Spring 4-26-2018

Non-commutative Massey-Omura Encryption with Symmetric Groups

Shannon Haley

Non-commutative Massey-Omura Encryption with Symmetric Groups

Shannon Haley

submitted in partial fulfillment of the requirements for Honors in Mathematics at the University of Mary Washington

Fredericksburg, Virginia

April 2018

This thesis by **Shannon Haley** is accepted in its present form as satisfying the thesis requirement for Honors in Mathematics.

Date	Approved
_____	_____
	Randall Helmstutler, Ph.D. thesis advisor
_____	_____
	Keith Mellinger, Ph.D. committee member
_____	_____
	James Collins, Ph.D. committee member

Contents

Abstract

We introduce two non-commutative variations on the original Massey-Omura encryption system using conjugations in the symmetric group S_n. Patented in 1986, the original system was based on the cyclic group $\mathbb{F}^*$ of units in a finite field $\mathbb{F}$. In place of the abelian group $\mathbb{F}^*$, we will work in the non-abelian group S_n using disjoint permutations as well as maximal abelian subgroups in order to potentially create a more secure system. Introducing the non-abelian group S_n presents the need to create a keyspace of commuting permutations and abelian subgroups of sufficient size. We analyze the security of our modified systems by examining the bit-level security of each and susceptibility to standard message attacks. Additionally, we find that the keycount for the first system grows factorially with n. We show that the keycount for the second variation grows exponentially with n while improving on the first modification by allowing any number of users to participate in communication.

1 Introduction

Mathematical cryptology is the study of creating and analyzing secure methods of communication over a public or insecure channel. Cryptographic protocols are used to allow two private parties, commonly referred to as Alice and Bob, to communicate by concealing sensitive information with difficult mathematics. These methods, called *encryption systems*, convert *plaintext* messages into *ciphertext* messages. Alice and Bob are called *validated users* as the two communicators that are permitted to possess special information about the encryption system, called the *key*. This key is what Alice uses to encrypt a plaintext message and what Bob uses to decrypt a ciphertext message back to the original plaintext. The hope in any encryption system is that only these two validated users possess the key which any adversaries or eavesdroppers, referred to as Eve, cannot recover in order to read their secrets.

Many of the earliest cryptographic protocols were *symmetric* or *private key* systems in which only the two verified users hold the same private key. Since the key is necessary to both encrypt and decrypt messages, only those two users can use the system. Although these cryptosystems are still used today, one of the disadvantages to symmetric protocols is that each distinct pair of people needs to establish and exchange their secret key prior to communicating. More recently, since their discovery in the 1970s, *asymmetric* or *public key* systems have become the more popular form of cryptographic protocol. In this type of system, Alice publishes a *public* encryption key that allows anyone to communicate with her. However, Alice will also have a *private* decryption key of her own that allows only her to decrypt messages. While it may seem that public key systems would be impossible to keep secure because of the public information, their security is based on one of several mathematical problems that are known to be significantly difficult to solve. These problems will be discussed in greater detail later.

The Massey-Omura protocol is the cryptosystem that will be the focus of this paper. This is a cryptosystem that was patented in 1986 by James L. Massey and Jimmy K. Omura having properties of both a public and private key system [3]. One of the most advantageous features of this system is that anyone can participate by creating their own keys, which resembles the asymmetric characteristic of a public key protocol. However, in a more private-key fashion, each person that communicates in Massey-Omura keeps both their encryption and decryption keys as secrets. Thus, it is hard to place this scheme into one category, as it is somewhat of a hybrid between the two.

A commonly used analogy for understanding this system envisions Alice placing a lock on a safe, Bob placing his own lock on the safe, Alice removing hers, and finally Bob removing his in

order to access the contents of the safe. At all times the secret, or in this case the contents of the safe, is kept secure and only the two authorized users can place and remove their locks from the safe in order to unlock the secret message.

As we will see shortly, this scheme works because of the commutative property of exponents. However, our goal will be instead to use non-abelian groups, in which commutativity does not hold, in order to make the algebra of the system more difficult. This harder algebra will hopefully increase the security of the system and make it less vulnerable to attacks.

2 The Classical Massey-Omura Scheme

2.1 The Original Protocol

The Massey-Omura encryption system, originally patented in 1986 by James Massey and Jimmy Omura [3], is a variation on a public key system and is an example of a *three-pass protocol*. In this system, Alice and Bob agree on a public parameter that allows them to create their own private encryption and decryption keys. By design, Massey-Omura allows for virtually any number of communicators to participate, while still keeping each key private. In this way, Massey-Omura is somewhat of a hybrid of a public and private key system. We will now present the setup of this system.

Massey-Omura in $\mathbb{Z}_p^*$

1. Alice and Bob agree on a large public prime p.
2. Alice selects two private numbers e_A and d_A such that $e_A d_A \equiv 1 \bmod p-1$.
3. Bob selects two private numbers e_B and d_B in the same manner.
4. Alice sends a message $m \in \mathbb{Z}_p^*$ to Bob by computing $m^{e_A} \bmod p$.
5. Bob raises this message to e_B and sends $m^{e_A e_B}$ back to Alice.
6. Alice raises this message to her decryption key d_A and sends the ciphertext $c = m^{e_A e_B d_A}$ back to Bob.
7. Bob decrypts by raising c to d_B and obtains the original message m.

Note that p is a public parameter, so any number of people can join the communication without causing Alice and Bob to change their keys. Now we will show why this encryption system is sound. That is, we will prove that Bob will always obtain Alice's original message upon decryption.

In order to prove this works, we need Euler's Theorem and a direct corollary to show that Massey-Omura is a valid system. This theorem is a basic result of algebra and number theory.

Theorem 2.1 (Euler's Theorem)**.** *Let p be a prime and let a be an integer such that p does not divide a. Then*

$$a^{p-1} \equiv 1 \bmod p.$$

Corollary 2.2. *Let p be a prime and a be an integer such that p does not divide a. If $x \equiv y \bmod p-1$, then $a^x \equiv a^y \bmod p$.*

Proof. Let $x, y \in \mathbb{Z}$ and let p be a prime. Also, let $a \in \mathbb{Z}$ such that p does not divide a. Because $x \equiv y \bmod p-1$, there exists some $k \in \mathbb{Z}$ such that $y = x + k(p-1)$. Then working mod p, we have

$$\begin{aligned} a^y &= a^{x+k(p-1)} \\ &= a^x a^{k(p-1)} \\ &= a^x a^{(p-1)^k}. \end{aligned}$$

By Theorem 2.1, $a^{p-1} = 1$ since p does not divide a. Thus, we have $a^y = a^x$. □

Using Theorem 2.1 and Corollary 2.2, we may show why Massey-Omura works.

