Research Article

Adarsh Srinivasan and Ayan Mahalanobis*

A McEliece cryptosystem using permutation codes

https://doi.org/10.1515/jmc-2024-0029
Received August 26, 2024; accepted July 14, 2025; published online November 21, 2025

Abstract: This paper is an attempt to build a new public-key cryptosystem, similar to the McEliece cryptosystem, using permutation error-correcting codes. We study a public-key cryptosystem built using two permutation error-correcting codes. We show that these cryptosystems are insecure. However, the general framework in these cryptosystems can use any permutation error-correcting code and is interesting. We present an enhanced McEliece cryptosystem, which subsumes the McEliece cryptosystem based on linear error correcting codes.

Keywords: McEliece cryptosystem; permutation error correcting codes; postquantum cryptosystems

MSC 2020: 94A60; 94B60

1 Introduction

McEliece and Niederreiter cryptosystems are very popular these days. One of the reasons behind this popularity is that there are some instances of these cryptosystems that can resist *quantum Fourier sampling attacks*. These make them secure against attacks by quantum computers. Such cryptosystems are called *quantum-secure cryptosystems*.

McEliece and Niederreiter cryptosystems use linear error-correcting codes. Similar to linear error-correcting codes, we can define **permutation error-correcting codes**. These codes use permutation groups the same way linear codes use vector-spaces. This paper is an attempt to build secure public-key cryptosystems using permutation codes in the same spirit as McEliece and Niederreiter cryptosystems were built using linear error-correcting codes. Public-key cryptosystems that came out of permutation codes are not secure. However, the journey we took to build these cryptosystems is interesting in its own right. Moreover, we were able to **enhance the original McEliece cryptosystem** by embedding it into a permutation group. Unlike the case of linear codes, there do not seem to exist families of permutation codes with enough combinatorial and algebraic richness to resist simple attacks. However, we hope to motivate more research into permutation codes through this work. We have used ideas and concepts from the theory of permutation groups freely, Seress [1, Section 1.2.2] or Cameron [2] are good references for permutation groups.

In this paper, we ask two questions:

Q1 Can one build a secure public-key cryptosystem using permutation codes, similar to cryptosystems built using linear error-correcting codes?

Q2 Are there any advantages of using permutation codes compared to linear codes in public-key cryptosystems?

***Corresponding author: Ayan Mahalanobis**, IISER Pune, Pune, India, E-mail: ayan.mahalanobis@gmail.com
Adarsh Srinivasan, Department of Computer Science, Rutgers University, New Brunswick, USA, E-mail: adarshsrinivasan256@gmail.com

We try to keep this paper as self-contained as possible. In the next section, we present a brief overview of permutation codes. Most of it is a review of Bailey's work [3–5] on permutation codes. We have performed some extra computational analysis on alternate decoding in Section 2.3.1. The section after that presents a public-key cryptosystem using permutation codes whose security depends on the hardness of decoding generic permutation codes. We study this cryptosystem using two permutation codes – the symmetric group acting on 2-subsets and a class of wreath product groups. We show that both these cryptosystems are insecure. The first one is insecure due to an information set decoding attack and the latter due to an inherent structure in the wreath product. We then present the enhanced McEliece cryptosystem in Section 3.5.

2 Permutation error-correcting codes

The use of sets of permutations in coding theory (also referred to as permutation arrays) was studied since the 1970s. Blake et al. [6, 7] were the first to discuss using permutations this way. They had certain applications in mind. One such example was powerline communications. However, permutation codes never got the level of attention that linear codes did. Bailey [5] 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*. They were also the first to exploit the algebraic structure of groups, to come up with decoding algorithms for several families of groups.

In this section, we explore the use of permutation groups as error-correcting codes. Let $[n]$ be the set $\{1, 2, \ldots, n\}$. A permutation on $[n]$ is defined to be a bijection from $[n]$ to itself. The set of all permutations form a group called the *symmetric group* and is denoted by S_n and e its identity element. Subgroups of S_n are called *permutation groups*. For a permutation $\pi \in S_n$, we define its support $\mathrm{Supp}(\pi) := \{i \in [n]\colon \pi(i) \neq i\}$ and set of fixed points $\mathrm{Fix}(g) := \{i \in [n]\colon \pi(i) = i\}$. A permutation 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 π acting on $\{1, 2, 3\}$ such that $\pi(1) = 2, \pi(2) = 3, \pi(3) = 1$. The other way to represent permutations is product of disjoint cycles. In this paper, we will use the list form more often than the product form. We now define the Hamming distance between two permutations.

Definition 2.1 (Hamming Distance). For $\sigma, \pi \in S_n$, the Hamming distance between them, $d(\sigma, \pi)$ is defined to be the size of the set $\{i \in [n] | \sigma(i) \neq \pi(i)\}$.

The Hamming distance between two permutations π, σ can be expressed using support and fixed points of $\sigma\pi^{-1}$.

Proposition 2.1. *For any two permutations $\sigma, \pi \in S_n$, $d(\sigma, \pi) = |\mathrm{Supp}(\sigma\pi^{-1})| = n - |\mathrm{Fix}(\sigma\pi^{-1})|$.*

Definition 2.2 (Permutation code). A permutation code with minimum distance d is a set $C \subseteq S_n$ such that $d = \min_{\sigma,\pi \in C} d(\sigma, \pi)$.

In this paper, we assume that the permutation code C forms a subgroup. We call such a permutation code a permutation group code. A permutation group is generated by a finite set called a set of *generators*.[1] For a permutation group code C, we can relate its minimum distance to supports of all nonidentity elements in C.

Definition 2.3 (Minimal degree). The minimal degree of a permutation group $G \subseteq S_n$ is defined to be

$$m(G) = \min_{\sigma \in G,\, \sigma \neq e} |\mathrm{Supp}(\sigma)| = n - \max_{\sigma \in G,\, \sigma \neq e} |\mathrm{Fix}(\sigma)|.$$

1 Every permutation in the permutation group can be expressed as a product of permutations from the set of generators.

The quantity $r = \left\lfloor \frac{m(G)-1}{2} \right\rfloor$ is called the *correction capacity* of the code. The distance between a permutation σ and a group G, $d(\sigma, G)$ is the distance between σ and the permutation in G closest to it. That is, $d(\sigma, G) = \min_{g \in G} d(\sigma, g)$. There is an ambient group in which both σ and G belong. For a proof of the next theorem, see Bailey [3, 5].

