Spring 4-28-2016

The Ko-Lee Key Exchange Protocol with Generalized Dihedral Groups

Christopher Lloyd

THE KO-LEE KEY EXCHANGE PROTOCOL WITH GENERALIZED DIHEDRAL GROUPS

An honors paper submitted to the Department of Mathematics
of the University of Mary Washington
in partial fulfillment of the requirements for Departmental Honors

Christopher Lloyd
April 2016

By signing your name below, you affirm that this work is the complete and final version of your paper submitted in partial fulfillment of a degree from the University of Mary Washington. You affirm the University of Mary Washington honor pledge: "I hereby declare upon my word of honor that I have neither given nor received unauthorized help on this work."

Christopher James Robert Lloyd
(digital signature)

04/28/16

The Ko-Lee Key Exchange Protocol with Generalized Dihedral Groups

Christopher Lloyd

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

Fredericksburg, Virginia

April 2016

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

Date	Approved
_____	_____
	Randall D. Helmstutler, Ph.D. thesis advisor
_____	_____
	Janusz Konieczny, Ph.D. committee member
_____	_____
	Keith E. Mellinger, Ph.D. committee member

Acknowledgments

I would like to thank Dr. Randall D. Helmstutler for his exceptional guidance and attention to detail. Furthermore I greatly appreciate the work of my committee members, Dr. Janusz Konieczny and Dr. Keith E. Mellinger. Lastly, I would like to thank my friend Pengcheng Zhang for his input at various stages of this paper.

Contents

1 Introduction **1**
1.1 The Diffie-Hellman Key Exchange Protocol 1
1.2 The Ko-Lee Key Exchange Protocol 2
1.3 Selecting a Platform Group 3

2 Background **4**
2.1 Generalized Dihedral Groups 4
2.2 Strengths of $D(A)$ 6
2.3 Complexity Theory 7

3 Analysis of Ko-Lee **8**
3.1 Presentation of $D(A)$ 8
3.2 Normal Forms 11
3.3 Ko-Lee with $D(A)$ 13
3.4 Conditions for Ko-Lee 15
3.5 Conjugate Attack on Ko-Lee 18

References **22**

Abstract

Given an arbitrary abelian group A, one may form the generalized dihedral group $D(A)$. As $D(A)$ is usually non-abelian, this makes it a possible candidate for use with certain non-commutative key exchange protocols. Specifically, we examine the security of using $D(A)$ with the Ko-Lee key exchange protocol. An appropriate presentation for $D(A)$ is developed alongside methods for computing within the group in the context of the Ko-Lee protocol. Lastly we show that for such groups Ko-Lee is susceptible to a polynomial time attack.

1 Introduction

1.1 The Diffie-Hellman Key Exchange Protocol

Key exchange protocols facilitate the construction of a common key between two parties over an insecure channel. The Diffie-Hellman protocol was the first of such protocols, originally published in 1976 [1]. We will present an implementation of the Diffie-Hellman protocol that operates on the group of units $\mathbb{Z}_p^*$ of the field $\mathbb{Z}_p$ for a fixed prime p.

Diffie-Hellman Protocol:

1. Alice and Bob agree on some large prime p.
2. Alice and Bob publicly agree on some $g \in \mathbb{Z}_p^*$.
3. Alice secretly selects $a \in \mathbb{Z}$ and sends g^a to Bob.
4. Bob secretly selects $b \in \mathbb{Z}$ and sends g^b to Alice.
5. Alice computes $K_A = \left(g^b\right)^a = g^{ba}$.
6. Bob computes $K_B = (g^a)^b = g^{ab}$.
7. The resulting private key is $K = K_A = K_B$ as $ab = ba$.

Example 1.1. Suppose Alice and Bob agree on $p = 1049$ and $g = 876$. Next they each decide on secret elements $a = 123$ and $b = 500$. Working modulo 1049, Alice computes $g^a = 681$ while Bob computes $g^b = 551$. Now Bob computes $K_B = (g^a)^b = 681^{500} = 8$ and Alice computes $K_A = \left(g^b\right)^a = 551^{123} = 8$. Thus $K_A = K_B$ as desired.

We will now highlight the main underlying assumption of the Diffie-Hellman protocol. It is assumed that given g and g^a that it is computationally infeasible to compute the value of a. If an attacker could obtain a from g and g^a the following attack would be possible.

Diffie-Hellman Attack:

Suppose that given g and g^a an attacker could efficiently compute a. As g^b is public, the attacker would be able to obtain the secret key by computing

$$(g^b)^a = K_A.$$

The problem of computing the value of a given g and g^a is known as the *discrete log problem*. Determining the computational complexity of the discrete log problem is still an open question,

although it is assumed to be computationally infeasible. This is all to say that research in cryptography is very active and for good reason. For the interest of developing more secure protocols, non-commutative key exchange protocols have been proposed. The simplest of such protocols is the Ko-Lee protocol, which will be presented in the following section.

1.2 The Ko-Lee Key Exchange Protocol

The Diffie-Hellman protocol relies on modular exponentiation as the primary operation. The Ko-Lee Protocol, which was originally published in 2000, utilizes conjugation as its primary operation [3].

Convention. Let x and y be elements of the group G. We write x^y for the element $y^{-1}xy$.

This is known as the *conjugate* of x by y. Here we will summarize some of the basic properties of conjugation.

Proposition 1.2. *Let G be a group. For any elements x, y, and z in G, the following statements hold:*

1. $(z^x)^y = z^{xy}$
2. $(z^x)^{-1} = \left(z^{-1}\right)^x$
3. $(z^x)^y = (z^y)^x$ *whenever* $xy = yx$
4. $(zy)^x = z^x y^x$.

The above laws follow directly from the fact that conjugation defines an action of G on itself by automorphisms [2]. Using these conjugates, we may now give a description of the Ko-Lee key exchange protocol.

Ko-Lee Protocol:

1. Alice and Bob publicly agree on a non-abelian group G with an abelian subgroup C.
2. They publicly agree on some $w \in G$.
3. Alice secretly selects $a \in C$ and sends w^a to Bob.
4. Bob secretly selects $b \in C$ and sends w^b to Alice.
5. Alice computes $K_A = (w^b)^a = w^{ba}$.
6. Bob computes $K_B = (w^a)^b = w^{ab}$.
7. The resultant private key is $K = K_A = K_B$ as $a, b \in C$.

From the third item in Proposition 1.2 we see that if $ab = ba$, then $(w^b)^a = (w^a)^b$. Thus to ensure that $K_A = K_B$, we must demand that C is an abelian subgroup.

Example 1.3. For this example we choose the platform group to be the symmetric group S_4. We will express the elements of S_4 in cycle notation. We select the abelian subgroup C to be the cyclic subgroup

$$C = \{e, (1\ 2\ 3\ 4), (1\ 3)(2\ 4), (1\ 4\ 3\ 2)\}.$$

It may be checked that C is generated by $(1\ 2\ 3\ 4)$. Next the public element $w = (1\ 2)$ is chosen. Now Alice and Bob select their secret elements from C. Alice selects $a = (1\ 4\ 3\ 2)$ and Bob selects

$b = (1\ 3)(2\ 4)$. Next the conjugates w^a and w^b need to be computed. This requires the values of a^{-1} and b^{-1}. It may be checked that $a^{-1} = (1\ 2\ 3\ 4)$ and $b^{-1} = (1\ 3)(2\ 4)$. Now Alice computes

$$\begin{aligned} w^a &= (1\ 4\ 3\ 2)^{-1}(1\ 2)(1\ 4\ 3\ 2) \\ &= (1\ 2\ 3\ 4)(1\ 2)(1\ 4\ 3\ 2) \\ &= (1\ 3\ 4)(1\ 4\ 3\ 2) \\ &= (2\ 3) \end{aligned}$$

and sends it to Bob. Next Bob computes

$$\begin{aligned} w^b &= ((1\ 3)(2\ 4))^{-1}\,(1\ 2)(1\ 3)(2\ 4) \\ &= (1\ 3)(2\ 4)(1\ 2)(1\ 3)(2\ 4) \\ &= (1\ 4\ 2\ 3)(1\ 3)(2\ 4) \\ &= (3\ 4) \end{aligned}$$

and sends it to Alice. The last step is for Alice and Bob to compute the secret key. Thus Alice computes

$$\begin{aligned} K_A &= (w^b)^a \\ &= (1\ 4\ 3\ 2)^{-1}(3\ 4)(1\ 4\ 3\ 2) \\ &= (1\ 2\ 3\ 4)(3\ 4)(1\ 4\ 3\ 2) \\ &= (1\ 3\ 4)(1\ 4\ 3\ 2) \\ &= (1\ 4). \end{aligned}$$

Likewise Bob computes

$$\begin{aligned} K_B &= (w^a)^b \\ &= ((1\ 3)(2\ 4))^{-1}\,(2\ 3)(1\ 3)(2\ 4) \\ &= (1\ 3)(2\ 4)(2\ 3)(1\ 3)(2\ 4) \\ &= (1\ 3\ 4\ 2)(1\ 3)(2\ 4) \\ &= (1\ 4). \end{aligned}$$

Hence $K_A = K_B$ as desired.