Proposition 2.3. *Under the protocol in Massey-Omura described above, by computing c^{d_B}, Bob will always obtain the original message m.*

Proof. We will show that Bob's final computation of c^{d_B} will allow him to uncover m. The scheme requires Alice and Bob to choose their encryption keys such that $e_A d_A \equiv 1 \bmod p-1$. By Corollary 2.2, this means that for any $a \in \mathbb{Z}$, $a^x = a^y$ when working in $\mathbb{Z}_p$. Using substitution and carrying out all calculations mod p, we know

$$\begin{aligned} c^{d_B} &= (m^{e_A e_B d_A})^{d_B} \\ &= m^{e_A e_B d_A d_B}. \end{aligned}$$

We know that our exponents commute, thus we have

$$\begin{aligned} c^{d_B} &= (m^{e_A d_A})^{e_B d_B} \\ &= m^{e_B d_B}. \end{aligned}$$

Finally, since Alice's encryption and decryption keys are inverses of one another, as are Bob's, we have $c^{d_B} = m$. □

This works because of Euler's Theorem in $\mathbb{Z}_p$ and the fact that integer exponents commute in any group, which allows both Alice and Bob's encryption and decryption keys to cancel and reveal the plaintext message m. Notice that Bob's final calculation is a decryption of Alice's ciphertext, $c = m^{e_A e_B d_A}$. Since the exponents commute, we know that $c = m^{e_A d_A e_B}$ in which Alice's encryption and decryption keys will cancel. Thus $c = m^{e_B}$ and upon Bob's final decryption he obtains the original message m.

Example 2.4. Suppose that Alice and Bob agree on the prime $p = 10067$, which they publicly announce. Of course, this means that they each will choose their private encryption and decryption keys in $\mathbb{Z}_{10066}$. Suppose they choose $e_A = 649$, $d_A = 8515$, $e_B = 465$, and $d_B = 4849$. We can confirm that the pairs e_A and d_A are inverses by computing $e_A d_A = (649)(8515) = 1 \bmod 10066$. Similarly, we can confirm that e_B and d_B are an inverse pair. Thus, Alice and Bob's private keys are valid in this modulus. Finally, suppose that the secret message that Alice wants to send to Bob is $m = 540$. Their communication will occur as follows.

Working in $\mathbb{Z}_{10067}$ for all calculations, Alice begins by sending $m^{e_A} = 540^{649} = 9674$ to Bob. Bob computes $(m^{e_A})^{e_B} = 9674^{465} = 7340$ and sends this back to Alice. Then, Alice computes her ciphertext $c = (m^{e_A e_B})^{d_A} = 7340^{8515} = 5501$ which is then sent to Bob. Finally, Bob decrypts by computing $c^{d_B} = 5501^{4849} = 540 = m$ and thus obtains Alice's original message.

We note that Alice and Bob never had to publicly announce any of their key data; both their encryption and decryption keys were kept private throughout communication.

2.2 Brute Force Attack on Classical Massey-Omura

When analyzing the effectiveness of a brute force attack on Massey-Omura, it is important to note how many key choices there are in the encryption system. Because the encryption and decryption keys e and d are found such that $ed \equiv 1 \bmod p-1$, e and d are units in $\mathbb{Z}_{p-1}$. Therefore, if we can find how many units there are in our modulus $p-1$ then we will know how many key choices Alice and Bob have. By well-known properties of the Euler φ-function, we know that the number of units in $\mathbb{Z}_n$ is equal to $\varphi(n)$. Since the encryption and decryption keys in Massey-Omura are found mod $p-1$, the number of units is $\varphi(p-1)$. Luckily, if we choose our prime p to be large, $p-1$ will also be large and so will be $\varphi(p-1)$. Let's show this with an example.

Example 2.5. Given the prime $p = 15487399$, we have $\varphi(p-1) = \varphi(15487398) = 5151312$. Here, $\frac{\varphi(p-1)}{p-1} = \frac{5151312}{15487398} \approx \frac{1}{3}$, so $\varphi(p-1)$ is still relatively large.

In general, a result of Hardy and Wright in [2] tells us that

$$\limsup_n \frac{\varphi(n)}{n} = 1.$$

Hence $\varphi(n)$ is roughly n in the long run. Thus, the effectiveness of a brute force attack depends on how large p is and how fast each key can be checked. In practice, we use a key space larger than $2^{80} \approx 10^{24}$ as standard 80-bit security.

There are some significant pros and cons to Massey-Omura. As previously mentioned, this public-key-like system allows anyone to participate while still keeping everyone's encryption *and* decryption keys private. Additonally, any new user can be added to the system without causing current users to create new keys. However, it is also important to note that if Eve discovers *either* an encryption or decryption key, she can find the other with no problem. By the nature of the system, the encryption and decryption keys are units in the modulus $p-1$ and are inverses of one another; thus, using a modular calculator or the Euclidean algorithm, Eve can break the system just by finding either one of the keys.

2.3 Message Attacks on Classical Massey-Omura

Now we will examine Massey-Omura's susceptibility to message attacks. In a *known plaintext attack*, Eve can see a plaintext message and its resulting ciphertext. Although it may seem that knowing the type of encryption system, the plaintext message, and the ciphertext message may give away all of the secrets of the system, this type of attack results in a famously hard problem called the *Discrete Logarithm Problem* (DLP). In the transmission, Eve will be able to see the secret message m, the ciphertext c, and will know the public parameter p. So, in all, when Eve sees c, she knows that $c = m^{e_A e_B d_A} = m^{e_B} \bmod p$. However, the missing information that Eve does not know is the exponent e_B. Extracting this exponent is known as the DLP and is currently known to be very difficult to solve. Therefore, Massey-Omura is not susceptible to a known plaintext attack. In this system, a known plaintext attack is identical to a *chosen plaintext attack*, in which Eve chooses her own plaintext message to send in order to find out more about the system. These two attacks are the same because in either Eve will know m, c, and p but will not know the exponent. Thus, both attacks result in the DLP. If p is huge, solving the DLP will be infeasible over $\mathbb{Z}_p$, in which case Massey-Omura is not vulnerable to either type of attack.

In a *known ciphertext attack*, Eve will simply see the ciphertext that is sent from Alice to Bob. This is by far the weakest message attack that Eve can implement on any encryption system because she knows the least amount of information, only the ciphertext c. In Massey-Omura, Eve

will intercept $c = m^{e_B}$, but unlike the two previous attacks, she will have two missing pieces of information, m and e_B. This problem is potentially harder than the DLP and thus Massey-Omura is not susceptible to a known ciphertext attack.

Lastly, for a *chosen ciphertext attack*, Eve will choose the ciphertext message that is created and then decrypted. By the nature of this system, c is always the message raised to the other person's encryption key. Thus, whether Eve, Alice, or Bob begins the communication, Eve cannot truly *choose* the ciphertext message that is created. However, Eve can implement the following message attack, assuming their target is naive, which is still referred to as a chosen ciphertext attack.

Chosen Ciphertext Attack on Classical MO

1. Eve creates a private number r such that it is a unit mod p.
2. Eve begins communicating with Bob and sends r disguised as m^{e_E} to Bob.
3. Bob sends back r^{e_B} as normal.
4. Eve stops communicating.
5. Eve intercepts a ciphertext message $c = m^{e_B}$ that was exchanged from Alice to Bob as normal.
6. Eve asks Bob to begin communicating with her and he sends an encrypted message.
7. Eve discards the message that she just received and instead sends $r^{e_B}c$.
8. Bob computes $(r^{e_B}c)^{d_B}$ as usual and sends the result to Eve.
9. Eve multiplies this message by r^{-1} and obtains Alice's original message m.

