UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS

FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN

LICENCIATURA EN CIENCIA DE LA COMPUTACIÓN

TRABAJO DE DIPLOMA

Algoritmo criptográfico basado en la presentación de grupos.

AUTOR:

Yairon Cid Ruiz.

TUTORES:

Dr. Eberto R. Morgado Morales.

MSc. Guillermo Sosa Gómez.

"Año 56 de la Revolución"

Santa Clara

2014

DICTAMEN

El que subscribe, Yairon Cid Ruiz, hago constar que el trabajo titulado **"Algoritmo criptográfico basado en la presentación de grupos"** fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de los estudios de la especialidad de Ciencia de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

_____

Firma del autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del tutor Firma del Jefe del Seminario

A mi familia,

mi guía de siempre.

AGRADECIEMIENTOS

Quiero expresar mis más sinceros agradecimientos:

A mis padres por ser mi origen de enriquecimiento, las personas de las que más he aprendido y una fuente interminable de aprecio, cariño, comprensión y amor.

A mi abuela por haberme entregado de forma incondicional todo su amor y estar presente para mí en todo momento.

A mi hermanita Jessica esa personita con la que siempre discuto y peleo, pero que al final siempre me apoya y me entiende.

A mi tía Dayi y a mi tío Lili por acogerme y quererme, a mis abuelos, en fin a toda mi familia que siempre ha estado a mi lado apoyándome.

A mis tutores Eberto Morgado Morales y Guillermo Sosa Gómez por enseñarme a trabajar e investigar con paciencia y dedicación; por compartir conmigo y ayudarme.

A Morgado por inculcarme esa hambre insaciable por la Bella Matemática y a la vez disfrutarla sin límites.

Al Guille por ser mi tutor, mi guía, mi profesor, pero sobre todo una figura paternal que siempre me ayudó en todo momento; mucho de lo que soy profesionalmente se lo debo a él.

A todos los demás profesores y amigos del laboratorio de Álgebra y Criptografía: Oristela, Perfetti, Yareida y Evaristo; por prestarme atención cuando la necesité y brindarme la posibilidad de estar junto a ellos durante estos cinco años.

También quiero agradecer a mis amigos pues sin ellos nunca habría logrado disfrutar de mi vida estudiantil, ni haber vivido todos esos momentos especiales.

A todos Muchas Gracias.

Resumen

Varios han sido los intentos de utilizar la Teoría de Grupos Combinatoria en la construcción de un algoritmo criptográfico de clave pública; sin embargo, esta rama de la Criptografía sigue dominada por la Teoría de Números y las Curvas Elípticas. En este trabajo el problema de las palabras en grupos es utilizado en la construcción de un algoritmo criptográfico de clave pública. Una forma específica de implementación es discutida, junto con los resultados experimentales recopilados. Este criptosistema es aleatorio, utiliza las relaciones de la presentación de un grupo para disfrazar los elementos del mismo grupo y es en parte inspirado por la idea de Wagner y Magyarik introducida en 1984.

Palabras Claves: Criptografía, Teoría de Grupos Combinatoria, Problema de las Palabras, Sistemas de Reescritura.

Abstract

There have been several attempts to use Combinatorial Group Theory in the construction of a public key cryptographic algorithm; however, this branch of cryptography is still dominated by the Number Theory and Elliptic Curves. In this paper the word problem in groups is used in the construction of a public key cryptographic algorithm. A specific form of implementation is discussed, together with the experimental results collected. This cryptosystem is randomized, uses the relationships of a group presentation to disguise the elements of the same group and is partly inspired by the idea of Wagner and Magyarik introduced in 1984.

Keywords: Cryptography, Combinatorial Group Theory, Word Problem, Rewriting Systems.

Tabla de contenidos

Introducción

Aun cuando las cartas no contienen información sensible o muy personal, a muchos de nosotros nos gusta pensar que el contenido de nuestra correspondencia personal es privado y que sellando el sobre lo protegemos de todos, excepto del destinatario intencional. Si nosotros enviáramos nuestras cartas con los sobres abiertos cualquiera podría leer su contenido. Para muchas personas el uso de correo electrónico es en la actualidad una alternativa para enviar las cartas a través del correo postal (Menezes, 1997); es unos de los medios más rápidos de comunicación pero, no hay ningún sobre para proteger los mensajes. De hecho se dice a menudo que enviar los mensajes mediante correo electrónico es como enviar una carta sin sobre. Claramente cualquiera que quiera enviar mensajes confidenciales, incluso personales, vía correo electrónico debe encontrar algún medio de protegerlos. Una solución común es usar la Criptografía y encriptar el mensaje. El uso de la encriptación para proteger los correos electrónicos no está extendido, pero se está extendiendo y es probable que esta proliferación continúe. De hecho en mayo del 2001 un grupo del bloque europeo recomendó a los usuarios de Europa encriptar todos sus correos electrónicos, para evitar ser espiados en lo adelante por el Reino Unido o EE.UU. que escuchan detrás de las puertas de la red.(Institute for Studies in Theoretical Physics and Mathematics, 2010)

La Criptografía es una ciencia bien establecida que ha tenido una influencia histórica significante por más de 2 000 años. Tradicionalmente sus usuarios principales eran los gobiernos y el ejército.