1.3 Selecting a Platform Group

Every key exchange protocol relies on the selection of an appropriate platform group. If the group is too small or has certain undesirable properties, then the resultant key exchange protocol may be insecure. The Ko-Lee protocol was originally used with infinite braid groups [3]. The idea of using infinite groups for key exchange protocols is very attractive. In an ideal implementation an attacker would be forced to check an infinite amount of things, which would make the protocol impossible to break by brute force. The question of the feasibility of using infinite groups for non-commutative key exchange protocols is very much under investigation. The research monograph [4] explores this question in great depth.

Here we will instead examine the use of an infinite class of finite non-abelian groups known as the *generalized dihedral groups*. From a cryptographic viewpoint these groups are interesting because they are easy to construct, are generally non-abelian, and contain a very large abelian subgroup. This subgroup provides ample choices for the secret elements needed for the Ko-Lee protocol.

2 Background

2.1 Generalized Dihedral Groups

Throughout this paper we will work mostly multiplicatively. Hence instead of using the additive group $\mathbb{Z}_n$ we will use the isomorphic multiplicative cyclic group C_n. In particular we may take C_2 to be the set $\{-1, 1\}$ under the operation of integer multiplication.

Definition 2.1. Let A be an abelian group. The set $C_2 \times A$ together with the operation

$$(a, b)(c, d) = (ac, b^c d)$$

forms a group called the *generalized dihedral group* on A. We denote this group by $D(A)$.

The requirement of A being abelian is non-negotiable, as without it $D(A)$ fails to form a group. It may be shown that $D(C_n)$ is isomorphic to the classical dihedral group D_n; this explains the terminology. A proof of this fact may be found in Proposition 3.13. From this definition it is immediate that the order of $D(A)$ is twice that of A. Under the given operation it is easy to check that the identity element of $D(A)$ is the element $(1, e)$, where e is the identity element of A.

Convention. An element x of $D(A)$ will always be given as $x = (x_1, x_2)$.

Proposition 2.2. *The computation of the inverse of an element x from $D(A)$ comes in two cases:*

$$(x_1, x_2)^{-1} = \begin{cases} \left(x_1, x_2^{-1}\right), & x_1 = 1 \\ (x_1, x_2), & x_1 = -1. \end{cases}$$

This may be checked through direct computation. Now we will characterize exactly when $D(A)$ is non-abelian.

Convention. Let A be an abelian group. We adopt the notation that A_2 denotes the subgroup

$$A_2 = \left\{ g \in A \mid g^2 = e \right\}.$$

Notice that the elements of A_2 are precisely the elements that are their own inverses.

Proposition 2.3. *Let A be an abelian group. The group $D(A)$ is abelian if and only if $A = A_2$.*

Proof. Suppose that $A = A_2$, so that every element of A is its own inverse. Let x and y be elements of $D(A)$. We may now compute as follows:

$$\begin{aligned} (x_1, x_2)(y_1, y_2) &= (x_1 y_1, x_2^{y_1} y_2) \\ &= (x_1 y_1, x_2 y_2) && (x_2^{\pm 1} = x_2) \\ &= (x_1 y_1, x_2 y_2^{x_1}) && (y_2^{\pm 1} = y_2) \\ &= (y_1 x_1, y_2^{x_1} x_2) \\ &= (y_1, y_2)(x_1, x_2). \end{aligned}$$

Thus $D(A)$ is abelian. Conversely, suppose that $D(A)$ is abelian and let $x_2 \in A$. Then we have

$$\begin{aligned} (-1, e)(-1, x_2) &= (-1, x_2)(-1, e) \\ (1, x_2) &= (1, x_2^{-1}). \end{aligned}$$

Thus we have $x_2^2 = e$ and thus $x_2 \in A_2$. As x_2 is an arbitrary element of A we have shown that $A = A_2$. □

There are four main types of elements of $D(A)$ that we will concern ourselves with throughout this paper. For the sake of discussion we give them names as follows:

$D(A)$ Element Types		
Name	Form	Condition
Type 1	$(1, b)$	$b^2 = e$
Type 2	$(1, b)$	$b^2 \neq e$
Type 3	$(-1, b)$	$b^2 = e$
Type 4	$(-1, b)$	$b^2 \neq e$

Notice that Type 2 and Type 4 elements only exist when $D(A)$ is non-abelian. Next we examine the commutativity of the various types of elements.

Proposition 2.4. *The subgroup $\{(1, x_2) \mid x_2 \in A\}$ of $D(A)$ is isomorphic to A.*

Proposition 2.5. *Let x and y be elements of $D(A)$. The elements x and y commute if and only if one of the following conditions holds:*

1. $x = (1, x_2)$, $y = (1, y_2)$
2. $x = (-1, x_2)$, $y = (-1, y_2)$, *and* $x_2^2 = y_2^2$
3. $x = (-1, x_2)$, $y = (1, y_2)$, *and* $y_2^2 = e$
4. $x = (1, x_2)$, $y = (-1, y_2)$, *and* $x_2^2 = e$.

Proof. Suppose that any one of the above cases holds. Through direct calculation one may check that the given elements commute. Conversely, suppose that $xy = yx$ for some x and y from $D(A)$. This implies that $x_2^{y_1} y_2 = y_2^{x_1} x_2$. If $x_1 = 1$ and $y_1 = 1$, then the first case holds. Next, if $x_1 = -1$ and $y_1 = -1$, then $x_2^2 = y_2^2$ which means the second case holds. Next, if $x_1 = -1$ and $y_1 = 1$, then $x_2 y_2 = y_2^{-1} x_2$. This simplifies to $y_2^2 = e$, thus the third case holds. Lastly, if $x_1 = 1$ and $y_1 = -1$, then $x_2^{-1} y_2 = y_2 x_2$. This simplifies to $x_2^2 = e$, hence the fourth case holds. □

Proposition 2.6. *If $D(A)$ is non-abelian, then its center is the set of all Type 1 elements.*

This result follows directly from Proposition 2.5. Now recalling that $\{(1, x_2) \mid x_2 \in A\} \cong A$, we see that the center of $D(A)$ is isomorphic to A_2.

Proposition 2.7. *The probability of an element $(x_1, x_2) \in D(A)$ commuting with another randomly chosen element is given by the function*

$$P(x_1, x_2) = \begin{cases} 1 & x_1 = 1,\ x_2^2 = e \\ \frac{1}{2} & x_1 = 1,\ x_2^2 \neq e \\ \frac{|A_2|}{|A|} & x_1 = -1. \end{cases}$$

Proof. To begin we consider an element of the form $(1, x_2)$ where $x_2^2 = e$. This element is in the center and hence $P(1, x_2) = 1$. Next, consider an element $(1, x_2)$ where $x_2^2 \neq e$. From Proposition 2.5 this element only commutes with elements of the form $(1, y_2)$, hence $P(1, x_2) = \frac{|A|}{|D(A)|} = \frac{1}{2}$. Lastly, consider an element of the form $(-1, x_2)$. This element commutes with an element of the form $(1, y_2)$ only if $y_2^2 = e$, hence $P(-1, x_2) \geq \frac{|A_2|}{|D(A)|}$. Thus the only remaining possibility is that

$(-1, x_2)$ may commute with an element of the form $(-1, y_2)$. This occurs precisely when $x_2^2 = y_2^2$. For convenience we define the set of all elements of A that square to the same value as $y \in A$ by

$$\mathrm{Sq}(y) = \left\{ z \in A \mid z^2 = y^2 \right\}.$$

This results in

$$P(-1, x_2) = \frac{|A_2|}{|D(A)|} + \frac{\mathrm{Sq}(x_2)}{|D(A)|}.$$

It may be checked that $f_y \colon A_2 \to \mathrm{Sq}(y)$ defined by $f_y(t) = yt$ is a bijection, hence $|A_2| = |\mathrm{Sq}(x_2)|$. Thus we have the simplified expression

$$P(-1, x_2) = \frac{2|A_2|}{|D(A)|} = \frac{|A_2|}{|A|}.$$

□

It is worth noting that $\frac{|A_2|}{|A|}$ is precisely the reciprocal of the index $[A : A_2]$.