Next we show why this attack works, again assuming the naivety of Eve's target.

Proposition 2.6. *When Eve multiplies $(r^{e_B}c)^{d_B}$ by r^{-1}, she will obtain the original message m.*

Proof. In his last step of communication, Bob computes $(r^{e_B}c)^{d_B}$. By substitution, we know this is equal to $(r^{e_B}m^{e_B})^{d_B}$. Then by distributing the exponent, the final result that he sends to Eve is actually rm, but Bob does not know this. Lastly, Eve multiplies this result by r^{-1} and will obtain m. □

Of course, this message attack will only work if Bob is naive enough to begin communicating with Eve after she asks. However, if he is so naive, then Eve will be able to use this attack to uncover any messages that Alice sends to Bob. We will note that this attack does *not* give away either e_B or d_B to Eve; she only obtains a single message. This means that Eve has not truly broken the system by finding any of Alice or Bob's private keys.

3 Massey-Omura with Cycles and Permutations

For traditional Massey-Omura, we were operating in the group of units $\mathbb{Z}_p^*$ under multiplication. This is a very well-understood group, which we would like to change in order to make the system even harder to break. In original Massey-Omura, we specify the group $\mathbb{Z}_p^*$ and then have a protocol. Our goal is to generalize this system by replacing $\mathbb{Z}_p^*$ by a non-abelian group and introducing

conjugations. We will now examine the possibility of using non-abelian groups in a Massey-Omura-style encryption system in order to potentially increase its security.

The main idea is that we will use conjugations in groups in place of exponents in our normal Massey-Omura scheme. As Proposition 3.2 will tell us, conjugations have the same properties as exponents, which make them optimal for creating a Massey-Omura-like system with groups. Let us start with a group theory review.

3.1 Group Theory Background

In order to make our new algebraic structures clear, we will first define conjugations and review some basic properties of group theory.

Definition 3.1. Let G be a group and suppose $g, x \in G$. The **conjugate** of x by g is the element $g^{-1}xg$. We will denote this element by x^g.

Proposition 3.2. *Let G be a group. For any $x, g, h \in G$, we have the following:*

1. $(x^g)^h = x^{gh}$
2. $x^e = x$
3. $x^g y^g = (xy)^g$
4. $x^g = y^g$ *implies* $x = y$
5. $(x^g)^{-1} = (x^{-1})^g$.

Proof. Starting with Property 1, we know that

$$\begin{aligned}(x^g)^h &= h^{-1}(x^g)h \\ &= h^{-1}(g^{-1}xg)h \\ &= (h^{-1}g^{-1})x(gh) \\ &= (gh)^{-1}x(gh) \\ &= x^{gh}.\end{aligned}$$

For Property 3, we have

$$\begin{aligned}x^g y^g &= (g^{-1}xg)(g^{-1}yg) \\ &= g^{-1}x(gg^{-1})yg \\ &= g^{-1}xyg \\ &= (xy)^g.\end{aligned}$$

Proofs of all other properties follow by similar, straightforward calculations. □

Definition 3.3. The **symmetric group** on n letters, denoted S_n, is the group of permutations of the set $\{1, 2, \ldots, n\}$ under function composition.

Definition 3.4. A subgroup of a symmetric group S_n is called a **permutation group**.

We will use juxtaposition as the notation for composition of permutations for simplicity of reading. It is important to note that permutation multiplication is usually not commutative. We will demonstrate this with the following example.

Example 3.5. Consider the following elements of S_4. Let

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix}$$

$$\tau = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}.$$

Then we have

$$\sigma\tau = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 2 & 3 \end{pmatrix},$$

yet

$$\tau\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 1 & 3 \end{pmatrix}.$$

We note that this non-commutativity is not unique to this example. In fact, S_n is non-abelian whenever $n \geq 3$.

Proposition 3.6. *Let G be a group with $a, b \in G$. Then, if a and b commute, then so do:*

- *a and b^{-1}*
- *a^{-1} and b*
- *a^{-1} and b^{-1}.*

Proof. To prove the first item, working in a group G with $a, b \in G$, suppose that a and b commute. We want to show that $ab^{-1} = b^{-1}a$. We have

$$\begin{aligned} ab^{-1} &= b^{-1}bab^{-1} \\ &= b^{-1}abb^{-1} \\ &= b^{-1}a. \end{aligned}$$

The others are proven similarly. □

3.2 Massey-Omura and Conjugation

We now present our new protocol using conjugation in S_n. Although any non-abelian group would work for this system, we choose to work with S_n as this group is a standard example of a non-abelian group. Suppose Alice and Bob have a way to create private double-blind commuting permutations. This means that Alice and Bob can fix a public parameter n and somehow create commuting permutations in S_n without ever making them public. Then, we can apply these conjugations to a non-commutative Massey-Omura scheme. Below we give a general protocol for implementing Massey-Omura encryption over S_n.

Massey-Omura in S_n

1. Let α and α^{-1} be Alice's encryption and decryption keys. Similarly, let β and β^{-1} be Bob's keys.
2. Assume that Alice and Bob can choose double-blind commuting permutations.
3. Alice has a message $m \in S_n$ structured as a permutation that she wishes to send to Bob. She first computes the conjugation $m^\alpha = \alpha^{-1} m \alpha$ and sends this to Bob.
4. Now, Bob sends $(m^\alpha)^\beta = m^{\alpha\beta}$ back to Alice.
5. Next, Alice would send back $c = (m^{\alpha\beta})^{\alpha^{-1}} = m^{\alpha\beta\alpha^{-1}}$.
6. Finally, Bob decrypts by computing $c^{\beta^{-1}} = (m^{\alpha\beta\alpha^{-1}})^{\beta^{-1}} = m^{\alpha\beta\alpha^{-1}\beta^{-1}}$.

Proposition 3.7. *When Bob decrypts the ciphertext c, he will obtain Alice's original message m.*

Proof. We want to show that $c^{\beta^{-1}} = m$. First, we must show that $c = m^\beta$ so that when Bob decrypts, he will obtain m. Using Proposition 3.2.1, we have

$$\begin{aligned} c &= (m^{\alpha\beta})^{\alpha^{-1}} \\ &= m^{\alpha\beta\alpha^{-1}}. \end{aligned}$$

Because α and β were chosen such that they commute, by Proposition 3.6 we know that β and α^{-1} also commute. Hence we have

$$\begin{aligned} c &= m^{\alpha\alpha^{-1}\beta} \\ &= m^\beta. \end{aligned}$$

Next, when Bob decrypts by his β^{-1}, we have

$$\begin{aligned} c^{\beta^{-1}} &= (m^\beta)^{\beta^{-1}} \\ &= m^{\beta\beta^{-1}} \\ &= m. \end{aligned}$$

Thus, Bob has successfully obtained Alice's original message m. □

This brings us to the primary focus of this thesis.