Proposition 2.2. *The minimal degree of a permutation group code is equal to its minimum distance.*

The security of public-key cryptosystems rests on some hardness assumption. Our hardness assumption is based on the *subgroup distance problem* [8].

Problem 2.1 (Subgroup distance problem).
Input: A set of generators S for a permutation group $G \subseteq S_n$, a permutation $\sigma \in S_n$.
Output: A permutation $\pi \in G$ minimizing $d(\pi, \sigma)$.

We can also generalize this problem to consider inputs σ to be lists of length n with symbols from $[n]$, which do not necessarily form a permutation. A nearest neighbor 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 neighbor need not be unique. However, if $d(\sigma, G) \leq r$, the nearest neighbor of σ in G is unique. The subgroup distance was shown to be NP-complete [8]. In fact, even the decision version of this problem, which is easier than the search version, is NP-complete. For the purpose of building cryptosystems, the average-case hardness of a problem is more important than the worst-case complexity. While there does not exist a worst case to average case reduction, it is believed that the problem of linear error correcting codes is hard for several families of random linear codes. As we formally prove in Section 3.5, the subgroup distance problem is a generalization of the linear code decoding problem, indicating that the average case hardness of the subgroup distance problem for some families of permutation codes may be a reasonable security assumption for developing public key cryptosystems [9].

A permutation group 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 was received, r or fewer errors were introduced. Such a channel is called a *Hamming channel*. The received message is a list w of length n with symbols from $[n]$. Note that w need not be a permutation. Because the correction capacity of G is r, the nearest neighbor of w in G is uniquely determined 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 decoding is possible in theory, because of the fact that the *subgroup distance problem is NP-complete*, generic algorithms cannot be used to decode permutation codes efficiently. We need a specific algorithm tailor-made for a particular family of permutation codes.

2.1 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 $G \subseteq S_n$ with correction capacity r acting on the set $\Omega = [n]$. For a permutation σ and $i \in \Omega$, we use $i \cdot \sigma$ to denote the element $\sigma(i)$. We start with defining a *base* for a permutation group, which is the analogue of a basis of a vector space. For more information, we refer to Seress [1, Chapter 4].

Definition 2.4 (base). A base for a permutation group G acting on Ω is $B = \{b_1, b_2, \ldots, b_m\}$, such that, $B \subseteq \Omega$ with the property that all elements in B is fixed only by the identity element of G.

Given a base $B = \{b_1, b_2, \ldots, b_m\}$, define subgroups $G = G^{[0]}$ and $G^{[1]}, \ldots, G^{[m]}$, such that, $G^{[i]} := \left\{\sigma \in G | \sigma(b_j) = b_j \quad \text{for all } j \leq i\right\}$ where $i = 1, 2, \ldots, m$. It is easy to see that

$$G = G^{[0]} \geq G^{[1]} \geq \ldots \geq G^{[m]} = \{e\}. \tag{1}$$

Algorithm 1: ElementReconstruction

```
Input: A base $B = \{b_1, \dots, b_m\}$ for $G \leq S_n$ and $X = \{x_1, \dots, x_m\}$
Output: $\sigma \in G$ such that $b_i \cdot \sigma = x_i$ for $i = 1$ to $m$. false if no such $\sigma$ exists
1  $\sigma \leftarrow e$
2  for $i = 1$ to $m$ do
3      Compute the fundamental orbit $\mathcal{O}_i = b_i \cdot G^{[i-1]}$ and the transversals $R_i$ as a dictionary, with
       $b_i \cdot R_i[x] = x$
4  for $i = 1$ to $m$ do
5      if $x_i \in \mathcal{O}_i$ then
6          $\sigma \leftarrow \sigma R_i[x]$
7      else
8          return false
9      for $j = 1$ to $m$ do
10         $x_j \leftarrow x_j \cdot R_i[x]^{-1}$
11 return $\sigma$
```

We say that a base B is *irredundant* if $G^{[i]} \neq G^{[i+1]}$ for all i. A generating set S of G is called a strong generating set if $\langle S \cap G^{[i]} \rangle = G^{[i]}$ for $1 \leq i \leq m$, where $\langle S \rangle$ is the group generated by the set S. Given a group G, a base and a strong generating set for it can be constructed in nearly linear time in the size of the input using the Schreier–Sims algorithm. 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 $G^{[i+1]}$ in $G^{[i]}$ 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. It follows from the definition of a base that every permutation in G is uniquely defined by its action on the elements of a base. There exists an algorithm (Algorithm 1), called the element reconstruction algorithm that can reconstruct the permutation from its action on all the elements in the base.

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 no such σ exists and we can exit the procedure. We then replace $(x_1, \dots, x_m)$ with $(x_1 \cdot r_1^{-1}, \dots, x_1 \cdot r_m^{-1})$. In the i-th iteration, we check if $x_i \cdot r_1^{-1} r_2^{-1} \dots r_{i-1}^{-1}$ lies in the orbit of G_{b_i}. If it does not, we exit the procedure; else we replace σ with σr_i; where r_i is the permutation in the transversal R_i that takes b_i to $x_i \cdot r_1^{-1} r_2^{-1} \dots r_{i-1}^{-1}$. More details on the Schreier–Sims algorithm is available in Seress [1, Chapter 4].

Definition 2.5 (UBB). 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 (an r-subset of Ω is a subset of Ω of size r); 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 [3, Proposition 7]. However, they have been constructed only for a few permutation groups. There is no known procedure to construct a small UBB for an arbitrary permutation group. Uncovering-by-bases can be used to decode as follows:

Let G be a permutation group acting on Ω. Consider a permutation $g \in G$ in the list form and w was obtained after r or fewer errors were introduced to g. Note that w can have repeated entries and need not be a permutation. Let R be a subset of Ω of size less than or equal to r. Errors were introduced in positions from R. The set R is 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. Thus, in the positions indexed by B, the lists w and g are identical. As B is a base, we can reconstruct the entire permutation g from just its action on the points in B. While the decoder does not know which base in $\mathcal{U}$ is the appropriate base B, it can just repeat this procedure for every base in $\mathcal{U}$.

```
Algorithm 2: 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 $\Omega$
  Output: $g \in G$ such that $d(\sigma, w) \leq r$. false if no such $g$ exists
1 for $B \in \mathcal{U}$ do
2     if $w$ has no repeated symbols in positions indexed by $B$ then
3         $g' = \mathrm{ElementReconstruction}(w, S, B)$
4         if $g' \neq$ false and $d(g, g') \leq r$ then
5             return $g'$
6 return false
```

This algorithm is similar to *permutation decoding* for linear codes proposed by MacWilliams and Sloane [10], which involves using a subset of the automorphism group of the code that moves any errors out of the information positions.