2.2 Strengths of $D(A)$

In the Ko-Lee protocol we saw the need for specifying an abelian subgroup C from which the secret elements a and b are selected. For many classes of non-abelian groups selecting an appropriate abelian subgroup is challenging. In $D(A)$ there is a very natural candidate for C, namely the abelian subgroup A from Proposition 2.4. Furthermore as A is index two, A is a maximal normal subgroup of $D(A)$.

Definition 2.8. Let A be a finite abelian group. We say that A has *signature* $[n_1, n_2, \ldots, n_k]$ if

$$A \cong C_{n_1} \times C_{n_2} \times \cdots \times C_{n_k}.$$

Due to the Fundamental Theorem of Finite Abelian Groups, every finite abelian group is given by some signature. It is important to note that the signature is by no means unique. There are at least as many signatures for A as ways to factor $|A|$ into powers of primes. This concept of signature allows us to quickly define an abelian group, and in turn a generalized dihedral group. Furthermore, with just the signature of A one may easily determine whether or not $D(A)$ is abelian.

Example 2.9. Consider $A = C_2 \times C_5 \times C_{10} \times C_{13}$. A valid signature for A is $[2, 5, 10, 13]$. Another more compact signature is $[10, 130]$, since $C_n \times C_m \cong C_{nm}$ when n and m are relatively prime.

Proposition 2.10. *Let A be a finite abelian group. The group $D(A)$ is abelian if and only if A is given by the signature $[2, 2, \ldots, 2]$.*

Proof. Suppose that A has signature $[2, 2, \ldots, 2]$. Then $A = A_2$, hence A is abelian. Conversely, suppose $D(A)$ is abelian and A has signature $[n_1, n_2, \ldots, n_k]$. Let α_i be a generator of C_{n_i}. As $D(A)$ is abelian, $A = A_2$ by Proposition 2.3. Then the element $(\alpha_1, \alpha_2, \ldots, \alpha_k) \in A$ must have order two. Thus we have $\alpha_i^2 = e$ for all i, hence C_{n_i} has order two. □

Convention. For the remainder of this paper we will assume that $D(A)$ is non-abelian unless otherwise noted.

2.3 Complexity Theory

Complexity theory concerns itself with the techniques of categorizing the difficulty of computational problems. One of the most common methods of categorization is asymptotic analysis, in which the growth of the input length of a problem is compared to the time required to solve the problem on that input. This is defined formally in terms of big-O notation. Let f and g be functions from the natural numbers into the positive reals. We write $f \in O(g)$ if there exist natural numbers d and N such that $f(n) \leq dg(n)$ whenever $n \geq N$. For an in-depth treatment of the basics of computational complexity see [5].

Example 2.11. Consider the functions f and g defined on $\mathbb{N}$ by $f(n) = 4n^3+2n+1$ and $g(n) = n^3$. Take $d = 12$ and $N = 1$. We may compute

$$\begin{aligned} dg(n) &= 12n^3 \\ &= 4n^3 + 4n^3 + 4n^3. \end{aligned}$$

Notice that for all $n \geq 1$ we have that $4n^3 \geq 2n$ and $4n^3 \geq 1$, hence for all $n \geq 1$ we have that

$$\begin{aligned} dg(n) &\geq 4n^3 + 2n + 1 \\ &\geq f(n). \end{aligned}$$

That is, $4n^3 + 2n + 1 \in O(n^3)$.

Now we will examine an algorithm that takes an integer n and returns a list of all divisors. Although seemingly simple, measuring this algorithm's complexity is deceptively complicated.

Example 2.12. This algorithm works by checking if each positive integer less than or equal to n divides n. Below is a sample implementation.

```
1:  procedure FindFactors(n)
2:      declare array[n]
3:      i := 1
4:      while i ≤ n do
5:          if n mod i = 0 then
6:              array[i] := True
7:          else
8:              array[i] := False
9:          end if
10:         i := i + 1
11:     end while
12:     return array
13: end procedure.
```

Notice that the loop runs only n times, which makes the complexity $O(n)$ with respect to the input integer n. For any practical implementation the input would be an integer expressed in binary, which means the proper way to measure the complexity of this algorithm would be with respect to the length of the binary string. For convenience we will denote the binary representation of n as n_{b2}. Suppose the input to our algorithm is

$$n_{b2} = 1\underbrace{0\ldots0}_{l-1}.$$

Converting this binary string to base 10 we obtain $n = 2^{l-1}$. That is, our algorithm must actually perform at least 2^{l-1} steps. Thus its complexity must be at least $O(2^{l-1})$ with respect to the length of the binary string input.

3 Analysis of Ko-Lee

3.1 Presentation of $D(A)$

There are many ways to represent a group, however one of the most compact methods uses generators and relations. Roughly speaking, a *presentation* of a group G is a set X of generators and their inverses, together with a set of relations R on X that define G. An element of R is simply an equation involving elements of X. We denote this presentation as $G = \langle X \mid R \rangle$. For a rigorous definition see [2].

Convention. Let $G = \langle X \mid R \rangle$. When enumerating such a presentation we will not list the inverses of the generators in X.

Definition 3.1. Let $G = \langle X \mid R \rangle$. A product of elements from X is called a *word* over G.

Definition 3.2. Let $w = x_1 x_2 \cdots x_n$ be a word over $G = \langle X \mid R \rangle$. The *length* of w, denoted $|w|$, is defined to be n.

Example 3.3. The multiplicative cyclic group C_n of order n is given by a single generator g and the single relation $g^n = e$. Thus C_n is given by the presentation $\langle g \mid g^n = e \rangle$. The elements of C_n with this presentation may be enumerated as

$$g, gg, ggg, \ldots, \underbrace{ggg\cdots g}_{n-1 \text{ times}}, e.$$

These expressions are words over the group $C_n = \langle g \mid g^n = e \rangle$. It is worth noting that group elements may be represented by many different words. For example, in the group C_4 the words gg and $gggggg$ represent the same element.

Of course, we are not restricted to presenting groups with only one generator and one relation.

Example 3.4. A presentation of the classical dihedral group D_n of order $2n$ is given by

$$\langle r,\ f \mid r^n = e, f^2 = e,\ frfr^{-1} = e \rangle.$$

The element r generates the rotations, the element f is a reflection, and the other elements come from words of the form $r^k f$.

Now we will examine a canonical presentation of an arbitrary finite abelian group.

Proposition 3.5. *Let A be a finite abelian group with signature $[n_1, n_2, \ldots, n_k]$. If α_i denotes a generator of C_{n_i} then A may be presented as*

$$A = \langle \alpha_i \mid \alpha_i^{n_i} = e,\ \alpha_i \alpha_j = \alpha_j \alpha_i \rangle .$$

Example 3.6. Let $A = C_{12} \times C_9$. A valid signature for A is $[12, 9]$. Thus A may be presented as

$$\langle \alpha_1, \alpha_2 \mid \alpha_1^{12} = e,\ \alpha_2^9 = e,\ \alpha_1 \alpha_2 = \alpha_2 \alpha_1 \rangle.$$

Notice that A also has signature $[3, 4, 9]$, therefore A is also presented as

$$\langle \mu_1, \mu_2, \mu_3 \mid \mu_1^3 = e,\ \mu_2^4 = e,\ \mu_3^9 = e,\ \mu_1\mu_2 = \mu_2\mu_1,\ \mu_1\mu_3 = \mu_3\mu_1,\ \mu_2\mu_3 = \mu_3\mu_2 \rangle.$$

Now we will show how a presentation for $D(A)$ may be constructed from a presentation for A. For this we will need the following definition.

Definition 3.7. Let $A = \langle X \mid R \rangle$ be a finite abelian group, and choose $c \notin A$. We define the group $J(A)$ by

$$J(A) = \langle X \cup \{c\} \mid R,\ c^2 = e,\ cxcx = e,\ \forall x \in X \rangle.$$

In what follows it is important to remember that the generator c is specifically chosen not to be an element of A. We will now show that $D(A)$ is isomorphic to $J(A)$.