La Criptografía moderna ha evolucionado considerablemente durante las últimas tres décadas. No sólo ha cambiado la tecnología, sino que hay un rango más amplio de aplicaciones. Además, es probable que todos seamos usuarios directos o nos afectemos por su uso. Todos necesitamos entender cómo funciona y lo que se puede lograr.

La idea de un sistema cifrado es enmascarar la información confidencial de tal manera que su significado sea incompresible a una persona desautorizada. La mayoría de los

usos comunes es, probablemente, guardar los datos firmemente en un archivo de la computadora o transmitirlo por un canal inseguro como la Internet. En cualquier escenario el hecho que el documento se encripte no previene que las personas desautorizadas tengan acceso a él, pero asegura que no puedan entender lo que ellos ven.

Cualquier persona que intercepta un mensaje durante la transmisión se llama, no sorprendentemente, un interceptor. Otros autores usan diferentes nombres, como “el intruso”, “el enemigo”, “el adversario”, o incluso “el tipo malo”. Sin embargo, debe reconocerse que, en ocasiones, los interceptores pueden ser los “tipos buenos”. Aun cuando ellos saben el algoritmo de desencriptar, los interceptores no saben, en general, la llave de desencriptación. Es esta falta de conocimiento, la que impide saber el texto claro. La Criptografía es la ciencia de diseñar sistemas cifradores, considerando al Criptoanálisis como el proceso de deducir la información sobre el texto claro del texto cifrado sin conocerse la llave apropiada. Criptología es el término colectivo para Criptografía y Criptoanálisis.

La Criptografía Asimétrica es el método criptográfico que usa un par de claves para el envío de mensajes. Una clave es pública y se puede entregar a cualquier persona, la otra clave es privada y el propietario debe guardarla de modo que nadie tenga acceso a ella. La clave pública es utilizada para codificar el mensaje antes de ser enviado y la llave privada es utilizada para decodificar el mensaje por parte del receptor.

Dada la importancia de la Criptografía y su gran aplicabilidad en la sociedad contemporánea, este trabajo tiene la ambiciosa meta de desarrollar la teoría de un método criptográfico basado en conocimientos puramente algebraicos, para su posterior implementación.

Planteamiento del problema

Desde que se creó el Grupo de Álgebra y Criptografía en la Universidad Central “Marta Abreu” de Las Villas uno de los temas en los que se ha trabajado ha sido en la investigación de proponer un nuevo Sistema Criptográfico usando puramente el Álgebra. La respuesta a este problema de investigación, aparentemente de interés solamente

teórico, conduce directamente a la investigación de los procesos internos de muchos algoritmos criptográficos y está estrechamente vinculada con las herramientas y propiedades de la Teoría de Grupo en dichos procesos.

Varios criptosistemas de clave pública basados en Teoría de Grupos Combinatoria han sido propuestos desde la década de 1980, el primero de estos fue probablemente el esbozo de Wagner y Magyarik (Neal R. Wagner, 1985), (Vasco, 2003), (M.I. González Vasco, 2002). El trabajo de Wagner y Magyarik propuso un criptosistema de clave pública basado en *el problema de las palabras,* resultando ser muy ambiguo en su esquema general y estar lejos de una real implementación; sin embargo, esta propuesta es destacable por ser pionera en el campo (Birget et al., 2005).

La idea para un criptosistema de clave pública propuesta por Wagner y Magyarik en 1984, es un tema interesante para investigar. La idea general es muy imprecisa para ser llamada criptosistema, y es un reto interesante hacer estas ideas precisas y concisas para obtener un sistema seguro. Además, en (Shpilrain Vladimir, 2006) se plantea que disfrazar los elementos de un grupo mediante sus relaciones definitorias, parece ser la forma más prometedora que la Combinatoria de Grupos tenga impacto real en la Criptografía. De ahí que el objetivo de este trabajo es desarrollar un nuevo algoritmo criptográfico basado en la presentación de grupos, y de esta forma explorar las potencialidades de esta teoría en la construcción de criptosistemas de llaves públicas.