2.2 Some constructions of uncoverings-by-bases

There are only a few examples of UBB that were constructed by Bailey. One of them is a symmetric group acting on 2-subsets, and the other is a wreath product of a regular permutation group with a symmetric group. We discuss them next.

2.2.1 Symmetric group acting on 2-sets

Let m be a positive integer. Let Ω be the set of all subsets of $[m]$ of size 2.

Any permutation $\sigma \in S_m$ acts on a set $\{i, j\} \in \Omega$ as $\{i \cdot \sigma, j \cdot \sigma\}$. Now the group G is S_m acting on Ω, and it has minimal degree $2(m-2)$ and correction capacity $m-3$ [3, 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. Thus, we can use graph theoretic results 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 [3, Fig. 2]. 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 1890s, Walecki [11] showed that K_m can be decomposed into $(m-1)/2$ Hamiltonian cycles when m is odd. 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 edges of K_m avoids at least one of these V-graphs. For more on this topic, the reader is referred to Bailey [3]. This UBB is of size $O(m)$, and each base is of size $O(m)$. Thus, we can use this UBB in the decoding algorithm, which would have time complexity $O(m^4) = O(n^2)$. This is because n is the number of edges in K_m, which is equal to $m(m-1)/2$.

2.2.2 Wreath product of regular groups

Let H be a group acting on a set Δ where $m = |\Delta|$. The action of H on Δ is said to be regular if, for every $x, y \in \Delta$, there exists exactly one $h \in H$ such that $x \cdot h = y$. Having defined the action of H on Δ, we can view H as a subgroup of S_m, and such a permutation group is called a *regular permutation group*. We now define the permutation group code $G = H \wr S_n$ to be the wreath product of H, a regular permutation group acting on m points with the symmetric group S_n. The group G acts imprimitively on mn points.

We define elements of G as all tuples of the form $(h_1, h_2, \dots, h_n, \sigma)$, where $h_i \in H$ and $\sigma \in S_n$. We use the symbol 1 to denote the identity element of H and e for the identity element of S_n. In this section, we redefine $\Omega = [m] \times [n]$ consisting of n copies of the set $[m]$ as n columns each with m rows.

We now define the action of an element $g = (h_1, h_2, \dots, h_n, \sigma)$ on (i, j) to be $(i \cdot h_{i \cdot \sigma}, j \cdot \sigma)$. The group S_n acts on the columns and the group H acts on the rows. Thus, the group G can be defined as a subgroup of the symmetric group acting on Ω. As a regular group acting on m points has size m, $|G| = m^n n!$. See [2] for a proof of $|H| = m$.

Theorem 2.1. *The minimal degree of the wreath product group G is m and its error correction capacity is $\left\lfloor \frac{m-1}{2} \right\rfloor$.*

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

Theorem 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 $(h, 1, \dots, 1, e)$ fixes every point in S. So, S cannot be a base for G. To prove the converse, consider the following set $\{(x_1, 1), (x_2, 2), \dots, (x_n, n)\}$ with each x_i belonging to the ith column. Suppose here exists $g = (h_1, h_2, \dots, h_n, \sigma)$ that fixes each of these points. As $(x_i, i) \cdot g$ must lie in the ith column for each i, this implies that σ must be the identity permutation. Thus, 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. □

Using this theorem, we can now construct a UBB for G. Each row of Ω forms a base and a collection of $r + 1$ rows 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 is disjoint from at least one of those bases. The UBB can be used in Algorithm 2, which runs in $O(m^2 n^2)$ complexity.

2.3 An alternative decoding algorithm for wreath product

In this section, we present a simple decoding algorithm introduced by Bailey and Prellberg [12], which uses majority logic principles for the wreath product groups and compare its performance to the original UBB algorithm. For simplicity, we assume H to be the cyclic group C_m of order m, this algorithm can be extended to any regular group.

Input: For some $g \in C_m \wr S_n$, w is constructed from g by introducing r or less errors. The list w is the input.

Output: We aim to recover the permutation g. Because $d(g, w) \leq r$, there is a unique permutation in G with distance at most r from w.

The algorithm: To recover the permutation $g = (h_1, h_2, \dots, h_n, \sigma) \in H \wr S_n$, it suffices to recover h and σ.

1. For each $i \in [n]$, define $\sigma(i)$ be the majority value of the second coordinates of $w((j, i))$, for each $j \in [m]$. This way, we can recover the permutation $\sigma \in S_n$.
2. For each $i \in [n]$, let $w_{i,j}$ be the first coordinate of $w((j, i))$, for each $j \in [m]$, and let $w'_{i,j} = w_{i,j} - w_{i,j} \bmod m$. Let w'_i be the majority value of these $w'_{i,j}$'s for all $j \in [m]$, and we define the permutation $h_i \in C_m$ to be the permutation associated with the cyclic shift by w'_i.

We refer to ref. [12] for a proof of correctness of the above algorithm.

Complexity analysis: 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 take a constant amount of time for each position. Thus, it takes $O(mn)$ 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 that would take $O(n)$ time. This procedure is repeated for each block. For the final part of the algorithm, which involves converting the reconstructed word back to a permutation can be done with $O(mn)$ arithmetic operations. Thus, 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.

2.3.1 Decoding more than r errors

In fact, the algorithm described above can correct more than r errors, as long as the majority of elements in each block are correct. There are some error patterns with up to nr errors, which can be decoded using this algorithm. An error pattern is a certain set of positions where the errors are induced. Bailey obtained the following expression for the number of error patterns of a certain length, which can be corrected by this algorithm.