Lemma 3.8. *If w is a word over A then in $J(A)$ we have $cwc = w^{-1}$.*

Proof. We will proceed by induction on the length of the word. If w is a word over A of length one, then w must be an element of X, hence by the defining relations $cwc = w^{-1}$. Now suppose that $cwc = w^{-1}$ for all words w of length n in A. Every word of length $n+1$ may be expressed as wx, where w is a word of length n and x is an element of X. Then we may compute

$$\begin{aligned} cwxc &= cwccxc && (c^2 = e) \\ &= cwcx^{-1} && (cxc = x^{-1}) \\ &= w^{-1}x^{-1} && (cwc = w^{-1}) \\ &= x^{-1}w^{-1} \\ &= (wx)^{-1}. \end{aligned}$$

Thus we have shown by induction that $cwc = w^{-1}$ for all words w over A. □

Lemma 3.9. *Let w be a word over $J(A)$ containing only one c. The word w is equivalent to another word w' over $J(A)$ of the form $w' = cx_1x_2\cdots x_n$ for $x_i \in X$.*

Proof. As w contains only a single c we have three cases to examine. If c takes the first position then w is already in the correct form. In the other cases c is either at the end of the word or somewhere in the middle of the word. If c appears at the end of the word we have that

$$\begin{aligned} w &= x_1x_2\cdots x_nc \\ &= ex_1x_2\cdots x_nc \\ &= ccx_1x_2\cdots x_nc && (c^2 = e) \\ &= c\left(x_1x_2\cdots x_n\right)^{-1} && (\text{Lemma 3.8}) \\ &= cx_n^{-1}\cdots x_2^{-1}x_1^{-1}. \end{aligned}$$

Thus we have found $w' = cx_n^{-1}\cdots x_2^{-1}x_1^{-1}$ as required. Next we examine the more complicated case when the c occurs somewhere in the middle of the word. In this case we have

$$\begin{aligned} w &= x_1x_2\cdots x_jcx_{j+1}\cdots x_n \\ &= ex_1x_2\cdots x_jcx_{j+1}\cdots x_n \\ &= ccx_1x_2\cdots x_jcx_{j+1}\cdots x_n && (c^2 = e) \\ &= c\left(x_1x_2\ldots x_j\right)^{-1}x_{j+1}\cdots x_n && (\text{Lemma 3.8}) \\ &= cx_j^{-1}\cdots x_2^{-1}x_1^{-1}x_{j+1}\cdots x_n. \end{aligned}$$

Thus we have $w' = cx_j^{-1}\cdots x_2^{-1}x_1^{-1}x_{j+1}\cdots x_n$. □

Next we show that every word in $J(A)$ reduces to a word of the form given in Lemma 3.9.

Proposition 3.10. *Every word w over $J(A)$ is equivalent to some word w' over $J(A)$ with at most one c.*

Proof. We will proceed by induction on the length of w. First note that the empty word contains zero c's and thus satisfies the statement. Next suppose that every word over $J(A)$ of length n satisfies the statement. Every word of length $n+1$ may be expressed as xw where $x \in X \cup \{c\}$ and w is a word of length n over $J(A)$. By the inductive hypothesis w contains at most one c. Thus if x is not c, then xw also contains at most one c. Now suppose that $xw = cw$. If w contains no c, then cw contains at most one c. Lastly suppose that w contains one c. Then by Lemma 3.9 we may write $w = cx_1x_2\cdots x_{n-1}$ where $x_i \in X$. We may now compute

$$cw = ccx_1x_2\cdots x_{n-1} = x_1x_2\cdots x_{n-1}.$$

Therefore cw has at most one c. We have thus shown that every word is equivalent to a word with at most one c. □

Lemma 3.11. *For any finite abelian group A we have $|J(A)| \leq |D(A)|$.*

Proof. Since A is a finite abelian group we may choose an isomorphism

$$A \cong C_{n_1} \times C_{n_2} \times \cdots \times C_{n_k}.$$

If we let α_i denote a fixed generator of C_{n_i}, then A may be presented as

$$A = \langle \alpha_i \mid \alpha_i^{n_i} = e,\ \alpha_i\alpha_j = \alpha_j\alpha_i \rangle$$

by Proposition 3.5. Let w be an arbitrary word over $J(A)$. By Lemma 3.9, Proposition 3.10, and the fact $\alpha_i\alpha_j = \alpha_j\alpha_i$ we have that

$$w = \alpha_1^{\beta_1}\alpha_2^{\beta_2}\cdots\alpha_k^{\beta_k} \qquad \text{or} \qquad w = c\alpha_1^{\beta_1}\alpha_2^{\beta_2}\cdots\alpha_k^{\beta_k}.$$

For the first type there are n_i distinct choices for each β_i, hence there are $n_1n_2\cdots n_k = |A|$ words of this type. Likewise there are $|A|$ words of the second type. Thus we have that $|J(A)| \leq 2|A| = |D(A)|$. □

Proposition 3.12. *For any finite abelian group $A = \langle X \mid R \rangle$ there exists an isomorphism $\theta\colon J(A) \to D(A)$.*

Proof. To begin we will define θ on the generators of $J(A)$. For each $x \in X$ we define $\theta(x) = (1, x)$ and for the special generator c we define $\theta(c) = (-1, e)$. As the generators of $J(A)$ satisfy the same relations as their images under θ we may uniquely extend θ to a homomorphism $\theta\colon J(A) \to D(A)$. Since the generators of $D(A)$ are in the image of θ we see that θ is surjective.

By Lemma 3.11 we have $|J(A)| \leq |D(A)|$. Since $\theta\colon J(A) \to D(A)$ is surjective it follows that $|J(A)| = |D(A)|$. Hence $\theta\colon J(A) \to D(A)$ is an isomorphism. □

Convention. For the remainder of this paper $D(A)$ is assumed to be written using the presentation

$$D(A) = \langle X \cup \{c\} \mid R,\ c^2 = e,\ cxcx = e,\ \forall x \in X \rangle$$

when A is presented as $\langle X \mid R \rangle$.

Now that we have a valid presentation for $D(A)$ we may show that the generalized dihedral groups really are a generalization of the classical dihedral groups.

Proposition 3.13. *For any cyclic group C_n, we have $D(C_n) \cong D_n$.*

Proof. Recall that $C_n = \langle g \mid g^n = e \rangle$. Then a presentation for $D(C_n)$ is given by

$$D(C_n) = \left\langle g, c \mid g^n = e,\ c^2 = e, cgcg = e \right\rangle.$$

This is precisely the presentation for D_n given in Example 3.4. □

3.2 Normal Forms

The idea of a normal form for elements of a group may be described as finding a standard presentation of an element. For a motivating example consider that 27 and 162 are congruent modulo 5. One may ask which representation is better. One would naturally prefer to represent these elements as 2 modulo 5. This is the key idea behind normal forms. If two elements are equivalent under some relation their normal forms should be the same. Furthermore this normal form should be simple and natural. The need for a normal form becomes even more apparent when we consider that every element in $D(A)$ has an infinite number of words that represent it. The importance of normal forms is amplified in the context of cryptography. A carefully chosen normal form allows for the hiding of information.

Example 3.14. Given an integer one may choose from a multitude of normal forms. One possible candidate for a normal form is the base ten representation, while another is its prime factorization. In the first normal form it is not clear at all by inspection which prime pair formed the composite integer 16637, while it is obvious in the second form, namely $127 \cdot 131$.

With this example in mind we wish to choose a normal form for elements of $D(A)$ that hides as much information as possible. The normal form will depend on the chosen signature for A: if the signature changes the normal form for the words will also change.

Convention. Given a finite abelian group A with signature $[n_1, n_2 \ldots, n_k]$, the element α_i will denote a fixed generator of C_{n_i}.

Definition 3.15. Let A be a finite abelian group with signature $[n_1, n_2 \ldots, n_k]$. A word w over A is said to be in *normal form* if

$$w = \alpha_1^{\beta_1} \alpha_2^{\beta_2} \cdots \alpha_k^{\beta_k}$$

where each β_i is reduced modulo n_i.

Definition 3.16. Let A be a finite abelian group with signature $[n_1, n_2 \ldots, n_k]$. A word w over $D(A)$ is said to be in *normal form* if

$$w = \alpha_1^{\beta_1} \alpha_2^{\beta_2} \cdots \alpha_k^{\beta_k} \qquad \text{or} \qquad w = c\alpha_1^{\beta_1} \alpha_2^{\beta_2} \cdots \alpha_k^{\beta_k}$$

where each β_i is reduced modulo n_i.

