangle$. The measurement then yields b_i with probability $|\beta_i|^2$. After the measurement, the qubit collapses to one of the basis states and the remaining information in the qubit is lost. For example, when a measurement is performed with respect to the computational basis on the state described in equation 5.2, $|\psi\rangle$ collapses to $|g\rangle$ with probability $|\alpha_g|^2$.

Operations on quantum states would have to map normalized states to normalized states. Such operators are called unitary operators. A quantum computation could involve creating an initial state $|\psi\rangle$, performing a unitary operation on the state and measuring the resultant state. The unitary operators used are typically realised in terms of quantum circuits, with the complexity of the algorithms determined by the circuit size. The data obtained from the measurements can then be classically post-processed [34].

5.2 The Hidden Subgroup Problem

Problem 5.2.1 (Hidden Subgroup Problem)**.** *Let G be a finite group and Σ be a finite set. Let f be a function from G to Σ given by a black box oracle. There exists a subgroup H of G such that $f(x) = f(y)$ if and only if $xH = yH$. Find a set of generators for H with polynomial number of oracle queries.*

To solve the hidden subgroup problem, one has to find a set of generators for H in $poly(\log(|N|)$ time. In the decision version of the hidden subgroup problem, the goal is to distinguish the subgroup H from the identity subgroup $\langle e \rangle$. This is an easier problem than the computational version of the problem in which we have to uncover a set of generators. There are efficient quantum algorithms for the hidden subgroup problem over abelian groups, but the non-abelian version seems to be more complicated. For a review on this topic, we refer to Childs and Van Dam [16].

As an example, we cast the order finding problem as an instance of the hidden subgroup problem. The ambient group is the additive group $\mathbb{Z}/N\mathbb{Z}$. The function is the function $f(x) = a^x$ itself. If the period of this function is p, it is clear that the function is constant on all cosets on the subgroup of $\mathbb{Z}/N\mathbb{Z}$ consisting of all multiples of p, which would be the hidden subgroup. Upon finding a set of generators for this group, we can sample multiple elements from this group and find their common factor which would turn out to be p with high probability.

The generalisation of Shor's algorithm to solve the hidden subgroup problem in generic groups is the quantum Fourier sampling (QFS) algorithm, which uses the quantum Fourier transform as a building block. We first recall some definitions from the representation theory of finite groups. For a finite group G, a representation is a homomorphism $\rho : G \to GL_{d_\rho}(\mathbb{C})$. Let $\widehat{G}$ denote the set of all irreducible representations of G. For any $g \in G$ and $\rho \in \widehat{G}$, $\rho(g)$ is a $d_\rho \times d_\rho$ unitary matrix with complex valued entries, and $\rho(g)_{ij}$ is the entry in the i^{th} row and j^{th} column of $\rho(g)$. It is a well known result that $\sum_\rho d_\rho^2 = |G|$.

The group algebra $\mathbb{C}[G]$ denotes the vector space of all complex valued functions on G. Let $|G| = N$. Any member of $\mathbb{C}[G]$ can be represented as a vector in $\mathbb{C}^N$, with the group elements forming a standard basis. As is standard in quantum computing, we denote the states of the vector space $\mathbb{C}[G]$ spanned by group elements with a $|\cdot\rangle$, with a normalized vector in the standard basis denoted as

$$|\psi\rangle = \sum_{g \in G} \alpha_g |g\rangle \text{ where } \alpha_g \in \mathbb{C} \text{ and } \sum_{g \in G} |\alpha_g|^2 = 1$$

For more details on the notation, we refer to [34].

Definition 5.2.1 (Quantum Fourier Transform)**.** *The Quantum Fourier Transform over a*

group G is the following unitary transformation on $\mathbb{C}[G]$ that performs a change of basis:

$$|g\rangle \mapsto \frac{1}{\sqrt{|G|}} \sum_{\rho,i,j} \sqrt{d_\rho} \rho(g)_{ij} |\rho, i, j\rangle$$

where ρ is an irreducible representation of G, d_ρ is its dimension and $1 \leq i, j \leq d_\rho$. $|\rho, i, j\rangle$ form another orthonormal basis of $\mathbb{C}[G]$, the representation basis or the Fourier basis. This transformation is a unitary transformation.