Main question: How can Alice and Bob blindly choose α and β so that they will always commute?

In traditional Massey-Omura, Bob's last step works because of the commutative property of exponents, allowing his encryption and decryption keys to cancel, leaving only the message behind. However, we cannot always guarantee that the factors in our previous conjugation, α and β, commute because S_n is a non-abelian group. Thus, Alice's exponents will not always cancel and our modified Massey-Omura system will not work. We will now explore the possibility of using commuting cycles to fix this problem.

3.3 Cycle Notation

We will now begin to use standard cycle notation for permutations; this is a much easier and more efficient representation for manipulating permutations. As an example, $\left(\begin{smallmatrix}1&2&3&4&5\\1&5&2&4&3\end{smallmatrix}\right)$ can be represented as a cycle of length 3, written (253). For an example of a product of two cycles, $(253)(145) = (14325)$. However, note that in this example, $(145)(253) = (14532) \neq (14325)$.

Definition 3.8. Two cycles in S_n, $\sigma = (a_1\ a_2\ \cdots\ a_k)$ and $\tau = (b_1\ b_2\ \cdots\ b_l)$, are **disjoint** if $a_i \neq b_j$ for all i and j.

For example, $\sigma = (135)$ and $\tau = (246)$ are disjoint in S_6. The following well-known proposition will be key in obtaining our goal of using permutation groups and conjugation in Massey-Omura.

Proposition 3.9. *Let σ and τ be two disjoint cycles in S_n. Then $\sigma\tau = \tau\sigma$.*

Thus, if we can create two disjoint cycles, they will always commute and we can use them in our new Massey-Omura scheme.

3.4 Massey-Omura with Disjoint Cycles

Now we will begin to use conjugation with disjoint cycles to finally create a new Massey-Omura-like system. In our system, Alice and Bob will agree on a public symmetric group S_n, where n is even, from which they will choose their disjoint cycles. The set $\{1, 2, \ldots, n\}$ is partitioned into two equal halves: $A = \{1, 2, \ldots, \frac{n}{2}\}$ for Alice and $B = \{\frac{n}{2} + 1, \ldots, n\}$ for Bob. Alice's job is to create a cycle by moving only her half, A, which means she will fix everything in Bob's half, B. Likewise, Bob will create a cycle by moving only his half, B, while fixing everything in Alice's half, A. These cycles are Alice and Bob's private encryption keys (α and β), and their respective inverses will be their private decryption keys (α^{-1} and β^{-1}). It is clear that α and β will always be disjoint cycles because Alice and Bob chose them by permuting only their own halves of $\{1, 2, \ldots, n\}$. Finally, we can introduce Massey-Omura with Disjoint Cycles. We will need the following definition for notation purposes.

Definition 3.10. Let $\sigma \in S_n$ be a permutation. We define the **moved-set** $M(\sigma)$ to be

$$M(\sigma) = \{x \in \{1, 2, \ldots, n\} \mid \sigma(x) \neq x\}.$$

Similarly, we define the **fixed-set** $F(\sigma)$ to be

$$F(\sigma) = \{x \in \{1, 2, \ldots, n\} \mid \sigma(x) = x\}.$$

Below is a description of how to implement our version of Massey-Omura encryption over S_n by using disjoint cycles.

Massey-Omura with Disjoint Cycles (MODC)

1. Alice and Bob agree on a public symmetric group S_n.
2. Alice creates a cycle α with $M(\alpha) \subseteq \{1, 2, \ldots, \frac{n}{2}\}$ as defined above. Similarly, Bob creates a cycle β with $M(\beta) \subseteq \{\frac{n}{2}+1, \ldots, n-1, n\}$.
3. Alice has a message $m \in S_n$ structured as a permutation that she wishes to send to Bob. She first computes the conjugation m^α and sends this to Bob.
4. Bob takes this message and computes $(m^\alpha)^\beta$ and sends it to Alice.
5. Alice creates the ciphertext c by computing $(m^{\alpha\beta})^{\alpha^{-1}} = m^\beta$ and sends it to Bob.
6. Bob finally decrypts by computing $(m^\beta)^{\beta^{-1}}$ and obtains the original message m.

Example 3.11. Suppose Alice and Bob agree to work in S_6. As a disclaimer, S_6 is too small for practical purposes, but we will still use it as an example. Then, Alice creates her own cycle by permuting only the first $\frac{6}{2} = 3$ numbers. Let Alice's encryption and decryption keys be

$$\alpha = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 1 & 4 & 5 & 6 \end{pmatrix} = (123) \text{ and } \alpha^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 1 & 2 & 4 & 5 & 6 \end{pmatrix} = (132).$$

Likewise, Bob creates his own cycle by permuting only the last 3 numbers. Let Bob's encryption and decryption keys be

$$\beta = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 6 & 4 & 5 \end{pmatrix} = (465) \text{ and } \beta^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 5 & 6 & 4 \end{pmatrix} = (456).$$

Finally, suppose Alice wants to send Bob the following message:

$$m = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 5 & 1 & 6 & 4 & 3 \end{pmatrix} = (125463).$$

First Alice sends $m^\alpha = (132)(125463)(123) = (154623)$ to Bob. Bob then sends $m^{\alpha\beta} = (456)(154623)(465) = (165423)$ back. Next, Alice sends $(m^{\alpha\beta})^{\alpha^{-1}} = m^\beta = (123)(165423)(132) = (126543)$. Finally, Bob decrypts by computing $(m^\beta)^{\beta^{-1}} = \beta(m^\beta)\beta^{-1} = (465)(126543)(456) = (125463) = m$.

Bob obtained the original message m in the end, just as he should. Note that this message was created as a cycle, but it does not have to be. Alice's m can be any permutation in whichever group S_n Alice and Bob agree upon.

Now that we have established our new Massey-Omura system with disjoint cycles, we can analyze the key count as a first measure of security. To find a formula, we want to count the number of possible cycles of length k. There are $k!$ ways to arrange k objects. We know that the way cycles are written is not unique; there are actually k ways to write each cycle. So, we find the number of *unique* k-cycles is $\frac{k!}{k} = (k-1)!$. In our system, though, Alice and Bob both create cycles of maximal length $k = \frac{n}{2}$, so the exact number of keys they each have is

$$\left(\frac{n}{2} - 1\right)!.$$

Using this formula, we can determine how big our keyspace would need to be in order to thwart a brute force attack. The three most standard measures, in practice, for analyzing security against

brute force are 80-bit, 128-bit, and 256-bit security. For example, 80-bit security requires that we have at least $2^{80} \approx 1.2 \times 10^{24}$ keys in our keyspace. In order to make brute force infeasible, we need to find which n value in our formula would make the size of the keyspace greater than 2^{80}. By using the above formula, we find that 52 is the smallest n such that $\left(\frac{n}{2} - 1\right)! > 2^{80}$. So, S_{52} gives the smallest keyspace that guarantees 80-bit security. Similarly, we find that S_{72} and S_{118} guarantee 128-bit and 256-bit security, respectively.