Example 3.17. Let A be a finite abelian group with signature $[5, 16, 9]$. We may consider the word $w = \alpha_1^3 \alpha_2^{10} \alpha_3^4$. This word is clearly in normal form, however it is important to note that the exponents are simply a typographical convenience. In practice the word is expressed as

$$w = \alpha_1\alpha_1\alpha_1\alpha_2\alpha_2\alpha_2\alpha_2\alpha_2\alpha_2\alpha_2\alpha_2\alpha_2\alpha_2\alpha_2\alpha_2\alpha_3\alpha_3\alpha_3\alpha_3.$$

The previous example exhibits that even when a word is given in normal form the value of β_i is not automatically apparent; it takes some computation to obtain its value. Specifically one must count the occurrences of α_i and then reduce this number modulo n_i. The following lemma is a formalization of this process.

Lemma 3.18. *Let A be a finite abelian group with signature $[n_1, n_2, \ldots, n_k]$. Given a word w over A and some fixed generator α_i we may produce the corresponding β_i in $O(|w|)$ steps.*

Proof. The value of β_i is precisely the number of occurrences of α_i in the word w, reduced modulo n_i. This procedure is captured in the following algorithm:

```
1: procedure CalcBeta(w = x_1 x_2 ··· x_l, α_i)
2:     β := 0
3:     j := 1
4:     while j ≤ l do
5:         if x_j = α_i then
6:             β := β + 1
7:         end if
8:         j := j + 1
9:     end while
10:    return β mod n_i
11: end procedure.
```

Notice that the main loop runs exactly l times. By recalling that l is the length of the word w we see that the algorithm runs in $O(|w|)$ steps. □

Lemma 3.19. *Let A be a finite abelian group with signature $[n_1, n_2 \ldots, n_k]$. Every word w over A may be reduced to its normal form in polynomial time with respect to the length of w.*

Proof. For any word w over A we may perform the following algorithm:

```
1: procedure NormalFormA(w = α_{m_1} α_{m_2} ··· α_{m_l})
2:     j := 1
3:     while j ≤ l do
4:         β_{m_j} := CalcBeta(w, α_{m_j})
5:         j := j + 1
6:     end while
7:     return α_1^{β_1} α_2^{β_2} ··· α_k^{β_k}
8: end procedure.
```

This algorithm is a formalization of the process of commuting all of the generators, while working modulo the order of the respective cyclic groups. We will now examine its complexity. Notice that the outer loop runs $|w|$ times, while the CalcBeta operation itself runs in $O(|w|)$ steps. This results in a total complexity of $O\left(|w|^2\right)$. □

Lemma 3.20. *Given a word w over $D(A)$ we may in polynomial time produce an element w' with at most one c equivalent to w.*

Proof. The methodologies utilized in the algorithm below are inspired from the inductive proof of Proposition 3.10. Specifically the c's are chosen in pairs and then removed by inverting all of the elements between them. This is formally given by the following algorithm:

```
1: procedure RemoveExtraCs(w = x_1 x_2 ··· x_l)
2:     i := 1
3:     while i ≤ l do
4:         if x_i = c then
5:             j := i + 1
6:             while j ≤ l and x_j ≠ c do
7:                 j := j + 1
```

```
            end while
            Replace $x_i x_{i+1} \cdots x_{j-1} x_j$ with $x_{j-1}^{-1} \cdots x_{i+1}^{-1}$
        end if
        $i := i + 1$
    end while
end procedure.
```

Notice that both the outer and inner loops run at most $|w|$ times, hence the algorithm runs in $O\left(|w|^2\right)$ steps. □

Proposition 3.21. *Let A be a finite abelian group with signature $[n_1, n_2 \ldots, n_k]$. Every word w over $D(A)$ may be reduced to its normal form in polynomial time with respect to the length of w.*

Proof. The first step in the procedure is to put the word in a form with at most one c. Then if the word contains a c it is brought to the front. After that it suffices to put the remainder of the word in the normal form over A using Lemma 3.19. The following algorithm makes these steps rigorous:

```
procedure NormalFormDA($w = x_1 x_2 \cdots x_l$)
    $w :=$ RemoveExtraCs($w$)
    $w :=$ RemoveExtraCs($cw$)
    $w :=$ RemoveExtraCs($cw$)
    if $x_1 = c$ then
        return $c \cdot$ NormalFormA($x_2 \cdots x_l$)
    end if
    return NormalFormA($w$)
end procedure.
```

By recalling that the algorithms RemoveExtraCs and NormalFormA both run in $O\left(|w|^2\right)$ steps, we obtain a total complexity of $O(|w|^2)$. □

We will now show that one may convert between the word form of an element and the tuple form in polynomial time. This allows us to switch between the two forms within our algorithms.

Proposition 3.22. *Given a word w over $D(A)$ that corresponds to the element (x_1, x_2), the word x_2 over A may be produced in polynomial time.*

Proof. Let w be a word over $D(A)$. By Proposition 3.21 we may put w in normal form in $O\left(|w|^2\right)$ steps. Then x_2 is given by $\theta(w)$. □

3.3 Ko-Lee with $D(A)$

Now that we have a presentation for $D(A)$ in which one may compute products and normal forms efficiently, we may examine how one may use an arbitrary generalized dihedral group as a platform group for the Ko-Lee protocol. First we will examine an implementation of this protocol with a specific generalized dihedral group.

Example 3.23. Let A be the abelian group given by the signature $[4, 9, 19, 31]$. Alice and Bob agree on choosing A as the abelian subgroup for selecting the secret elements. They then select the public element w to be

$$w = c\alpha_1^3\alpha_2^6\alpha_3^{15}\alpha_4^{12}.$$

Now Alice and Bob select their secret elements from A. Alice selects

$$a = \alpha_1^3\alpha_2^4\alpha_3^{17}\alpha_4^{26}$$

while Bob selects

$$b = \alpha_1^2\alpha_2^3\alpha_3^{14}\alpha_4^{15}.$$

Now Alice computes w^a by

$$\begin{aligned}
w^a &= \left(\alpha_1^3\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right)^{-1}\left(c\alpha_1^3\alpha_2^6\alpha_3^{15}\alpha_4^{12}\right)\left(\alpha_1^3\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right)\\
&= cc\left(\alpha_1^3\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right)^{-1}c\left(\alpha_1^3\alpha_2^6\alpha_3^{15}\alpha_4^{12}\right)\left(\alpha_1^3\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right)\\
&= c\left(\alpha_1^3\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right)\left(\alpha_1^3\alpha_2^6\alpha_3^{15}\alpha_4^{12}\right)\left(\alpha_1^3\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right)\\
&= c\alpha_1^9\alpha_2^{14}\alpha_3^{49}\alpha_4^{64}\\
&= c\alpha_1^1\alpha_2^5\alpha_3^{11}\alpha_4^2.
\end{aligned}$$

Next Bob computes the value of w^b by

$$\begin{aligned}
w^b &= \left(\alpha_1^2\alpha_2^3\alpha_3^{14}\alpha_4^{15}\right)^{-1}\left(c\alpha_1^3\alpha_2^6\alpha_3^{15}\alpha_4^{12}\right)\left(\alpha_1^2\alpha_2^3\alpha_3^{14}\alpha_4^{15}\right)\\
&= cc\left(\alpha_1^2\alpha_2^3\alpha_3^{14}\alpha_4^{15}\right)^{-1}c\left(\alpha_1^3\alpha_2^6\alpha_3^{15}\alpha_4^{12}\right)\left(\alpha_1^2\alpha_2^3\alpha_3^{14}\alpha_4^{15}\right)\\
&= c\left(\alpha_1^2\alpha_2^3\alpha_3^{14}\alpha_4^{15}\right)\left(\alpha_1^3\alpha_2^6\alpha_3^{15}\alpha_4^{12}\right)\left(\alpha_1^2\alpha_2^3\alpha_3^{14}\alpha_4^{15}\right)\\
&= c\alpha_1^7\alpha_2^{12}\alpha_3^{43}\alpha_4^{42}\\
&= c\alpha_1^3\alpha_2^3\alpha_3^5\alpha_4^{11}.
\end{aligned}$$