Efficient quantum algorithms (of running time $poly(\log(|G|))$ exist to implement this transform efficiently over many groups of interest, in particular over the symmetric group [9]. We now briefly describe the standard method of QFS.

Initialise the quantum state to a uniform superposition over all the group elements, and apply f reversibly to a uniform superposition of G and measure the output register, obtaining the random coset state $|cH\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |ch\rangle$ for a random c. We then perform the quantum Fourier transform over this state and measure in the Fourier basis. We may either just measure the label of the representation ρ or also the row and column indices. The former is referred to as *weak Fourier sampling* and the latter as *strong Fourier sampling.*

Algorithm 5: Quantum Fourier Sampling (QFS)

Input: A two state black box operater U_f such that $U_f |x, y\rangle = |x, y \oplus f(x)\rangle$

Output: Generators of the hidden subgroup H

Procedure `QFS`(U_f)**:**

- $|\Psi\rangle \leftarrow \sum_{g \in G} |g, 0\rangle$
- $|\Psi\rangle \leftarrow U_f |\Psi\rangle$
- Measure the second component of $|\Psi\rangle$ to get $|cH\rangle = \frac{1}{\sqrt{|H|}} \sum_{h \in H} |ch\rangle$
- Perform QFS on $|cH\rangle$
- Measure ρ, i, j for strong QFS or measure just ρ for weak QFS

Classically post-process the information obtained from repeated measurements to obtain generators for H

This measurement leads to a probability distribution on these states:

$$P_{Hc}(\rho, i, j) = \frac{d_\rho}{|G||H|} \left| \sum_{h \in H} \rho(hc)_{ij} \right|^2 \tag{5.3}$$

As c is chosen uniformly and randomly, the probability distribution that we observe is $P_H = \frac{1}{|G|}\sum_c P_{Hc}$. For the case of weak Fourier sampling, as we average over the indices and only observe the label of the representation, the resulting probability distribution can be shown to be

$$P_H(\rho) = \frac{d_\rho}{|G|}\sum_{h\in H}\chi_\rho(h) \tag{5.4}$$

where χ_ρ is the character of the representation ρ. From repeated measurements, it is possible to gain some information about this probability distribution. To solve the decision version of the hidden subgroup problem, we need to be able to distinguish this probability distribution from the probability distribution corresponding to the identity subgroup, by classically post-processing this information. Distinguishing H from the identity subgroup is only possible if the L_1 distance between P_H and $P_{\{e\}}$ is greater than some inverse polynomial in $\log|G|$. For the case of weak QFS, this distance can be calculated to be:

$$\mathcal{D}_H = \frac{1}{|G|}\sum_{\rho\in\widehat{G}} d_\rho \left|\sum_{h\in H, h\neq e}\chi_\rho(h)\right| \tag{5.5}$$

If $\mathcal{D}_H \geq (\log|G|)^{-c}$ for some c, we say H is *distinguishable* using weak QFS. Otherwise, we say that H is *indistinguishable*. Crucially, if H is indistinguishable, it cannot be distinguished from the identity subgroup using a polynomial number of measurements. The converse however is not necessarily true. For example, in the dihedral group, all subgroups are distinguishable using the weak standard method, meaning that the method provides enough information to do so. There is however, no known method to efficiently post-process this information, and the most efficient quantum algorithm for that group is a sub-exponential algorithm by Kuperberg [29].

The hidden subgroup problem in the symmetric group has been extensively studied by Kempe and Shalev [28]. They completely characterised the subgroups of S_n which can be distinguished from the identity subgroup using weak QFS, showing that these are exactly the subgroups whose minimal degree is bounded above by an absolute constant. More precisely, they proved the following theorem [28, Theorem C]:

Theorem 5.2.1. *Let $H \leq S_n$ be a subgroup. If H is distinguishable, then it has a bounded minimal degree. Moreover, if $\mathcal{D}_H \geq n^{-c}$ where c is a fixed absolute constant, then $m(H) \leq g(c)$, where $g(x) = ax + b$ is some fixed linear function, independent of n or H.*

It is interesting that these are precisely the subgroups which can be distinguished from identity by classical search, by enumerating the permutations in S_n according to their support and conducting a polynomial number of membership tests.