Let k be a positive integer. Let $P_{n,r}(k)$ be the set of all partitions of k into at most n parts where each part is of size at most r. We write 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 2.3. *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 [5, Section 6.7]. □

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

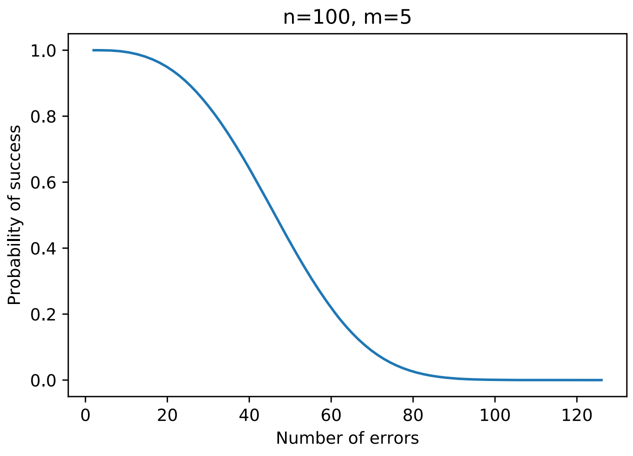

Figure 1: Plot of fraction of errors that can be corrected for $n = 100, m = 5$.

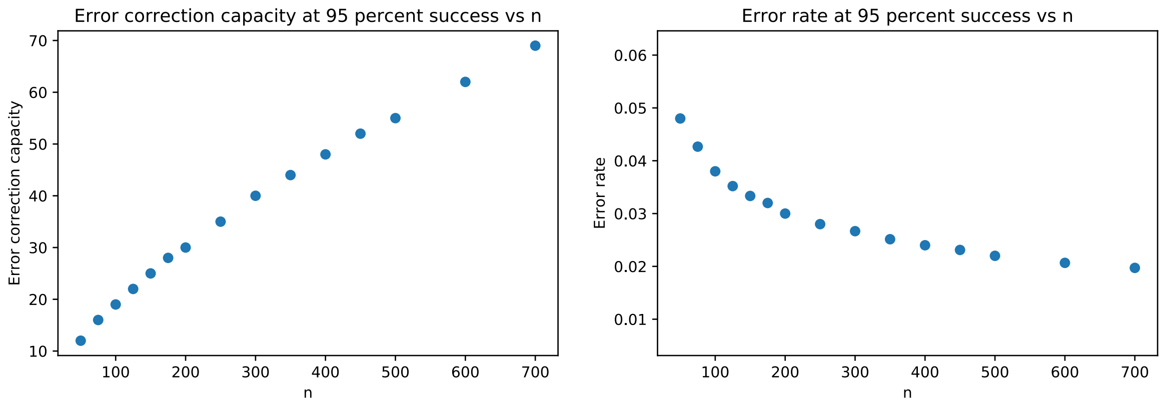

Figure 2: Plot of error correction capacity with 95 % probability of success.

Using a computer program, we calculated the value of $E_{n,r}(k)$ for different values of k and obtained the value of $\frac{E_{n,r}(k)}{mn}$ to obtain the probability that the alternate decoding algorithm succeeds. We plot this for $n = 100$ in Figure 1.

We can deduce from the data that 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.

Interestingly, the number of errors that can be corrected increases with n (for a fixed probability of success). This is despite the minimal degree (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 2).

3 A cryptosystem on permutation error-correcting codes

The McEliece cryptosystem is a very popular cryptosystem using linear codes, first proposed by Robert McEliece [13] in 1978. Due to its large key sizes, it never gained a lot of popularity at that time. Recently with the rise of quantum computing, it has gained a lot of attention along with its counterpart proposed by Neiderreiter [14] as a *postquantum* 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 3.1. 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 nonsingular $k \times k$ 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. Premultiplying G by a nonsingular matrix will not change the code. Postmultiplying G by a permutation matrix 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 fast decoding algorithm. However, the public key matrix generates a linear code do 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 section, we studied permutation groups as error-correcting codes. In this section, 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. Assume that ϕ is an isometry 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 $\widehat{H} = \phi(H)$ with no good decoding algorithm. The isometries in the symmetric group were studied and classified by Farahat [15] 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 should be homomorphisms. We would like both codes H and $\widehat{H}$ to be subgroups of S_n. This is because a permutation group can be represented efficiently using a generator set. A permutation array without a group structure cannot be represented efficiently this way. Thus, we are interested in isometries of S_n that are homomorphisms. We now combine these ideas to create a public-key cryptosystem. Our ingredients are a permutation code with a fast decoding algorithm and a conjugation map.

3.1 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. The private key is kept secret, and the public key is available to the public.

Private key Let G be a permutation group acting on n letters with a fast decoding algorithm. We choose a subgroup $H \leq G \leq S_n$ and a permutation g from S_n. The private key is a set of generators $\{h_1, \dots, h_s\}$ of H and g. We will use the decoding algorithm in G as a decoding algorithm for H.

Public key The public key is a set of elements $\{\hat{h}_i\} = \{g^{-1}h_i g\}$, which generate the conjugate group $\widehat{H} = g^{-1}Hg$. Note that a security assumption is that $\widehat{H}$ does not have a fast decoding algorithm. Note that $\widehat{H}$ might not be a subgroup of G.

Encryption The plaintext is $\hat{h}$ of $\widehat{H}$. We introduce r errors in $\hat{h}$ 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 neighbor in H. We then compute $\hat{h} = g^{-1}hg$.

The conjugation map is an isometry of the Hamming space. The distance between h and H is equal to the distance between $\hat{h}$ and $\widehat{H}$. In the McEliece cryptosystem, the Hamming isometry is postmultiplication by a permutation matrix. The secret scrambler scrambles the basis of a linear code to create a new basis. It does not change the linear code. 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.

Remark 3.1. There is a possibility of using isometries other than the conjugation map. However, the errors have to be introduced more carefully. For example, suppose we have a group $G \leq S_n$ and a map from G to G, which is an isometry on G. Our permutation code is a subgroup $H \leq G \leq S_n$. Thus, if we introduce errors in H so that the ciphertext c lies in the group G, the cryptosystem using this isometry will work. We have not studied this problem carefully and leave it for future work.

3.2 The security of our cryptosystem

3.2.1 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 $\{h'_1, \dots, h'_s\}$ 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.