It is important to note that it is logical to create two equal halves from which Alice and Bob choose permutations. If Alice, for example, had a larger set than Bob to choose from, Bob's keys would inherently be more vulnerable since they came from a smaller keyspace. Thus, for creating a system for two parties to use, splitting S_n into two *equal* halves is optimal.

3.5 Massey-Omura with Disjoint Permutations

We will now extend this new system by creating another variation using disjoint permutations in place of cycles. Let us start with the following definition and proposition.

Definition 3.12. Permutations $\sigma, \tau \in S_n$ are **disjoint** if $M(\sigma) \cap M(\tau) = \emptyset$.

Proposition 3.13. *Let σ and τ be two disjoint permutations in S_n. Then $\sigma\tau = \tau\sigma$.*

Proof. Let σ and τ be disjoint permutations in S_n. We want to show that $\sigma(\tau(x)) = \tau(\sigma(x))$ for all $x \in \{1, 2, \ldots, n\}$. There are two cases in which we must prove this to be true.
Case 1: x is fixed by both σ and τ; $x \in F(\sigma)$ and $x \in F(\tau)$.

This means that $\sigma(x) = x$ and $\tau(x) = x$. In this case, we have

$$\sigma(\tau(x)) = \sigma(x) = x.$$

We also have

$$\tau(\sigma(x)) = \tau(x) = x.$$

Hence, $\sigma(\tau(x)) = \tau(\sigma(x))$.
Case 2: x is fixed by only one of σ or τ.

Suppose first that $x \in M(\sigma)$ and $x \in F(\tau)$. Then, $\sigma(\tau(x)) = \sigma(x)$. We can also show that $\tau(\sigma(x)) = \sigma(x)$. To prove this, we claim that $\sigma(x) \in F(\tau)$. Suppose instead $\sigma(x) \in M(\tau)$. Then $\sigma(x) \in F(\sigma)$. This implies that $\sigma(\sigma(x)) = \sigma(x)$. As σ is one-to-one, $\sigma(x) = x$ and hence, $x \in F(\sigma)$. However, this is a contradiction to our assumption that $x \in M(\sigma)$. Hence, $\sigma(x) \in F(\tau)$ and thus $\tau(\sigma(x)) = \sigma(x)$. The proof for when $x \in M(\tau)$ and $x \in F(\sigma)$ follows similarly. In all cases, it is true that $\sigma\tau = \tau\sigma$. □

Hence, instead of using cycles, we can also use any disjoint permutations since they will also commute. Now we will introduce Massey-Omura with Disjoint Permutations.

Similar to our MODC system, Alice and Bob will again agree on a public symmetric group S_n, where n must be even, from which they will choose their disjoint permutations. Again, we create two equal partitions by halving $\{1, 2, \ldots, n\}$: $A = \{1, 2, \ldots, \frac{n}{2}\}$ for Alice and $B = \{\frac{n}{2} + 1, \ldots, n\}$ for Bob. Alice will create *any* permutation by moving only elements in her half, A, which means she will fix everything in Bob's half, B. Similarly, Bob will create his permutation by only moving elements in his half, B, while fixing all elements in Alice's half, A. These are Alice and Bob's private encryption keys (α and β), and their respective inverses will be their private decryption keys (α^{-1} and β^{-1}). Just as before, α and β will always be disjoint permutations because Alice and Bob only permuted their own halves, A and B. Hence, they will commute.

Massey-Omura with Disjoint Permutations (MODP)

1. Alice and Bob agree on a public symmetric group S_n.
2. Alice chooses a permutation α with $M(\alpha) \subseteq \{1, 2, \ldots, \frac{n}{2}\}$. Similarly, Bob chooses a permutation β with $M(\beta) \subseteq \{\frac{n}{2}+1, \ldots, n-1, n\}$.
3. Alice has a message $m \in S_n$ structured as a permutation that she wishes to send to Bob. She first computes the conjugation m^α and sends this to Bob.
4. Bob takes this message and computes $(m^\alpha)^\beta$ and sends it to Alice.
5. Alice creates the ciphertext c by computing $(m^{\alpha\beta})^{\alpha^{-1}} = m^\beta$ and sends it to Bob.
6. Bob finally decrypts by computing $c^{\beta^{-1}}$ and obtains the original message m.

As shown in the proof of Proposition 3.7, Bob will obtain m by decrypting the ciphertext c because α and β were again chosen such that they commute.

Example 3.14. Suppose Alice and Bob work in S_8. Then, Alice creates a permutation by moving only the first $\frac{8}{2} = 4$ entries of $\{1, 2, \ldots, 8\}$. Let Alice's encryption and decryption keys be

$$\alpha = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 4 & 1 & 2 & 5 & 6 & 7 & 8 \end{pmatrix} \text{ and } \alpha^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 4 & 1 & 2 & 5 & 6 & 7 & 8 \end{pmatrix}.$$

Similarly, Bob creates his own permutation by moving only the last 4 numbers. Let Bob's encryption and decryption keys be

$$\beta = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 8 & 7 & 6 & 5 \end{pmatrix} \text{ and } \beta^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 8 & 7 & 6 & 5 \end{pmatrix}.$$

Finally, suppose Alice wants to send Bob the following message:

$$m = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 5 & 4 & 3 & 1 & 6 & 7 & 2 & 8 \end{pmatrix}.$$

First Alice sends

$$m^\alpha = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 3 & 5 & 2 & 6 & 7 & 4 & 8 \end{pmatrix}$$

to Bob. Next, Bob sends

$$m^{\alpha\beta} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 3 & 8 & 2 & 5 & 4 & 6 & 7 \end{pmatrix}$$

back to Alice. Then, Alice responds by sending

$$m^{\alpha\beta\alpha^{-1}} = m^\beta = c = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 8 & 4 & 3 & 1 & 5 & 2 & 6 & 7 \end{pmatrix}$$

back to Bob. Lastly, Bob computes

$$c^{\beta^{-1}} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 5 & 4 & 3 & 1 & 6 & 7 & 2 & 8 \end{pmatrix}$$

and obtains the original message m.

Now that we have established Massey-Omura with disjoint permutations, we can analyze its security as we did with our previous modified system with disjoint cycles. First we will look at the key count. Since Alice is permitted to create any permutation from her half of the $\frac{n}{2}$ elements, the key count for this system can be found with

$$\left(\frac{n}{2}\right)!.$$

Using this formula, we can determine how big our n would need to be in order to thwart a brute force attack. For an example, to find the smallest n to guarantee 80-bit security, we find the smallest n such that $\frac{n}{2}! > 2^{80} \approx 1.2 \times 10^{24}$. We find that 50 would be the minimal permutation length that is necessary, while for 128-bit security, we would need to work in S_{70}.

3.6 Comparing MODC and MODP

First, we will note that it is difficult to compare the security of our modified systems with that of the original Massey-Omura encryption system. Original MO has a key count based on the Euler φ-function, while the key counts of MODC and MODP are based on the lengths of cycles and permutations in an agreed upon n for S_n. Since these are two different mathematical environments, it is not easy to compare them directly. However, the fact that MODC and MODP's key counts grow by a factorial gives a very fast-growing count, which is good for high security.