The weak standard method has its limitations. For example, as the obtained probability distributions depend only on the characters, it cannot distinguish between conjugate subgroups. It also fails to provide sufficient information to solve the HSP in most non-abelian groups. The difference in strength between the strong and weak forms of Fourier sampling is an interesting question. The first thing to note is that the probability distribution obtained for the strong form of QFS depends on the basis we chose to for the representations. A natural way to do this would be to choose the basis randomly (with respect to the Haar measure for unitary matrices). This approach is useful in some cases. For example, it provides enough information to solve the HSP in the dihedral group and the Heisenberg group [37]. In many other cases of interest, like the symmetric group, however it is unhelpful. Grigni et al [23] addressed the difference in strength between weak QFS and strong QFS with a random choice of basis.

Firstly, they showed that measuring the row index i provides no additional information. The probability of measuring (ρ, i, j) does not depend on i. Hence the row index can be discarded. Secondly, they showed that if the H is sufficiently small and if G has a small number of conjugacy classes compared to it's size, the strong version of QFS using a random basis provides exponentially little information compared to the weak version. More precisely, for the case of the symmetric group, they showed that there exists an absolute constant δ such that if $|H| \leq |G|^{1/2-\delta}$ we need exponentially many samples to gain useful information by measuring j. It might be possible that performing strong QFS in a specific choice of basis might provide non-negligible information over the weak version, but as of now, we don't know of any example of a HSP for which a clever choice of basis leads to a significant improvement over a random basis.

There have been negative results on this front too, however. In 2008, Moore, Russell and Schulman [32] showed that strong QFS using any basis fails to provide enough information to distinguish subgroups of the symmetric group generated by fixed point free involutions from identity. Dinh et al [18] also provided a wider set of subgroups for generic groups in which strong QFS fails to provide enough information.

This concludes our review of the method of quantum Fourier sampling. As the problems

we are looking at are set in the symmetric group, henceforth we choose to just focus on the weak version of QFS, using Kempe and Shalev's approach.

5.3 QFS attacks on our cryptosystem

We consider an attack that involves finding the secret conjugator g given the private key and public key groups H and $\widehat{H}$. Even though H is a secret, we assume that a set of generators for both H and $\widehat{H}$ are available to the attacker. The question is, can they find the conjugator g from this information using QFS? Making H public would only make it easier for them. This would entail solving the following problem which is analogous to the *scrambler-permutation problem* described by Dinh et al [18].

Problem 5.3.1 (Subgroup Conjugacy Search Problem)**.** *Given the generating sets of two subgroups H and $\widehat{H}$ of S_n such that $\widehat{H} = g^{-1}Hg$ for some $g \in S_n$, find g.*

As any q-ary linear code can be embedded naturally as a subgroup of ${S_q}^n < S_{qn}$, every instance of the scrambler permutation problem can be reduced to an instance of this problem. The only known way to solve this problem is to reduce it to an instance of the hidden shift problem, which can in turn be reduced to an instance of the hidden subgroup problem over a wreath product.

Problem 5.3.2 (Hidden Shift Problem)**.** *Let G be a finite group and Σ be a finite set. Let f_0 and f_1 be two functions from G to Σ given by black boxes. An element $s \in G$ is called a shift if $f_1(x) = f_0(sx)$ for all $x \in G$. Given the promise that there is such a shift, find such a shift.*

Given two groups H and $\widehat{H}$, we can define the functions from $Sym(\Omega)$ to the set of all its subgroups- $f_0 : x \mapsto x^{-1}Hx$ and $f_1 : x \mapsto x^{-1}\widehat{H}x$, and $f_1(x) = f_0(gx)$ and $f_1(x) = f_0(gx)$ for all x. Hence, we can cast our problem as a hidden shift problem. The issue here is, unlike vector spaces, there is no canonical way to represent permutation groups as bit-strings. Without that, the usual hidden subgroup approach using QFS cannot be used. For the reminder of this section we assume that we have a labelling scheme for all subgroups of S_n along with an oracle which computes it.