In this paper, we present a cryptosystem that seems to satisfy (for known attacks) **AS1** but not **AS2** using the symmetric group acting on 2-subsets. The cryptosystem using wreath product group falls apart on **AS1** but is resistant to ISD like 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 neighbor 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.

For our cryptosystem to work, the group H must have 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. The security of the cryptosystem depends on the structure of the permutation code H used and its available decoding algorithm, and not just on parameters like size, correction capacity, etc. This is especially true in the context of structural attacks. Similar to McEliece cryptosystem, we cannot provide any theoretical basis for the claim that a cryptosystem using a certain permutation code is secure. 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 section 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.

We now present some generic attacks on our cryptosystem. Generic attacks are designed without taking into account the choice of the permutation code. Their effectiveness depends only on public parameters, like the correction capacity and size of the code. In Section 3.3, we demonstrate the existence of a code with parameters that make the cryptosystem secure against these attacks.

3.2.2 Brute force attacks

One obvious method to solve the nearest neighbor problem in $\widehat{H}$ is to enumerate all elements of $\widehat{H}$ and calculate their Hamming distance from the ciphertext. This implies that a necessary condition for b-bit security is that $|H| \geq 2^b$.

Another way is to enumerate all possible permutations within a distance r from the codeword c and check if they belong to $\widehat{H}$. 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.

3.2.3 Information set decoding

Information set decoding is one of the more successful attacks on the McEliece cryptosystem. The original idea was proposed by Prange [16] in 1962, and many improvements on this basic attack have been proposed [17, 18]. For a $[n, k]$ linear code, it involves picking k coordinates at random and trying to reconstruct the codeword from those positions. In this section, we recreate a version of Prange's ISD attack for cryptosystems using permutation

Algorithm 3: Information Set Decoding

```
  Input: generators of $\widehat{H}$, a ciphertext $c$
1 Procedure ISD():
2     Choose a base $B$ and SGS for $\widehat{H}$ arbitrarily
3     if There are no repeated symbols in positions indexed by $B$ then
4         Use element reconstruction algorithm to try and find an element $h \in \widehat{H}$ agreeing with $c$
            on these positions.
5         if $h$ exists and $d_H(h,c) \le r$ then
6             return $h$
7     return False
8 Procedure repeated till it succeeds
```

codes. We note that there are several notable improvements to Prange's original ISD algorithm. It is not clear to us whether these improvements can translate to permutation codes as they extensively use the algebraic structure of linear codes. We refer the reader to Weger et al. [9, Section 5] for an excellent survey of the ISD algorithms. We prove a sufficient (but not necessary) condition for a permutation code based cryptosystem to be resistant to ISD attacks.

This algorithm is similar to the uncovering-by-bases decoding algorithm proposed by Bailey, except that, we do not know the UBB and thus the running time is not bounded by the size of the UBB. The correctness of this algorithm follows from the proof in Bailey [3, Proposition 7] in which he shows that given any r-subset of $\{1,\dots,n\}$, 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. This occurs with a finite nonzero 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 where g is the secret conjugator.

Theorem 3.1. *Let $k_{\max}$ be the size of the irredundant base of largest size of H. The ISD algorithm succeeds in returning a plain text permutation $h \in \widehat{H}$ with probability $1-o(1)$ in time $O\left(2^{\alpha(k_{\max},r)n}\right)$, where $\alpha(k,r) = H_2\left(\frac{r}{n}\right) - \left(1-\frac{k}{n}\right)H_2\left(\frac{r}{n-k}\right)$ and $H_2(p) = -p\log_2 p - (1-p)\log_2(1-p)$ is the binary entropy function.*

Proof. The algorithm uses a procedure to choose an arbitrary base B for the group. 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 does not have any error reduces and 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 do not have any error positions is high. Thus, 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 $\{1,\dots,n\}$,

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

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

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

Let $k_{\max}$ be the size of the irredundant base of largest size of H. As the Pr[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 probability Pr[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 effectively parallelized. Using the following well-known formula for binomials,

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

we can show that this algorithm has expected complexity

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

□

The running time of this algorithm increases with both error rate and information rate. The value of $k_{\max}$, which is the size of the largest irredundant base of H, is the analogue of the rank of the permutation code H. For example, it can be shown that the value of $k_{\max}$ for the symmetric group S_n acting on 2-subsets of $[n]$ is $n - 1$ or $n - 2$, depending on the parity of n [19, Theorem 1.1]. The security level of a cryptosystem is measured from the amount of computational resources needed to break it. A cryptosystem is said to be b-bit secure if it requires 2^b operations to be broken. For asymmetric cryptosystems, this security level is upper bounded by the running time of the best known attacks on them. Using the ISD attack, a necessary condition for our cryptosystem to have b-bit security is $\alpha(k_{\max}, r)n \geq b$. If we want a cryptosystem with b-bit security, we need to choose a permutation code with the appropriate parameters.

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 Section 2.2 in our cryptosystems.

3.3 Our cryptosystem using wreath product groups

The security and the performance of our cryptosystem depend 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 2.3. We show that it can be made resistant to information set decoding attacks using appropriate parameters. However, the decoding algorithm can be modified to work in the conjugate group $\widehat{H}$ as well, which makes the cryptosystem 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 attacked 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 to 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 by plugging in this value of r into Equation (2). As we can see, the computational

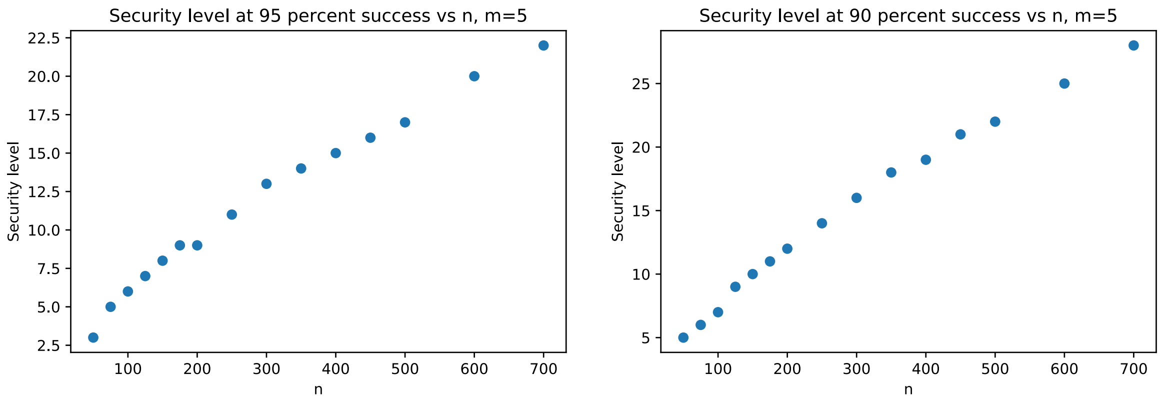

Figure 3: Security levels for $m = 5$.