Now Alice computes the secret key by

$$\begin{aligned}
K_A &= (w^b)^a\\
&= \left(\alpha_1^3\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right)^{-1}\left(c\alpha_1^3\alpha_2^3\alpha_3^5\alpha_4^{11}\right)\left(\alpha_1^3\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right)\\
&= cc\left(\alpha_1^3\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right)^{-1}c\left(\alpha_1^3\alpha_2^3\alpha_3^5\alpha_4^{11}\right)\left(\alpha_1^3\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right)\\
&= c\left(\alpha_1^3\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right)\left(\alpha_1^3\alpha_2^3\alpha_3^5\alpha_4^{11}\right)\left(\alpha_1^3\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right)\\
&= c\alpha_1^9\alpha_2^{11}\alpha_3^{39}\alpha_4^{63}\\
&= c\alpha_1\alpha_2^2\alpha_3\alpha_4.
\end{aligned}$$

Next Bob computes the secret key by

$$\begin{aligned}
K_B &= (w^a)^b\\
&= \left(\alpha_1^2\alpha_2^3\alpha_3^{14}\alpha_4^{15}\right)^{-1}\left(c\alpha_1^1\alpha_2^5\alpha_3^{11}\alpha_4^2\right)\left(\alpha_1^2\alpha_2^3\alpha_3^{14}\alpha_4^{15}\right)\\
&= cc\left(\alpha_1^2\alpha_2^3\alpha_3^{14}\alpha_4^{15}\right)^{-1}c\left(\alpha_1^1\alpha_2^5\alpha_3^{11}\alpha_4^2\right)\left(\alpha_1^2\alpha_2^3\alpha_3^{14}\alpha_4^{15}\right)\\
&= c\left(\alpha_1^2\alpha_2^3\alpha_3^{14}\alpha_4^{15}\right)\left(\alpha_1^1\alpha_2^5\alpha_3^{11}\alpha_4^2\right)\left(\alpha_1^2\alpha_2^3\alpha_3^{14}\alpha_4^{15}\right)\\
&= c\alpha_1^5\alpha_2^{11}\alpha_3^{39}\alpha_4^{32}\\
&= c\alpha_1\alpha_2^2\alpha_3\alpha_4.
\end{aligned}$$

Thus $K_A = K_B$ as desired.

Convention. For the remainder of this paper we assume that the Ko-Lee key exchange protocol has been properly set up on $D(A)$. We let w denote the public element from step two of the Ko-Lee protocol, and we let a and b denote Alice's and Bob's secret elements chosen from the abelian subgroup C.

We have seen how a secret key may be established using a generalized dihedral group as a platform group for the Ko-Lee protocol, however we have not yet analyzed how secure this really is. We will start by exhibiting a general attack on the Ko-Lee protocol.

Conjugate Attack on Ko-Lee:

Suppose that from w and w^a one may obtain some $x \in C$ such that $w^x = w^a$. Then we may compute

$$\left(w^b\right)^x = x^{-1}b^{-1}wbx.$$

As $x \in C$, we have that x and b commute, thus:

$$\begin{aligned}\left(w^b\right)^x &= x^{-1}b^{-1}wbx \\ &= b^{-1}x^{-1}wxb \\ &= (w^x)^b && (w^x = w^a) \\ &= (w^a)^b = K_B.\end{aligned}$$

Thus an attacker may always produce the secret key if they can find such an x. The problem of finding such an x is known as the *conjugacy search problem*. Notice that the condition that x is an element of the commuting subgroup C may be weakened to demanding that x simply commutes with the secret element b. At a glance this observation seems frivolous, as how could one demand that x commutes with b without knowing the value of b? From Proposition 2.5 we see that certain types of elements always commute while others never do. In the next section we will see how it is not the value of b that matters, but only its type.

3.4 Conditions for Ko-Lee

In the Ko-Lee protocol the secret elements a and b are demanded to commute. From Proposition 2.5 it is evident that only a few combinations of element types commute. This places restrictions on the possible choices for a and b. These restrictions are displayed in the table below.

	Type 1	Type 2	Type 3	Type 4
Type 1	Yes	Yes	Yes	Yes
Type 2	Yes	Yes	No	No
Type 3	Yes	No	Yes	No
Type 4	Yes	No	No	Maybe

Based on the table above we are able to discard all cases in which a and b do not commute. Next we will eliminate a few more cases on the grounds that they result in the *public* transmission of the *secret* key.

Lemma 3.24. *If a is Type 1, then the secret key is the public element w^b.*

Proof. As a is Type 1 it is in the center of $D(A)$. Hence,

$$w^a = a^{-1}wa = wa^{-1}a = w.$$

Then the secret key is given by

$$K_B = (w^a)^b = w^b.$$ □

The previous argument was made on the secret element a, however without loss of generality the same is true when one chooses b to be Type 1. This completely eliminates the feasibility of either Alice or Bob selecting a Type 1 secret element. We will now show that two other cases result in the public transmission of the secret key.

Lemma 3.25. *If a and b are both Type 3, then the secret key is the public element w.*

Proof. We will consider two cases for w. If $w_1 = 1$ then we have

$$\begin{aligned} w^a &= (-1, a_2)^{-1}(1, w_2)(-1, a_2) \\ &= (-1, a_2)(1, w_2)(-1, a_2) \\ &= (-1, a_2 w_2)(-1, a_2) \\ &= \left(1, w_2^{-1} a_2^{-1} a_2\right) \\ &= \left(1, w_2^{-1}\right) \\ &= w^{-1}. \end{aligned}$$

Similarly one may show that $w^b = w^{-1}$, therefore $w^a = w^b$. Bob computes the key

$$\begin{aligned} K_B &= (w^a)^b \\ &= (w^b)^b \\ &= w^{b^2} \\ &= w^e = w. \qquad (b \text{ is Type 3}) \end{aligned}$$

Hence $K_B = w$ as claimed. Now we will consider the case when $w_1 = -1$. We compute

$$\begin{aligned} w^a &= (-1, a_2)^{-1}(-1, w_2)(-1, a_2) \\ &= (-1, a_2)(-1, w_2)(-1, a_2) \\ &= \left(1, a_2^{-1} w_2\right)(-1, a_2) \\ &= \left(-1, w_2^{-1} a_2 a_2\right). \end{aligned}$$

Now recall that as a is Type 3 we have $a_2^2 = e$, hence $w^a = \left(-1, w_2^{-1}\right)$. Thus similarly we have that $w^b = \left(-1, w_2^{-1}\right)$. Therefore $w^a = w^b$ and by the previous argument we see that $K_B = w$. □

Lemma 3.26. *If a and b are both Type 4, then the secret key is the public element w.*

Proof. As a and b commute and are both Type 4 we have by Proposition 2.5 that $a_2^2 = b_2^2$. If $w_1 = 1$ then

$$\begin{aligned} w^a &= (-1, a_2)^{-1}(1, w_2)(-1, a_2) \\ &= (-1, a_2)(1, w_2)(-1, a_2) \\ &= (-1, a_2 w_2)(-1, a_2) \\ &= (1, w_2^{-1} a_2^{-1} a_2) \\ &= \left(1, w_2^{-1}\right) = w^{-1}. \end{aligned}$$

By a similar computation we find that $w^b = w^{-1}$ also. Now Bob computes the key by

$$\begin{aligned} K_B &= (w^a)^b \\ &= \left(w^{-1}\right)^b \\ &= \left(w^b\right)^{-1} \\ &= \left(w^{-1}\right)^{-1} = w. \end{aligned}$$

Next we will consider the other case. If $w_1 = -1$ then

$$\begin{aligned} w^a &= (-1, a_2)^{-1}(-1, w_2)(-1, a_2) \\ &= (-1, a_2)(-1, w_2)(-1, a_2) \\ &= \left(1, a_2^{-1} w_2\right)(-1, a_2) \\ &= \left(-1, w_2^{-1} a_2^2\right). \end{aligned}$$

A similar computation yields $w^b = \left(-1, w_2^{-1} b_2^2\right)$. Then since $a_2^2 = b_2^2$, we have $w^a = w^b$. Thus Bob computes the key as

$$\begin{aligned} K_B &= (w^a)^b \\ &= (w^b)^b \\ &= w^{b^2} \\ &= w^e = w. \qquad (b \text{ is Type 4}) \end{aligned}$$

In all cases we have shown the private key to be the public element w. □

In short, we have the following proposition.