The general procedure to solve this problem is to reduce it to an instance of the hidden subgroup problem [18, Proposition 3] over the wreath product group $S_n \wr \mathbb{Z}_2$. A member of $S_n \wr \mathbb{Z}_2$ is of the form $((x, y), b)$ where $x, y \in S_n$ and $b \in \mathbb{Z}_2$. The function is defined as follows:

$$f((x,y),b) = \begin{cases} (f_0(x), f_1(y)), & \text{if } b = 0 \\ (f_1(y), f_0(x)), & \text{if } b = 1 \end{cases}$$

The hidden subgroup is:

$$K := (K_0, g^{-1}K_0g), 0) \cup ((K_0g, g^{-1}K_0), 1)$$

where $K_0 = \{x \in S_n : f_0(x) = f_0(1)\}$ which is the normalizer of H in S_n. $S_n \wr \mathbb{Z}_2$ is a subgroup of S_{2n} acting imprimitively on the disjoint union $\Omega \sqcup \Omega$. As this is a subgroup of the symmetric group $Sym(\Omega \sqcup \Omega)$, we can use Kempe and Shalev's approach to prove that the subgroup K is indistinguishable.

5.3.1 Proof of indistinguishability of K

If the subgroup K is indistinguishable, that means that our cryptosystems using the subgroup H cannot be broken in time polynomial in n. Let us assume that there exists a constant c such that $\mathcal{D}_K \leq O(n^c)$. That means that the minimal degree of K must be less than $g(c)$. Assume that the minimal degree of K is less than $g(c)$. Any element in $((K_0g, g^{-1}K_0), 1)$ must swap the two copies of Ω, and hence must be fixed point free. Consider any element in $(K_0, g^{-1}K_0g), 0)$, say $((x, y), 0)$. x acts on the fist copy of Ω and y acts on the second copy of Ω. The support of this element of the union of the supports of x and y. Hence the minimal degree of K is lower bounded the minimal degree of K_0 and K has an element of bounded support if and only if K_0 has an element of bounded support. K_0 is the normalizer of H in S_n. We prove that if the minimal degree of the normalizer of a transitive permutation group is bounded, the minimal degree of the group also must be bounded.

Theorem 5.3.1. *Consider any transitive permutation group $H \leq S_n$. If $m(N(H)) \leq c$ where $N(H)$ is its normalizer in S_n, then $m(H) \leq 2c$*

Proof. Let $C(H)$, the centralizer of H in S_n. It is a subgroup of $N(H)$. Now we see what happens if $C(H)$ has an element of bounded support. Let σ be an element of bounded

support with support $Supp(\sigma)$. Now, we prove that $Supp(\sigma)$ is entirely contained in an orbit of H. If it is not, there exists $i \in Supp(\sigma), \tau \in H$ such that $\tau \cdot i \notin Supp(\sigma)$. Consider the action $\sigma\tau$ on i. τ moves i out of $Supp(\sigma)$ and hence, $\sigma\tau \cdot i = \tau \cdot i \notin Supp(\sigma)$. The permutation σ moves i to some point $j \neq i$ and hence, $\tau\sigma \cdot i = \tau \cdot j \neq \tau \cdot i = \sigma\tau \cdot i$. Hence, τ and σ do not commute and this means that $\sigma \notin C(H)$. We have proved that if $C(H)$ has an element of bounded support, H has an orbit bounded by that same quantity. As $Supp(\sigma)$ is an orbit of H, H must be a subgroup of a direct product of two permutation groups, one acting on $Supp(\sigma)$ and the other on $\Omega \setminus Supp(\sigma)$. However, as H is a transitive group, this is a contradiction.

Now, assume that the set $N(H)\backslash C(H)$ contains an element of bounded support. Let σ be that element whose support is bounded above by c. There exists $\tau \in H$ such that $\sigma\tau \neq \tau\sigma$. Also, as σ is a member of the normalizer of H, $\sigma^{-1}\tau^{-1}\sigma\tau \in H$ and it is not the identity. We now prove that $|Fix(\sigma^{-1}\tau^{-1}\sigma\tau)| \geq n - 2c$, which means that $|Supp(\sigma^{-1}\tau^{-1}\sigma\tau)| \leq 2c$.