resources needed to break this cryptosystem are nontrivial and increase 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 were 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. Judging by the trend in the graphs, however, we can conclude that the values of m and n can be appropriately increased to attain larger levels of security. For example, we can attain 128 bit security using the algorithm 95 percent success rate with $n = 1,645$, and with 95 percent success rate, we can achieve the same with $n = 1,257$ (Figures 3 and 4).

3.3.1 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 a block, then for all permutations $g \in G$, $x \cdot g$ and $y \cdot g$ are also in a 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.

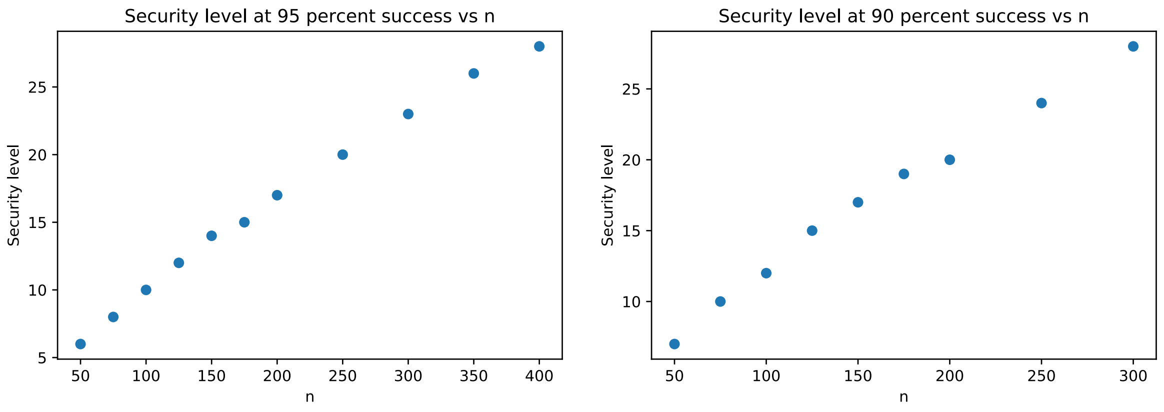

Figure 4: Security levels for $m = 7$.

Theorem 3.2. *Consider a permutation group G with a block system $\mathcal{B} = \{B_1, \dots, B_k\}$. The conjugate permutation group $G' = \sigma^{-1}G\sigma$ has the block system $\mathcal{B}' = \{B_1 \cdot \sigma, \dots, B_k \cdot \sigma\}$.*

Proof. Every permutation in G' is of the form $\sigma^{-1}g\sigma$ where g is a member of G. Consider any two points x and y in the set $B_i \cdot \sigma$. Then, $x \cdot \sigma^{-1}g\sigma = x' \cdot g\sigma$ 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 mn points, it is easy to see that the columns form a maximal block system, with the group H acting on the blocks and the group S_n permuting them. The public key group $\widehat{H} = \sigma^{-1}H\sigma$ has a corresponding block structure. First, we observe that the alternative decoding algorithm can be modified for a more generic type of group with a block system, such that, the group acts regularly on each block. Let $\mathcal{B}' = \{[m] \times \{1\} \cdot \sigma, [m] \times \{2\} \cdot \sigma, \dots, [m] \times \{n\} \cdot \sigma\}$ be the maximal block system of G'.

Input and output. For $h \in H \subseteq C_m \wr S_n$. Let $\widehat{h}\colon [m] \times [n] \to [m] \times [n]$ be a function obtained by introducing at most r errors to $\sigma^{-1}h\sigma$. The aim is to recover the permutation h (σ and h are hidden).

Step 1: recovering the block system. The first step is to recover the block system $\mathcal{B}'$ of the permutation group G'. There exist standard algorithms for this that run in linear time [1]. We then compute a set of generators for a subgroup of G', which stabilizes this block system. This subgroup will act regularly on each block. We call these regular subgroups $H_1, H_2, \dots, H_n$ (each of these are conjugate to C_m) and store their generators. 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, given the action on one point. We note that each element of $\widehat{H}$ can be described using the permutations $h_1, \dots, h_n \in H'$, where $h_i \in H_i$ and a permutation $\tau \in S_n$. Thus, it is enough to recover $h_1, \dots, h_n$ and τ.

Step 2: Recovering τ. For each $i \in [n]$, let $i \cdot \sigma$ be the label of the block that the majority of elements within the block B_i is moved to by $\widehat{h}$. Thus, we have recovered the permutation τ.

Step 3: Recovering $h_1, \dots, h_n$. As noted above, for each H_i, an element in H_i can be recovered given just the action on a single point. For every point in $\mathcal{B}_i$, consider the action of $\widehat{h}$ on it. This way, we can recover m permutations, and we set h_i to be the majority value among all these permutations.

3.4 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 symmetric group S_m acting on Ω the set of all two subsets of $\{1, \dots, m\}$. Let $n = \frac{m(m-1)}{2}$ and we choose H to be a subgroup of G. However, H is also a subgroup of S_n. For that there is a relabeling of the two subsets of $\{1, 2, \dots, m\}$ by elements of $\{1, 2, \dots, n\}$. This labeling is not publicly available. Thus publicly we see $H \leq G \leq S_n$.

We use the decoding algorithm using the UBB described in Section 2.2.1. 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 G. Unlike the case of the alternative decoding algorithm for the wreath product groups, we cannot easily construct a UBB for the conjugate group $\widehat{H} \leq S_n$ using the same techniques used to construct one for G. To recall, the UBB is constructed using the Hamiltonian cycles of a complete graph. Now the conjugation map is just relabeling. When this relabeling happens, the V-graphs changes arbitrarily making them not bases. There seems to be no way of constructing a UBB without finding the conjugator. Furthermore, note that, since H is a subgroup of G, there is insufficient information on the action of the G on the complete graph from which the UBB is constructed.