When comparing the security of MODC to MODP, we will first examine their minimum key lengths for 80- and 128-bit security. For 80-bit security, MODC requires a minimum cycle of length 52 while MODP only requires a permutation length of 50. Similarly, for 128-bit security, MODC requires a minimum cycle length of 72 as compared to the minimum permutation length of 70 in MODP. Although in both cases this is only a difference of length 2, it is still a slight difference that is important to note.

The message attacks for MODC and MODP are parallel to those in the original Massey-Omura system. In the original protocol, security is based on the well-known Discrete Logarithm Problem (DLP). That is, if at any point Eve knows both the message m and its encrypted output, say m^{e_A}, finding e_A is not feasible. Our two systems are similarly not vulnerable to a known or chosen plaintext attack. In our new systems, the algebra is no longer based on exponentiation, but rather conjugation. In MODP, for example, the missing piece of information is not an exponent, but rather a permutation and that permutation's inverse. Here, if Eve knows m and a resulting encryption of $\alpha^{-1}m\alpha$, finding the value of α is known as the *conjugacy search problem* (CSP). Presented in [4], this problem is the conjugacy equivalent of the DLP and is hence thought to be a substantially difficult problem. Next, in a known ciphertext attack, Eve would only know one piece of information in the last pass, namely c. Again, similar to the original system, this attack would result in a problem that is potentially harder than CSP. So, MODC and MODP are not vulnerable to a known ciphertext attack.

Lastly, MODC and MODP cannot really have a true chosen ciphertext attack because c is always the message conjugated by the other communicator's encryption key, just as in the original Massey-Omura system. However, the same variation as before on a chosen ciphertext attack can be implemented. We leave the description of this attack as an exercise for the reader. Similar to our original system, if Eve discovers either the encryption or decryption key in MODC or MODP (the cycle/permutation and its inverse, respectively), she can find the other with no problem.

As previously mentioned in Section 2.2, original Massey-Omura is a public-key-like system that allows anyone to participate while also keeping everyone's keys private. This is a significant advantage to the original system because it allows for as many communicators to join as needed.

On the other hand, MODC and MODP only allow for two people to communicate because of the way the private keys are chosen. Since Alice and Bob *must* agree on an S_n and *must* create their keys by splitting the cycle or permutation in two parts, only two people can use this agreed upon system. So, for every unique pair of communicators, a new S_n would need to be decided on and thus each person would need to create two new keys.

The only way that Alice and Bob could invite another communicator, Carl, to their system is if they create their keys by dividing the set $\{1, 2, 3, \ldots, n\}$ into three parts and agreeing to only permute their assigned third. However, this would affect the minimum size of n they can choose to thwart a brute force attack. For three communicators, to reach 80-bit security, they would need to work in S_{75} as opposed to S_{50}. Similarly, to reach 128-bit security, they would need to work in S_{105} as opposed to S_{70}. This is a 25-bit and 35-bit length difference for just adding one more communicator, so we can see that adding many more parties would require a significantly larger choice of S_n. We will now further examine this same idea, incorporating k parties into MODP.

3.7 MODP with Multiple Parties

We can further modify MODP to create a system in which k parties can use the system. Here we take the set $\{1, 2, 3, \ldots, n\}$, divide it into k blocks of equal size (assuming n is a multiple of k), and allow participants to choose permutations from their respective block. Taking the formula for the original key count for MODP, we can calculate the key count for this system with k parties with

$$\left(\frac{n}{k}\right)!.$$

Looking at five examples of k values, we can see how the n necessary to reach a certain key count increases as the number of parties increases. The table below gives the minimal n in the symmetric group S_n required to reach standard bit level security for k parties.

k	80-bit	128-bit	256-bit
2	50	70	116
3	75	105	174
4	100	140	232
6	150	210	348
8	200	280	464

As we see in the table above, the growth of each n is linear as k increases. To see why that is, consider the following example of the inequality $m! > 2^{80}$. Here, we are finding the minimal m such that this inequality is true so that we can reach standard 80-bit security. Using a computer search we find that this minimum value of m is 25. Since we are working with k parties, we know that $m = \frac{n}{k}$. Knowing this, we have

$$\begin{aligned} \frac{n}{k} &= 25 \\ n &= 25k. \end{aligned}$$

Thus, at each k for 80-bit security, we see that the necessary n increases by 25 each time. We can find similar equations for 128- and 256-bit security as well.

In practice, if we know our machine can only handle an n equal to 500, we can follow a similar procedure to find how many parties we can allow to participate in communication. Suppose we

want 128-bit security, where each increase of k by an increment of 1 increases n by 35. Then we have

$$500 = 35k.$$

Solving for k, we obtain $k \approx 14.3$, meaning that we can allow 14 people to communicate in this system considering our limitations of a 500-length permutation and 128-bit security.

4 Massey-Omura with Maximal Abelian Subgroups

We will now use maximal abelian subgroups of S_n in place of disjoint cycles and permutations in our modified version of Massey-Omura. As mentioned earlier, our first two modifications only allow for a limited number of people to participate in communication. Although we additionally created MODP with k parties, we saw that the size of n grew quickly with each additional user. Massey-Omura with Abelian Subgroups (MOAS) will be independent of the number of allowed parties and will allow any number of participants to communicate. This is optimal in an encryption system because it does not require communicators to create separate keys each time they wish to communicate with a different user.

In MOAS, Alice and Bob will agree on any public group S_n from which they will find an abelian subgroup A, which they will also publish. Alice will choose an element $\alpha \in A$ while Bob chooses a $\beta \in A$, which are their private encryption keys. It follows that they both also have α^{-1} and β^{-1}, their private decryption keys. MOAS proceeds in the following way.

Massey-Omura with Maximal Abelian Subgroups (MOAS)

1. Alice and Bob agree on a symmetric group S_n with a public abelian subgroup A.
2. Alice chooses an element $\alpha \in A$ while Bob chooses $\beta \in A$.
3. Alice has a message $m \in S_n$ that she wishes to send to Bob. She first computes the conjugation m^α and sends this to Bob.
4. Bob takes this message and computes $(m^\alpha)^\beta$ and sends it to Alice.
5. Alice creates the ciphertext c by computing $(m^{\alpha\beta})^{\alpha^{-1}} = m^\beta$ and sends it to Bob.
6. Bob finally decrypts by computing $(m^\beta)^{\beta^{-1}}$ and obtains the original message m.

Because α and β are elements of the abelian subgroup A, they will always commute. Thus, as proved in Proposition 3.7, Bob will obtain m by decrypting the ciphertext c.

Now that we have established MOAS, we can analyze its security as we did with our previous two modified systems. First we will examine the key count of this new system. We want to find the size of the keyspace, which is precisely the size of the subgroup A.