For any $i \in \Omega$, if i and $\tau \cdot i$ belong to $Fix(\sigma)$, then i belongs to $Fix(\sigma^{-1}\tau^{-1}\sigma\tau)$. Hence,

$$|Fix(\sigma^{-1}\tau^{-1}\sigma\tau)| \geq |Fix(\sigma) \cap \tau^{-1} \cdot Fix(\sigma)|$$

There are only c points in Ω that are outside $Fix(\sigma)$, and $\tau^{-1} \cdot Fix(\sigma)$ has the same size as $Fix(\sigma)$. Hence the intersection of these two sets must have size at least $n - 2c$. This means that $|Supp(\sigma^{-1}\tau^{-1}\sigma\tau)| \leq 2c$. As $\sigma^{-1}\tau^{-1}\sigma\tau \in H$, this means that the minimal degree of H must be less than or equal to $2c$. □

This theorem means that, for the case of a regular group, as both the examples we considered were, we have obtained a condition for the cryptosystem to be quantum secure. To break this cryptosystem in time $O(n^c)$, the minimal degree of K_0 must be less than or equal to $g(c)$ and this means that the minimal degree of H must be less than or equal to $2g(c)$. As H is the error correcting code, we know its minimal degree (or a lower bound on it at least). Essentially, for the QFS attack to succeed in polynomial time, the minimal degree of H needs to be upper bounded by an absolute constant. Notably, this means that the cryptosystem using the symmetric group acting on 2-sets is quantum secure. It does however mean that the cryptosystem using the wreath product groups and the alternative decoding algorithm may not be quantum secure. Hence, we have proved the following theorem:

Theorem 5.3.2. *Consider our cryptosystem using the subgroup $H \leq S_n$ as the permutation*

code, with H being transitive and having minimal degree $m(H)$. Consider the hidden subgroup K obtained using the QFS attack. If $\mathcal{D}_K \geq O(n^{-c})$, then $m(H) \leq 2g(c)$, where $m(H)$ is the minimal degree of H and g is a fixed linear function.

If $m(H)$ is bounded by $g(c)$, we can decode in $\widehat{H}$ using exhaustive search itself in time $n^{2g(c)}$ by listing all the possible permutations within that distance of the ciphertext and testing membership in $\widehat{H}$. This is assuming that we are not using a decoding algorithm with an error probability and correcting more errors than the correction capacity of the code. Asymptotically, this result means that if we can break this cryptosystem in polynomial time using QFS, we can also break it in polynomial time using exhaustive search.

In this chapter, we have reviewed the literature on the technique of quantum Fourier sampling. We have discussed an attack on our cryptosystem similar to the attack on the McEliece and Neiderreiter cryptosystems using the scrambler permutation problem and were able to obtain a condition for the cryptosystem using regular groups to be quantum secure.

Chapter 6

Conclusion and Further Research

In this thesis, we have studied some computational problems in permutation group theory and applied them to develop a new public-key cryptosystem. In Chapter 2, we studied the membership testing problem in permutation groups. To do so, we introduced the fundamental concept of a base and a strong generating set and presented an algorithm that runs in nearly linear time in the input size. We then framed a generalisation of this problem, the subgroup distance problem which we prove to be **NP**-complete. We then compare these problems and algorithms to their well known counterparts in the world of linear algebra and coding theory. In Chapter 3, we study some novel work by Bailey and Cameron on the use of permutation groups as error-correcting codes. We studied the alternative decoding algorithm in more detail using computational techniques which helped us gain more insight into how it works.