Proposition 3.27. *If either secret element a or b is not Type 2, then the private key is publicly transmitted.*

From the above proposition we see that the only hope for a secure protocol is the case when both secret elements are Type 2. Returning to Example 3.23 we see that a and b were both Type 2. Furthermore, it is apparent that the secret key was never publicly transmitted. The next proposition shows that the private key generated when a and b are Type 2 is dependent on the values of both secret elements. In the interest of security this is precisely what one would hope for.

Proposition 3.28. *If the public element w is of the form $w = (-1, w_2)$ and the private elements a and b are both Type 2, then the private key is given by*

$$K = \left(-1, a_2^2 b_2^2 w_2\right).$$

Proof. We obtain the value for K through direct computation. First we compute

$$\begin{aligned} w^a &= (1, a_2)^{-1}(-1, w_2)(1, a_2) \\ &= \left(1, a_2^{-1}\right)(-1, w_2)(1, a_2) \\ &= (-1, a_2 w_2)(1, a_2) \\ &= \left(-1, a_2^2 w_2\right). \end{aligned}$$

Similarly we find that $w^b = \left(-1, b_2^2 w_2\right)$. Now Bob computes the secret key as

$$\begin{aligned} K_B &= (w^a)^b \\ &= (1, b_2)^{-1}\left(-1, a_2^2 w_2\right)(1, b_2) \\ &= \left(1, b_2^{-1}\right)\left(-1, a_2^2 w_2\right)(1, b_2) \\ &= \left(-1, b_2 a_2^2 w_2\right)(1, b_2) \\ &= \left(-1, a_2^2 b_2^2 w_2\right). \end{aligned}$$

Thus $K = \left(-1, a_2^2 b_2^2 w_2\right)$ as claimed. □

Notice that within this secret key it is the squares of the secret elements that matter, and not the elements themselves. This suggests that it may be sufficient for an attacker to find an element $x \in D(A)$ that satisfies $x^2 = a^2$ in order to recover the secret key.

3.5 Conjugate Attack on Ko-Lee

In this section we will reduce the problem of finding $x \in D(A)$ such that $w^x = w^a$ to the problem of solving a quadratic equation over the finite abelian group A. To begin we will explore when the equation $x^2 = g$ has a solution for $x \in A$ given a fixed element $g \in A$. We will then construct an algorithm that produces such a solution. We will now consider a motivating example.

Example 3.29. Consider the abelian group A with signature $[5, 16, 9]$. One may ask whether the equation

$$x^2 = \alpha_1^1 \alpha_2^4 \alpha_3^5$$

is solvable over A. One may be inclined to believe that this equation has no solution as not all of the exponents are even. However the equation does in fact have a solution. It is easily checked that $x = \alpha_1^3 \alpha_2^2 \alpha_3^7$ is a solution by computing

$$\begin{aligned} x^2 &= \left(\alpha_1^3 \alpha_2^2 \alpha_3^7\right)\left(\alpha_1^3 \alpha_2^2 \alpha_3^7\right) \\ &= \alpha_1^6 \alpha_2^4 \alpha_3^{14} \\ &= \alpha_1^1 \alpha_2^4 \alpha_3^5. \end{aligned}$$

Lemma 3.30. *Let $g = \alpha_1^{\beta_1} \alpha_2^{\beta_2} \cdots \alpha_k^{\beta_k}$ be an element of a finite abelian group A with signature $[n_1, n_2, \ldots, n_k]$. If $x^2 = g$ has a solution in A, then either β_i is even or $\beta_i + n_i$ is even.*

Proof. Suppose that $x_0 = \alpha_1^{\lambda_1} \alpha_2^{\lambda_2} \cdots \alpha_k^{\lambda_k}$ is a solution to $x^2 = g$. We now compute

$$\begin{aligned} x_0^2 &= \alpha_1^{\beta_1} \alpha_2^{\beta_2} \cdots \alpha_k^{\beta_k} \\ \alpha_1^{2\lambda_1} \alpha_2^{2\lambda_2} \cdots \alpha_k^{2\lambda_k} &= \alpha_1^{\beta_1} \alpha_2^{\beta_2} \cdots \alpha_k^{\beta_k}. \end{aligned}$$

Thus $\beta_i \equiv 2\lambda_i \pmod{n_i}$ for each i. Therefore there exists some $q_i \in \mathbb{Z}$ such that

$$\beta_i = n_i q_i + 2\lambda_i.$$

If n_i is even then β_i is even. Thus we may assume that n_i is odd. We are left with two cases, either q_i is even or odd. If q_i is even, then β_i is once again even. Thus we may take q_i to be odd. Since q_i is odd there exists some integer l_i such that $q_i = 2l_i + 1$. We see that $\beta_i + n_i$ is even, since

$$\begin{aligned} \beta_i + n_i &= (n_i q_i + 2\lambda_i) + n_i \\ &= 2\left(n_i(l_i + 1) + \lambda_i\right). \end{aligned}$$

Thus in all cases either β_i or $\beta_i + n_i$ is even. □

Lemma 3.31. *Let A be a finite abelian group with signature $[n_1, n_2, \ldots, n_k]$. Suppose that $g = \alpha_1^{\beta_1} \alpha_2^{\beta_2} \cdots \alpha_k^{\beta_k}$ is a word over A. We may determine if $x^2 = g$ has a solution in A in polynomial time with respect to the length of g.*

Proof. By the contrapositive of Lemma 3.30, if β_i and $\beta_i + n_i$ both fail to be even, then the equation does not have a solution. This fact yields the following algorithm:

```
1: procedure IsSolvable(g = α_{m_1} α_{m_2} ··· α_{m_l})
2:     i := 1
3:     while i ≤ l do
4:         β_{m_i} = CalcBeta(α_{m_i})
```

```
        if β_{m_i} is odd and β_{m_i} + n_{m_i} is odd then
            return False
        end if
        i := i + 1
    end while
    return True
end procedure.
```

As l is the length of g the main loop runs $|g|$ times. Recalling that the CalcBeta procedure runs in $O(|g|)$ steps, we see that the overall complexity is $O\left(|g|^2\right)$. □

Proposition 3.32. *Let $g = \alpha_1^{\beta_1}\alpha_2^{\beta_2}\cdots\alpha_k^{\beta_k}$ be an element of A with signature $[n_1, n_2, \ldots, n_k]$. If $x^2 = g$ is solvable over A, we may produce a solution in polynomial time with respect to the length of g.*

Proof. It is easy to see that $x = \alpha_1^{\lambda_1}\alpha_2^{\lambda_2}\cdots\alpha_k^{\lambda_k}$ is a solution of $x^2 = g$ if and only if $\beta_i \equiv 2\lambda_i \pmod{n_i}$; one may follow the same proof techniques from Lemma 3.30. Thus to find a solution it suffices to find each λ_i of the correct form. As $x^2 = g$ is assumed to be solvable we have by Lemma 3.30 that either β_i or $\beta_i + n_i$ is even. In the first case we have that $\beta_i = 2l$ for some $l \in \mathbb{Z}$. Thus we may take $\lambda_i = l = \frac{\beta_i}{2}$. If β_i is not even, then $\beta_i + n_i$ must be, hence $\beta_i + n_i = 2m$ for some $m \in \mathbb{Z}$. Thus we may take $\lambda_i = m = \frac{\beta_i + n_i}{2}$. This method of selecting the correct λ_i is the basis for the following algorithm:

```
procedure ProduceSolution(g = α_{m_1} α_{m_2} ··· α_{m_l})
    i := 1
    while i ≤ l do
        β_{m_i} = CalcBeta(α_{m_i})
        if β_{m_i} is even then
            λ_{m_i} := β_{m_i} / 2
        else
            λ_{m_i} := (β_{m_i} + n_{m_i}) / 2
        end if
    end while
    return α_1^{λ_1} α_2^{λ_2} ··· α_k^{λ_k}
end procedure.
```

As l is the length of g the main loop runs at most $|g|$ times. Recalling that the CalcBeta procedure runs in $O(|g|)$ steps we see that the overall complexity is $O\left(|g|^2\right)$. □

Now we have the algorithms necessary to solve any quadratic of the form $x^2 = g$ over an abelian group A with a known signature. Next we prove a small result that shows if a Type 2 element exists in $D(A)$ one may find it quickly.