We want our keyspace to be as large as possible for security reasons, so we need to find an abelian subgroup A of S_n of *maximal order* to choose for our keyspace. As shown by Burns and Goldsmith's result in [1], we have the following theorem. This theorem gives the maximal order of abelian subgroups of the symmetric group. This result will give us three cases in which the size of the keyspace depends on the congruence class of n mod 3. We note that this theorem gives the size and anatomy of the subgroups that we will implement in MOAS, but we also need to find examples for actual implementation of the system. This theorem gives the maximal abelian subgroups of S_n up to isomorphism type; it does not give concrete examples of who these groups are. We will give several examples separately.

Theorem 4.1 (Burns-Goldsmith [1])**.** *Let A be an abelian subgroup of maximal order of the symmetric group S_n. Then*

(i) $A \cong \mathbb{Z}_3^k$ *if* $n = 3k$,

(ii) $A \cong \mathbb{Z}_2 \times \mathbb{Z}_3^k$ *if* $n = 3k + 2$,

(iii) either $A \cong \mathbb{Z}_4 \times \mathbb{Z}_3^{k-1}$ *or* $A \cong \mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_3^{k-1}$ *if* $n = 3k + 1$.

We now have the following two important questions:

- What are these groups? How can we find examples for implementation?
- How can we calculate key counts?

Now, by finding examples of maximal abelian subgroups as specified by [1], we will answer our first important question.

Proposition 4.2. *Let G be a group with elements $a_1, a_2, \ldots, a_k$. Suppose that these elements commute, so that $a_i a_j = a_j a_i$ for all i and j. Then the subset*

$$H = \{a_1^{r_1} a_2^{r_2} \cdots a_k^{r_k} : r_i \in \mathbb{Z}\}$$

is an abelian subgroup of G.

Proof. To show that H is a subgroup of G, we will apply the Subgroup Test. We know that $a_1^0 a_2^0 \cdots a_k^0 = e \in H$ when all $r_i = 0 \in \mathbb{Z}$. Hence, the identity element from G is in H. Next we must show that for any elements $a, b \in H$, $ab \in H$. Let

$$a = a_1^{r_1} a_2^{r_2} \cdots a_k^{r_k} \text{ and } b = a_1^{q_1} a_2^{q_2} \cdots a_k^{q_k}.$$

Then

$$ab = (a_1^{r_1} a_2^{r_2} \cdots a_k^{r_k})(a_1^{q_1} a_2^{q_2} \cdots a_k^{q_k}).$$

Since $a_i a_j = a_j a_i$ for all i, j, we have

$$\begin{aligned} ab &= (a_1^{r_1} a_1^{q_1})(a_2^{r_2} a_2^{q_2}) \cdots (a_k^{r_k} a_k^{q_k}) \\ &= a_1^{r_1+q_1} a_2^{r_2+q_2} \cdots a_k^{r_k+q_k} \in H. \end{aligned}$$

Hence the operation is closed. Lastly, we will show that each element $a \in H$ has an inverse within H. If $a = a_1^{r_1} a_2^{r_2} \cdots a_k^{r_k}$ then $a^{-1} = a_1^{-r_1} a_2^{-r_2} \cdots a_k^{-r_k}$, and so each element $a \in H$ has an inverse within H. Because all three conditions in the Subgroup Test were satisfied, H is a subgroup of G. Since all of the elements a_i and a_j commute in G, H is automatically abelian. Thus, H is an abelian subgroup of G. □

Now that we have created an abelian subgroup H of G, in order to correctly find $|H|$, we will need to have the following definition.

Definition 4.3. Let $S = \{a_1, a_2, \ldots, a_k\}$ be a finite set of elements in an abelian group H. We say that the set S is **independent** in H if whenever a power of a_i is expressed in terms of the other elements a_j $(j \neq i)$ as

$$a_i^s = a_1^{r_1} \cdots a_{i-1}^{r_{i-1}} a_{i+1}^{r_{i+1}} \cdots a_k^{r_k}$$

we have $a_i^s = e$.

Lemma 4.4. *Suppose that H is an abelian group and that $S = \{a_1, a_2, \ldots, a_k\}$ is a finite subset of H. Suppose that each element $a_i \in S$ has finite order m_i. If S is an independent set in H, then $a_1^{r_1} a_2^{r_2} \cdots a_k^{r_k} = e$ implies that $r_i \equiv 0 \bmod m_i$ for all i.*

Proof. Let $S = \{a_1, a_2, \ldots, a_k\}$ be a finite set of elements in an abelian group H and suppose S is independent. Consider the equation

$$e = a_1^{r_1} \cdots a_{i-1}^{r_{i-1}} a_i^{r_i} a_{i+1}^{r_{i+1}} \cdots a_k^{r_k}.$$

Solving for $a_i^{r_i}$, we have

$$a_i^{r_i} = a_1^{-r_1} \cdots a_{i-1}^{-r_{i-1}} a_{i+1}^{-r_{i+1}} \cdots a_k^{-r_k}.$$

Since S is independent, this means that $a_i^{r_i} = e$. By Lagrange, we have $r_i \equiv 0 \bmod m_i$. □

Now we can finally state our theorem.

Theorem 4.5. *Let $S = \{a_1, a_2, \ldots, a_k\}$ be a finite set of elements in the abelian group $H = \{a_1^{r_1} a_2^{r_2} \cdots a_k^{r_k} : r_i \in \mathbb{Z}\}$. If $S = \{a_1, a_2, \ldots, a_k\}$ is an independent set in H and $m_i = \operatorname{ord}(a_i)$ is finite, then*

$$|H| = m_1 m_2 \cdots m_k.$$

Proof. Let $S = \{a_1, a_2, \ldots, a_k\}$ be an independent set in the abelian group $H = \{a_1^{r_1} a_2^{r_2} \cdots a_k^{r_k} : r_i \in \mathbb{Z}\}$. Also let $m_i = \operatorname{ord}(a_i)$ be finite. We may restrict $r_i, s_i < m_i$. Suppose

$$a_1^{r_1} a_2^{r_2} \cdots a_k^{r_k} = a_1^{s_1} a_2^{s_2} \cdots a_k^{s_k}.$$

Then

$$a_1^{r_1 - s_1} = a_2^{s_2 - r_2} \cdots a_k^{s_k - r_k}.$$

Because S is independent, we know that $a^{r_1 - s_1} = e$ which implies that $r_1 - s_1 \equiv 0 \bmod m_1$. So, since we chose $r_1, s_1 < m_1$, we find $r_1 = s_1$.

Now, we can left cancel and obtain $a_2^{r_2} \cdots a_k^{r_k} = a_2^{s_2} \cdots a_k^{r_k}$. Following the same argument, we find that $r_i = s_i$ for each i. Then, since there are m_i ways to choose each r_i, we find $|H| = m_1 m_2 \cdots m_k$. □

The proof of the following is immediate.

Corollary 4.6. *In the situation above, there is an isomorphism*

$$H \cong \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \cdots \times \mathbb{Z}_{m_k}$$

given by

$$a_1^{r_1} a_2^{r_2} \cdots a_k^{r_k} \mapsto (r_1, r_2, \ldots, r_k).$$

We now present examples of how we find such maximal abelian subgroups of S_n as shown in [1].