Most of the original work in this thesis is presented in Chapter 4 and Chapter 5. We defined a framework for McEliece cryptosystems using permutation codes instead of linear codes. We also proved that this cryptosystem is secure against the Quantum Fourier sampling attacks similar to those employed against linear code based cryptosystems, provided the cryptosystem is classically secure. Unlike the situation with linear codes however, this proof does not depend on the specific permutation code used and is valid for all permutation codes. We believe that using permutation codes has the potential to lead to significant reductions in key sizes compared to existing linear code based cryptosystems. However, using existing permutation codes proposed by Bailey and Cameron, this cryptosystem is insecure. The

reason for the insecurity of the cryptosystem using the symmetric group acting on 2-sets is simply the parameters of the code which makes it susceptible to ISD attacks. Hence, there is a motivation to come up with permutation codes with improved parameters. The wreath product cryptosystem is insecure for a more subtle reason. The decoding algorithm only uses the block structure of the group, which is preserved by the conjugation operation. Hence, we would also like cryptosystems that depend on specific algebraic properties of the private key group, which are not preserved by conjugation. However, it would be interesting to see if we can come up with new permutation codes with better parameters and decoding methods for them. Unlike linear codes, which have a plethora of decoding algorithms and techniques, the only two algorithms known for linear codes are the ones discussed by Bailey in his thesis. Going beyond the UBB framework to come up with new decoding methods is an interesting problem, and the work in this thesis provides a motivation for doing so.

Bibliography

[1] Miklós Ajtai. "Generating hard instances of lattice problems". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 99–108.

[2] Miklós Ajtai and Cynthia Dwork. "A public-key cryptosystem with worst-case/average-case equivalence". In: *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. 1997, pp. 284–293.

[3] Gorjan Alagic et al. *Status report on the first round of the NIST post-quantum cryptography standardization process*. US Department of Commerce, National Institute of Standards and Technology, 2019.

[4] Gorjan Alagic et al. "Status report on the second round of the NIST post-quantum cryptography standardization process". In: *US Department of Commerce, National Institute of Standards and Technology* (2020).

[5] Brian Alspach. "The wonderful Walecki construction". In: *Bulletin of the Institute of Combinatorics and its Applications* 52 (Jan. 2008).

[6] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[7] Vikraman Arvind and Pushkar S Joglekar. "Algorithmic problems for metrics on permutation groups". In: *International Conference on Current Trends in Theory and Practice of Computer Science*. Springer. 2008, pp. 136–147.

[8] Robert F. Bailey. "Uncoverings-by-bases for base-transitive permutation groups". In: *Designs, Codes and Cryptography* (2006).

[9] Robert Beals. "Quantum computation of Fourier transforms over symmetric groups". In: *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. 1997, pp. 48–53.

[10] Elwyn Berlekamp, Robert McEliece, and Henk Van Tilborg. "On the inherent intractability of certain coding problems (corresp.)" In: *IEEE Transactions on Information Theory* 24.3 (1978), pp. 384–386.

[11] Daniel J Bernstein, Tanja Lange, and Christiane Peters. "Attacking and defending the McEliece cryptosystem". In: *International Workshop on Post-Quantum Cryptography*. Springer. 2008, pp. 31–46.

[12] Richard E Blahut. *Algebraic codes for data transmission*. Cambridge university press, 2003.

[13] Ian F Blake, Gérard Cohen, and Mikhail Deza. "Coding with permutations". In: *Information and Control* 43.1 (1979), pp. 1–19.

[14] Christoph Buchheim, Peter J Cameron, and Taoyang Wu. "On the subgroup distance problem". In: *Discrete mathematics* 309.4 (2009), pp. 962–968.

[15] Peter J Cameron and Dmitrii G Fon-Der-Flaass. "Bases for permutation groups and matroids". In: *European Journal of Combinatorics* 16.6 (1995), pp. 537–544.

[16] Andrew M. Childs and Wim van Dam. "Quantum algorithms for algebraic problems". In: *Rev. Mod. Phys.* 82 (1 Jan. 2010), pp. 1–52. DOI: 10.1103/RevModPhys.82.1. URL: https://link.aps.org/doi/10.1103/RevModPhys.82.1.

[17] John D. Dixon and Brian Mortimer. *Permutation groups*. Graduate texts in Mathematics 163. Springer, 1996.

[18] Russell A. Dinh H. Moore C. "McEliece and Niederreiter Cryptosystems That Resist Quantum Fourier Sampling Attacks". In: *Advances in Cryptology – CRYPTO 2011* 6841 (2011), pp. 761–779. DOI: https://doi.org/10.1007/978-3-642-22792-9_43.