3.4.1 Conjugator search attack

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

Conjugacy in S_n vs. S_m. 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.[2] It is possible 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.

Determining cycle-type 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 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.

Searching in cosets of the centralizer. 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 [1, Section 9.1.2]. Thus, taking $g \in S_n$ does not help with the security of the cryptosystem.

3.4.2 ISD attacks

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

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

which is upper bounded by a polynomial in m. Hence, we can use information set decoding to break this cryptosystem in polynomial time. To demonstrate this more clearly, we plotted this as a function of m (Figure 5), and we can see that the security level of even 80-bits is not attainable for reasonable values of m.

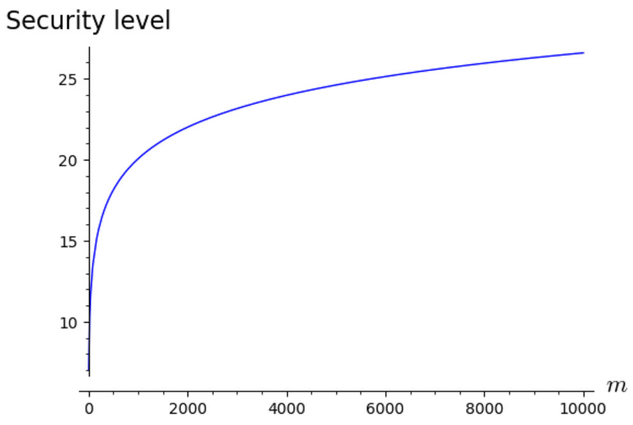

Figure 5: Security of cryptosystem using 2-sets.

2 We remind the reader that any permutation can be expressed as a product of different cycles, and this decomposition depends on the action.

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 using the alternative probabilistic decoding algorithm.

3.5 An enhanced McEliece cryptosystem

Permutation codes are permutation groups, which have less structure than linear codes which are vector spaces. Linear codes can also be seen as permutation codes. Let $q = p^t$ where p is a prime and t a positive integer. 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. We now *enhance the classical McEliece cryptosystem over linear codes* using permutation codes. For this, we describe an embedding.

Embedding a linear code into the symmetric group. We start with a monomorphism from the abelian group of $\mathbb{F}_q$ to the symmetric group S_{tp} where $q = p^t$ and p is a prime. The representation for $\mathbb{F}_q$ is the polynomial representation. This means that $\mathbb{F}_q \equiv \mathbb{F}_p[x]/\phi(x)$ where $\phi(x)$ is an irreducible polynomial of degree t. This says that every element of $\mathbb{F}_q$ is a polynomial $\alpha_0 + \alpha_1 x + \alpha_2 x^2 + \cdots + \alpha_{t-1}x^{t-1}$. Now, we fix t disjoint cycles $c_0, c_1, \dots, c_{t-1}$ of length p in S_{pt} and order them. One can choose $c_0 = (1, 2, \dots, p)$ as the first cycle, $c_1 = (p + 1, \dots, 2p)$ as the second, and so on. Note that disjoint cycles commute. Now we define a map, which we call the **basis map**, from the additive group of $\mathbb{F}_q$ to S_{pt}:

$$0 \mapsto (\,),\ \alpha_0 \mapsto c_0^{\alpha_0} \text{ and } \alpha_i x^i \mapsto c_i^{\alpha_i} \quad \text{for } i = \{1, 2, \dots, t-1\}.$$

Thus, $\alpha_0 + \alpha_1 x + \alpha_2 x^2 + \cdots + \alpha_{t-1}x^{t-1} \mapsto c_0^{\alpha_0} c_1^{\alpha_1} \dots c_{t-1}^{\alpha_{t-1}}$. It is easy to check that this map is an embedding – a monomorphism. Note that the map depends on the choice of the cycles. Now if we have a vector space V of dimension n over the field $\mathbb{F}_q$, corresponding to a fixed ordered basis $\mathcal{B}$ of V, every element of V is a vector of size n over $\mathbb{F}_q$ – the coordinate vector. These coordinate vectors depend on the fixed ordered basis of the vector space. Then, the embedding of this vector space in the symmetric group S_{pt}^n is done in the obvious way. A vector in $V = \langle v_1, \dots, v_n \rangle$, where the basis is $\mathcal{B} = (v_1, \dots, v_n)$, is of the form $\nu_1 v_1 + \cdots + \nu_n v_n$ where ν_i are field elements – the coordinates. Then, v corresponds to the ordered tuple $(\nu_1, \dots, \nu_n)$ of field elements. This ordered tuple is then mapped to an ordered tuple $(\hat{\nu}_1, \dots, \hat{\nu}_n)$ in S_{pt}^n a direct product of n copies of S_{pt} in the obvious way, using the basis map already defined. This embedding respects addition in the field and then in the vector space. It also respects scalar multiplication as long as the scalar is in the prime subfield. Thus, a linear code S being a subspace of the vector space V can be embedded as an elementary abelian subgroup of the symmetric group – a permutation group.

Let $S = \langle s_1, s_2, \dots, s_d \rangle$ be a d-dimensional subspace of a n-dimensional vector space V, which is a code with a good decoding algorithm. This code has a $d \times n$ generator matrix. This basis matrix depends on a fixed ordered basis $\mathcal{B}$ of V. One can define a classical McEliece cryptosystem based on this code S. As in a McEliece cryptosystem, we define the public key as $S' = ASP$ where A and P are the scrambler and permutation matrices, respectively. Now S' is also a $d \times n$ matrix over $\mathbb{F}_q$. One can use the basis map defined earlier to transfer this matrix to a matrix S_1 over S_{pt}. Here, S_1 is constructed by taking the embedding of each element in S'. Then, each element of S_1 is conjugated by g an element of the symmetric group S_{pt} and denoted by S_1 as well. Thus, $S_1 = g^{-1}S_1 g$. Recall that this conjugation is just a relabeling of the permutations.