Example 4.7. We will work in S_{12} for this example. Since $12 \equiv 0 \bmod 3$, we take the first case from Theorem 4.1. So, the maximal abelian subgroups of S_{12} are isomorphic to

$$\mathbb{Z}_3 \times \mathbb{Z}_3 \times \mathbb{Z}_3 \times \mathbb{Z}_3.$$

We will then divide $\{1, 2, \ldots, 12\}$ into four disjoint subsets of size three, like $\{1, 2, 3\}$, $\{4, 5, 6\}$, $\{7, 8, 9\}$, and $\{10, 11, 12\}$. From this partition, we can create disjoint permutations, for example:

$$\sigma_1 = (312),\ \sigma_2 = (645),\ \sigma_3 = (978),\ \sigma_4 = (11\ 12\ 10).$$

Let A be the subgroup generated by these four permutations. Since these cycles are disjoint, they clearly commute and are independent. By Corollary 4.6, it is immediate that $A \cong \mathbb{Z}_3^4$ and is therefore maximal by Theorem 4.1. Note that in this case, this abelian subgroup has order 81.

Now that we have shown an example of how to find examples of these subgroups, we will now show an example of communication between Alice and Bob in S_{12}.

Example 4.8. Using the same cycles as in Example 4.7, let $\alpha = \sigma_1^2\sigma_3\sigma_4$ and let $\beta = \sigma_2^2\sigma_3\sigma_4^2$. Let $m = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 1 & 4 & 7 & 2 & 5 & 6 & 10 & 12 & 8 & 3 & 9 & 11 \end{pmatrix}$. Communication between Alice and Bob proceeds as follows.

1. Alice sends $m^\alpha = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 9 & 2 & 4 & 3 & 5 & 6 & 11 & 7 & 12 & 8 & 10 & 1 \end{pmatrix}$ to Bob.
2. Bob sends $m^{\alpha\beta} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 8 & 2 & 5 & 4 & 3 & 6 & 9 & 10 & 12 & 1 & 7 & 11 \end{pmatrix}$ back to Alice.
3. Alice sends $m^{\alpha\beta\alpha^{-1}} = c = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 1 & 5 & 9 & 4 & 2 & 6 & 10 & 7 & 11 & 12 & 3 & 8 \end{pmatrix}$ back to Bob.
4. Finally Bob decrypts by computing $c^{\beta^{-1}} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 1 & 4 & 7 & 2 & 5 & 6 & 10 & 12 & 8 & 3 & 9 & 11 \end{pmatrix}$, which is m.

For clarity, we also present an example to show how maximal abelian subgroups are found in the second case of Theorem 4.1.

Example 4.9. We will work in S_{14} for this example. Since 14 can be represented as $3(4)+2$, we take the second case from Theorem 4.1. So, the maximal abelian subgroups of S_{14} are isomorphic to $\mathbb{Z}_2 \times \mathbb{Z}_3^4$. We can divide $\{1, 2, \ldots, 14\}$ into five disjoint subsets, like $\{1, 2\}, \{3, 4, 5\}, \{6, 7, 8\}, \{9, 10, 11\}$ and $\{12, 13, 14\}$. From these partitions, five disjoint permutations can created. For example, $\sigma_1 = (12), \sigma_2 = (354), \sigma_3 = (687), \sigma_4 = (11\ 9\ 10)$, and $\sigma_5 = (12\ 13\ 14)$. Let A be the subgroup generated by these five permutations. Since these cycles are disjoint, they clearly commute and are independent. By Corollary 4.6, it is immediate that $A \cong \mathbb{Z}_2 \times \mathbb{Z}_3^4$ and is therefore maximal by Theorem 4.6. Note that in this case, this abelian subgroup has order 162.

We can find the key counts for Alice and Bob in each of the three cases. For the first case, $A \cong \mathbb{Z}_3^k$, so we have k number of $\mathbb{Z}_3$'s. Hence, we find that there are $3^k = 3^{\frac{n}{3}}$ key choices. We similarly solve for the other two cases and obtain the following theorem.

Theorem 4.10. *When Alice and Bob choose their private keys from a maximal abelian subgroup of S_n, we obtain a valid encryption scheme over S_n using MOAS. The resulting system has the following key counts:*

- $3^{\frac{n}{3}}$ *key choices when* $n = 3k$
- $2 \cdot 3^{\frac{n-2}{3}}$ *key choices when* $n = 3k + 2$
- $4 \cdot 3^{\frac{n-4}{3}}$ *key choices when* $n = 3k + 1$.

All of these key counts are on the order of $\left(\sqrt[3]{3}\right)^n \approx 1.44^n$.

5 Summary Key Counts and Comparisons

A natural final question now presents itself. Of the two versions of our modified Massey-Omura system we have created, MODP and MOAS, which is optimal? By comparing the n necessary in the group S_n to reach each standard bit level security for both systems, we can start to answer this question. The table below gives the minimal n required to reach standard bit level security, comparing multiple parties in MODP versus MOAS.

	80-bit	128-bit	256-bit
2 parties	50	70	116
3 parties	75	105	174
4 parties	100	140	232
5 parties	125	175	290
6 parties	150	210	348
7 parties	175	245	406
8 parties	200	280	464
9 parties	225	315	522
Max ab subgp	152	243	485

By looking at this table, we see that in the case of communication between only 2 parties, using MODP is optimal because it requires a significantly lower n than that of MOAS. However, in order to reach 80-bit security for 7 parties, the necessary n is 175, while MOAS only requires $n = 152$ and allows for any number of participants. This gives way to the idea of where our two systems *break even* in relation to the number of communicators participating and the desired bit-level security. Similarly, the break even point for 128-bit security is also at 7 parties. The break even point for 256-bit security is at 9 parties, in contrast. Hence, the number of parties in a particular instance of communication is one way to determine which of the two systems is optimal. For instance, if a group of 9 people wish to communicate with 128-bit security, they should use the maximal abelian subgroup system, as they will work in S_{243} as opposed to S_{315}.

This research brings about several questions that have yet to be answered. Among them are:

- Can we determine exactly the complexity of the conjugacy search problem in S_n?
- When using disjoint permutations, do vulnerabilities arise because Alice's key fixes half of the elements? Does this reveal information to Eve about the plaintext?
- Can this actually be implemented? Are these systems computationally feasible?

Answering these questions would tell us more about the usability of these systems.

References

[1] J. M. Burns and B. Goldsmith, *Maximal order abelian subgroups of symmetric groups*, Bulletin of the London Mathematical Society **21** (1989), no. 1, 70.

[2] G.H. Hardy, E.M. Wright, R. Heath-Brown, A. Wiles, and J. Silverman, *An introduction to the theory of numbers*, Oxford Mathematics Series, Oxford University Press, 2008.

[3] J.L. Massey and J.K. Omura, *Method and apparatus for maintaining the privacy of digital messages conveyed by public transmission*, January 28 1986, US Patent 4,567,600.

[4] A. Myasnikov, V. Shpilrain, and A. Ushakov, *Group-based cryptography*, Advanced Courses in Mathematics CRM Barcelona, Birkhauser Verlag, 2008.