[19] Robert F. Bailey. "Error-correcting codes from permutation groups". In: *Discrete Mathematics* 309 (2009), pp. 4253–4265.

[20] Robert F. Bailey. "Permutation groups, error-correcting codes and uncoverings". PhD thesis. University of London, 2006.

[21] *GAP – Groups, Algorithms, and Programming, Version 4.11.0*. The GAP Group. 2020. URL: %5Curl%7Bhttps://www.gap-system.org%7D.

[22] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman & Co., 1979. ISBN: 0716710447.

[23] Michelangelo Grigni et al. “Quantum Mechanical Algorithms for the Nonabelian Hidden Subgroup Problem”. In: *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*. STOC ’01. New York, NY, USA: Association for Computing Machinery, 2001, pp. 68–74. ISBN: 1581133499. DOI: 10.1145/380752.380769. URL: https://doi.org/10.1145/380752.380769.

[24] Richard W Hamming. “Error detecting and error correcting codes”. In: *The Bell system technical journal* 29.2 (1950), pp. 147–160.

[25] I. Blake. “Permutation codes for discrete channels”. In: *IEEE Transactions on Information Theory* (1974).

[26] Peter J. Cameron. “Permutation codes”. In: *European Journal of Combinatorics* 31.2 (2010), pp. 482–490.

[27] Ingo Janiszczak and Reiner Staszewski. “Isometry invariant permutation codes and mutually orthogonal Latin squares”. In: *Journal of Combinatorial Design* (2019).

[28] Julia Kempe, László Pyber, and Aner Shalev. “Permutation groups, minimal degrees and quantum computing”. In: *Groups, Geometry, and Dynamics* 1 (Aug. 2006). DOI: 10.4171/GGD/24.

[29] Greg Kuperberg. “A subexponential-time quantum algorithm for the dihedral hidden subgroup problem”. In: *SIAM Journal on Computing* 35.1 (2005), pp. 170–188.

[30] Jessie Macwilliams. “Permutation decoding of systematic codes”. In: *The Bell System Technical Journal* 43.1 (1964), pp. 485–505.

[31] Robert J McEliece. “A public-key cryptosystem based on algebraic coding theory”. In: *Coding Thv* 4244 (1978), pp. 114–116.

[32] Cristopher Moore, Alexander Russell, and Leonard J Schulman. “The symmetric group defies strong Fourier sampling”. In: *SIAM Journal on Computing* 37.6 (2008), pp. 1842–1864.

[33] Harald Niederreiter. “Knapsack-type cryptosystems and algebraic coding theory”. In: *Prob. Control and Inf. Theory* 15.2 (1986), pp. 159–166.

[34] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

[35] Christiane Peters. “Information-set decoding for linear codes over $\mathbb{F}_q$”. In: *International Workshop on Post-Quantum Cryptography*. Springer. 2010, pp. 81–94.

[36] E. Prange. “The use of information sets in decoding cyclic codes”. In: *IRE Transactions on Information Theory* 8.5 (1962), pp. 5–9.

[37] Jaikumar Radhakrishnan, Martin Rötteler, and Pranab Sen. “Random measurement bases, quantum state distinction and applications to the hidden subgroup problem”. In: *Algorithmica* 55.3 (2009), pp. 490–516.

[38] Àkos Seress. *Permutation group algorithms*. Cambridge tracts in Mathematics. Cambridge university press, 2002.

[39] Claude E Shannon. “A mathematical theory of communication”. In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

[40] Peter W Shor. “Algorithms for quantum computation: discrete logarithms and factoring”. In: *Proceedings 35th annual symposium on foundations of computer science*. Ieee. 1994, pp. 124–134.

[41] Peter W Shor. “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer”. In: *SIAM review* 41.2 (1999), pp. 303–332.

[42] Charles C Sims. “Computation with permutation groups”. In: *Proceedings of the second ACM symposium on Symbolic and algebraic manipulation*. 1971, pp. 23–28.

[43] Charles C Sims. “Computational methods in the study of permutation groups”. In: *Computational problems in abstract algebra*. Elsevier. 1970, pp. 169–183.