In this enhanced McEliece cryptosystem, the public key is the rows of the matrix S_1. The whole structure of the classical McEliece cryptosystem S and S' is secret and so is the basis map and the conjugator g. The plaintext is $[a_1, \dots, a_n]$ where each a_i is in the $\mathbb{F}_p$ the prime subfield of $\mathbb{F}_q$. Then, one computes $a = \prod_{i=1}^{n} (S_1[i])^{a_i}$ where $S_1[i]$ is the ith row of S_1. Then, we introduce errors in a. This is the ciphertext.

On receiving the ciphertext, use the conjugator g to restore the original labeling. Then, one decomposes it as product of cycles. Then, one computes most of the a_i from the exponent. Then that gives rise to a vector of length n over $\mathbb{F}_q$ via the embedding. Assume that this is the ciphertext for the classical McEliece cryptosystem. Decrypt it using the classical McEliece cryptosystem. We will get the plaintext $[a_1, \dots, a_n]$ back.

There are few things to note. First, since the basis map is a secret, it enhances the classical McEliece cryptosystem. Thus, one might be able to use linear codes for the enhanced McEliece cryptosystem that is otherwise not secure. When it comes to cycles, computing the exponent has a lot of redundancy. Say, in a cycle after introducing errors, there is one point that gets repeated. Then, we can simply ignore the repeated points and compute the exponent. It is the action on one point that determines the exponent, when it comes to a cycle. Furthermore, a field element now has a permutation representation. Based on this representation, we might be able to introduce much more errors than is possible to fix for a permutation error correcting code. This topic, what is the right way to introduce errors and how many errors we can introduce, is left unsettled in this paper. Lastly, there is no need for the matrix S'. One can just move from S to S_1 directly without using S'. In this case, use a scrambler matrix over the prime subfield. Then, the extra step in decoding can be avoided.

Assume now there is a classical McEliece cryptosystem over linear codes that is secure. Then, for a suitable ciphertext of the original McEliece cryptosystem, one can map it to S_1 by the basis map. If the original plaintext is recovered from the ciphertext in the permutation setting, then that breaks the original McEliece cryptosystem. Note, we assume that the basis map is known in this case. The McEliece cryptosystem using Goppa codes has been remarkably resilient to both classical and quantum attacks. However, this McEliece cryptosystem is an enhanced version of the classical McEliece cryptosystem. It might be secure for other linear codes on which the original McEliece cryptosystem is not secure. Thus, this enhancement is bit more than a mere academic interest and deserves further study. However, at our present state of knowledge, it provides no respite from large key sizes that plagues the original McEliece cryptosystem.

3.6 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. On the other hand, for the symmetric group acting on 2-sets, the decoding algorithm cannot 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 modeled as the symmetric group acting on a complete graph. The parameters of this code 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 kinds of attacks demonstrated earlier.

Permutation groups differ from vector spaces in their noncommutativity, and it is an interesting question to ask if this would lead to any significant improvements in key size, speed, etc. over traditional linear 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 [2, Theorem 1.13]. For example, the symmetric group can be generated by two of its elements (the transposition (1, 2) and the cycle $(1, 2, \dots, n)$). This is a characteristic shared by many nonabelian permutation groups. Thus, 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 that 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 very large key size. So, this would be a direction of research worth pursuing.

Funding Information: AM was partially supported by an NBHM research grant and AS was supported by a INSPIRE fellowship. Both these fundings were from the Govt. of India.
Author contribution: Both authors contributed equally and approve the final version of the manuscript.
Conflict of Interest: Both authors report no conflict of interest.

Acknowledgments: This paper was part of the first author's Master's thesis at IISER Pune. Both authors thank Upendra Kapshikar for stimulating discussions. Both anonymous referees deserve a special mention for their detailed report, which has benefited this manuscript immensely. We used GAP [20] for our computations.

References

1. Seress Á. Cambridge tracts in mathematics. In: Permutation group algorithms. Cambridge: Cambridge University Press; 2003.
2. Cameron PJ. London mathematical society student texts. In: Permutation groups. Cambridge: Cambridge University Press; 1999.
3. Bailey RF. Error-correcting codes from permutation groups. Discret Math 2009;309:4253—65.
4. Bailey RF. Uncoverings-by-bases for base-transitive permutation groups. Des Codes Cryptogr 2006;41:153—76.
5. Bailey RF. Permutation groups, error-correcting codes and uncoverings [Ph.D. thesis]. University of London; 2006.
6. Blake I. Permutation codes for discrete channels. IEEE Trans Inf Theor 1974;20:138—40.
7. Blake IF, Cohen G, Deza M. Coding with permutations. Inf Control 1979;43:1—19.
8. Buchheim C, Cameron PJ, Wu T. On the subgroup distance problem. Discret Math 2009;309:962—8.
9. Weger V, Gassner N, Rosenthal J. A survey on code-based cryptography. [Online]; 2022. Available from: https://arxiv.org/abs/2201.07119.
10. Macwilliams J. Permutation decoding of systematic codes. The Bell Syst Tech J 1964;43:485—505.
11. Alspach B. The wonderful Walecki construction. Bull Inst Comb Appl 2008;52.
12. Bailey RF, Prellberg T. Decoding generalised hyperoctahedral groups and asymptotic analysis of correctible error patterns. Contrib Discrete Math 2012;7. https://doi.org/10.55016/ojs/cdm.v7i1.62080.
13. McEliece RJ. A public-key cryptosystem based on algebraic coding theory. In: The deep space network progress report, DSN PR 42—44; 1978:114—16 pp.
14. Niederreiter H. Knapsack-type cryptosystems and algebraic coding theory. Probl Control Inf Theor 1986;15:159—66.
15. Farahat HK. The symmetric group as a metric space. J Lond Math Soc 1960;35:215—20.
16. Prange E. The use of information sets in decoding cyclic codes. IEEE Trans Inf Theor 1962;8:5—9.
17. Peters C. Information-set decoding for linear codes over $\mathbb{F}_q$. In: International workshop on post-quantum cryptography. Springer; 2010:81—94 pp.
18. Bernstein DJ, Lange T, Peters C. Attacking and defending the McEliece cryptosystem. In: International workshop on post-quantum cryptography. Springer; 2008:31—46 pp.
19. Gill N, Lodà B. Statistics for S_n acting on k-sets. J Algebra 2022;607:286—99.
20. The GAP Group. GAP — groups, algorithms, and programming, version 4.11.0; 2020. Available from: https://www.gap-system.org/.