Lemma 3.33. *Let A be a finite abelian group with signature $[n_1, n_2, \ldots, n_k]$. If $D(A)$ is non-abelian one may produce a Type 2 element in polynomial time with respect to k.*

Proof. As $D(A)$ is non-abelian, by Proposition 2.10 there exists an n_i where $n_i \neq 2$. This means that $\alpha_i^2 \neq e$, hence $(1, \alpha_i)$ is Type 2. We show that this element may be found in polynomial time with respect to the length of the signature by the following algorithm:

```
procedure FindType2([n_1, n_2, ..., n_k])
    i := 1
```

```
    while i ≤ k do
        if n_i ≠ 2 then
            return α_i
        end if
    end while
end procedure.
```

The main loop will run at most k times, thus the algorithm has complexity $O(k)$. □

Now we will show that the problem of finding an element x such that $w^x = w^a$ reduces to the problem of solving a quadratic over A. This shows that the conjugate attack described in Section 3.3 may be implemented in polynomial time.

Proposition 3.34. *Let w and w^a be public elements of $D(A)$ where A has signature $[n_1, n_2, \ldots, n_k]$. If it is known that the secret element a is Type 2, then an attacker may produce in polynomial time a Type 2 element x such that $w^x = w^a$.*

Proof. First, if $w_1 = 1$ then we have

$$\begin{aligned} w^a &= (1, a_2)^{-1}(1, w_2)(1, a_2) \\ &= (1, a_2^{-1})(1, w_2)(1, a_2) \\ &= (1, a_2^{-1} w_2 a_2) \\ &= (1, w_2) = w. \end{aligned}$$

As the value of w^a is not actually dependent on the value of a, any other Type 2 element x will satisfy $w^x = w$. By Lemma 3.33 we may produce a Type 2 element in polynomial time with respect to the length of the signature; that is, we may produce such an x.

In the second case, if $w_1 = -1$ then we have

$$\begin{aligned} w^a &= (1, a_2)^{-1}(-1, w_2)(1, a_2) \\ &= \left(1, a_2^{-1}\right)(-1, w_2)(1, a_2) \\ &= (-1, a_2 w_2)(1, a_2) \\ &= (-1, a_2 w_2 a_2) \\ &= \left(-1, a_2^2 w_2\right). \end{aligned}$$

As w and w^a are public we may obtain a_2^2 by multiplying $a_2^2 w_2$ on the right by w_2^{-1}. Now notice that the value of w^a is determined completely by a_2^2 and w_2, thus it suffices to find an element $x = (1, x_2)$ such that $x_2^2 = a_2^2$. As this equation is known to have a solution, namely a_2, we may invoke the algorithm from Proposition 3.32. This produces a solution x for this equation in polynomial time. As a is Type 2 by definition $a_2^2 \neq e$, hence $x_2^2 \neq e$. This means that x is Type 2 and $w^x = \left(-1, x_2^2 w_2\right) = \left(-1, a_2^2 w_2\right) = w^a$ as desired. □

Recall from Proposition 3.28 that the case in which both secret elements are Type 2 results in the construction of a secret key that is dependent on both secret elements. Although this seemed relatively secure, our next theorem presents an attack that is guaranteed to produce the secret key in polynomial time.

Theorem 3.35. *Let w, w^a, and w^b be public elements from $D(A)$. If the secret elements a and b are selected to be Type 2, then an attacker may produce the secret key in polynomial time.*

Proof. As a is Type 2 and w^a is public, by Proposition 3.34 an attacker may produce in polynomial time a Type 2 element $x \in D(A)$ such that $w^x = w^a$. From Proposition 2.5 all Type 2 elements commute with one another, hence x must commute with b. The attacker may then compute

$$(w^b)^x = (w^x)^b = (w^a)^b = w^{ab} = K_A.$$

That is, an attacker may produce the secret key in polynomial time. □

Thus we have proven that the Ko-Lee key exchange protocol is never secure when the platform group is a generalized dihedral group. We conclude by returning to the key exchange from Example 3.23 from the perspective of an attacker.

Example 3.36. Recall that the platform group was $D(A)$ where A had signature [4,9,19,31]. The only public elements were:

$$w = c\alpha_1^3\alpha_2^6\alpha_3^{15}\alpha_4^{12} \qquad w^a = c\alpha_1^1\alpha_2^5\alpha_3^{11}\alpha_4^2 \qquad w^b = c\alpha_1^3\alpha_2^3\alpha_3^5\alpha_4^{11}.$$

This is precisely the information an attacker would have. With the knowledge that the secret elements must be Type 2, the attacker may invoke Proposition 3.34. Notice that $w_1 = -1$ since w begins with a c. This means that w^a takes the form $w^a = \left(-1, a_2^2 w_2\right)$. Recall from Proposition 3.22 that w_2 is given by $\theta(w)$, hence $w_2 = \alpha_1^3\alpha_2^6\alpha_3^{15}\alpha_4^{12}$ and $(w^a)_2 = \alpha_1^1\alpha_2^5\alpha_3^{11}\alpha_4^2$. Thus an attacker may compute the value of a_2^2 by

$$\begin{aligned} a_2^2 &= a_2^2 w_2 w_2^{-1} \\ &= (w^a)_2 w_2^{-1} \\ &= \left(\alpha_1^1\alpha_2^5\alpha_3^{11}\alpha_4^2\right)\left(\alpha_1^3\alpha_2^6\alpha_3^{15}\alpha_4^{12}\right)^{-1} \\ &= \left(\alpha_1^1\alpha_2^5\alpha_3^{11}\alpha_4^2\right)\left(\alpha_1\alpha_2^3\alpha_3^4\alpha_4^{19}\right) \\ &= \alpha_1^2\alpha_2^8\alpha_3^{15}\alpha_4^{21}. \end{aligned}$$

Now recall that if an attacker finds $x_2 \in A$ such that $x_2^2 = a_2^2 = \alpha_1^2\alpha_2^8\alpha_3^{15}\alpha_4^{21}$, then $(1, x_2)$ is a solution to $w^x = w^a$. This may be accomplished in polynomial time by the algorithm from Proposition 3.32. Simply put, the attacker divides β_i by two when it is even, and divides $\beta_i + n_i$ by two when β_i is odd. Thus the attacker obtains

$$x_2 = \alpha_1\alpha_2^4\alpha_3^{17}\alpha_4^{26}.$$

Now notice that $\theta^{-1}(1, x_2) = \alpha_1\alpha_2^4\alpha_3^{17}\alpha_4^{26}$. By Theorem 3.35 the attacker may compute the key as follows:

$$\begin{aligned} (w^b)^x &= \left(\alpha_1\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right)^{-1}\left(c\alpha_1^3\alpha_2^3\alpha_3^5\alpha_4^{11}\right)\left(\alpha_1\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right) \\ &= cc\left(\alpha_1\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right)^{-1} c\left(\alpha_1^3\alpha_2^3\alpha_3^5\alpha_4^{11}\right)\left(\alpha_1\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right) \\ &= c\left(\alpha_1\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right)\left(\alpha_1^3\alpha_2^3\alpha_3^5\alpha_4^{11}\right)\left(\alpha_1\alpha_2^4\alpha_3^{17}\alpha_4^{26}\right) \\ &= c\alpha_1^5\alpha_2^{11}\alpha_3^{39}\alpha_4^{63} \\ &= c\alpha_1\alpha_2^2\alpha_3\alpha_4 = K. \end{aligned}$$

By returning to Example 3.23, we see this is precisely the secret key computed by Alice and Bob. In short, the attacker is able to produce the secret key solely from public elements.

References

[1] Whitfield Diffie and Martin E. Hellman, *New directions in cryptography*, IEEE Trans. Information Theory **IT-22** (1976), no. 6, 644–654.

[2] Thomas W. Hungerford, *Algebra*, Graduate Texts in Mathematics, vol. 73, Springer-Verlag, New York-Berlin, 1980, Reprint of the 1974 original.

[3] Ki Hyoung Ko, Sang Jin Lee, Jung Hee Cheon, Jae Woo Han, Ju-sung Kang, and Choonsik Park, *New public-key cryptosystem using braid groups*, Advances in cryptology—CRYPTO 2000 (Santa Barbara, CA), Lecture Notes in Comput. Sci., vol. 1880, Springer, Berlin, 2000, pp. 166–183.

[4] Alexei Myasnikov, Vladimir Shpilrain, and Alexander Ushakov, *Group-based cryptography*, Advanced Courses in Mathematics, CRM Barcelona, Birkhäuser Verlag, Basel, 2008.

[5] Michael Sipser, *Introduction to the theory of computation*, 3rd ed., Course Technology, 2012.