Objetivos del trabajo

Como objetivo general de este trabajo se propone:

Crear e implementar un nuevo algoritmo criptográfico de clave pública utilizando la presentación de grupos para su posterior estandarización como un método seguro de enviar información.

Con la finalidad de cumplir el objetivo general se proponen los siguientes objetivos específicos:

1. Exponer la teoría que sirve como sustento del nuevo método propuesto.

2. Implementar las funcionalidades del nuevo sistema criptográfico en una aplicación para su uso de forma segura, tanto por parte del emisor como del receptor.
3. Desarrollar representaciones de diferentes grupos finitos para que puedan ser utilizadas como claves en el sistema criptográfico.
4. Validar el algoritmo criptográfico utilizando algunas técnicas de Criptoanálisis.

Preguntas de investigación

1. ¿Cómo representar los grupos mediante sus generadores?
2. ¿Cómo implementar las funcionalidades del nuevo algoritmo criptográfico?
3. ¿De qué forma se puede realizar la validación del algoritmo criptográfico implementado?

Contribución del trabajo

Los principales aportes de este trabajo son desde el punto de vista teórico, pues se desarrolla un nuevo algoritmo criptográfico basado en la Teoría de Grupos Combinatoria. Después de hacer un análisis exhaustivo de la bibliografía sobre el tema, se llegó a la conclusión de que varios han sido los intentos de utilizar la Teoría de Grupos Combinatoria en la construcción de un algoritmo criptográfico de clave pública, pero sin embargo esta rama de la criptografía sigue dominada por la Teoría de Números y Curvas Elípticas, con lo que se justifica la importancia teórica de este trabajo. Después de ser implementado este algoritmo, pudiera ser utilizado como un nuevo método de enviar información de forma segura, dándole un valor de carácter práctico y con alcance social.

Capítulo 1

1 Fundamentos matemáticos y computacionales

Este capítulo está dedicado a realizar una descripción de las bases teóricas que sustentan el algoritmo criptográfico. Está estructurado por cuatro epígrafes para recopilar la teoría necesaria. El primer epígrafe está dedicado a la comprensión de algunas ideas básicas de la Criptografía, necesarias para la comprensión de las siguientes secciones del trabajo. El segundo epígrafe contiene fundamentos algebraicos, principalmente, sobre Teoría de Grupos, que es la piedra angular sobre la que se sustenta la mayor parte de la investigación. El tercer epígrafe trata contenidos un poco más avanzados del algebra, entre ellos: teoría de categorías, grupos libres, generadores y por último, la presentación de un grupo que es el concepto fundamental usado en este trabajo. El cuarto epígrafe presentará *los sistemas de reescrituras*, que serán los encargados de procesar en la práctica una presentación de grupo.

1.1 Introducción a los términos básicos de la Criptografía

Según el Diccionario de la Real Academia, la palabra Criptografía proviene de la unión de los términos griegos oculto y escritura, y sus definiciones: “Arte de escribir con clave secreta o de un modo enigmático”. Obviamente la Criptografía hace años que dejó de ser un arte para convertirse en una técnica, o más bien un conglomerado de técnicas, que tratan sobre la protección —ocultamiento frente a observadores no autorizados— de la información. Entre las disciplinas que engloba cabe destacar la Teoría de la Información, la Teoría de Números —o Matemática Discreta, que estudia las propiedades de los números enteros— y la Complejidad Algorítmica.

Existen dos trabajos fundamentales sobre los que se apoya prácticamente toda la teoría criptográfica actual. Uno de ellos, desarrollado por Claude Shannon en sus artículos “A Mathematical Theory of Communication” (1948) y “Communication Theory of Secrecy Systems” (1949), sienta las bases de la Teoría de la Información y de la Criptografía

Moderna. El segundo, publicado por Whitfield Diffie y Martin Hellman en 1976, se titulaba "New directions in Cryptography", e introducía el concepto de Criptografía Asimétrica, ampliando el abanico de aplicación de esta disciplina.

Conviene hacer notar que la palabra Criptografía sólo hace referencia al uso de códigos, por lo que no engloba las técnicas que se usan para romper dichos códigos, conocidas en su conjunto como Criptoanálisis. En cualquier caso ambas disciplinas están estrechamente vinculadas; no olvidemos que cuando se diseña un sistema para cifrar información, hay que tener muy presente su posible criptoanálisis, ya que en caso contrario podrían obtenerse resultados no deseados.

Finalmente, el término Criptología, aunque no está recogido aun en el Diccionario, se emplea habitualmente para agrupar Criptografía y Criptoanálisis.

1.1.1 Criptosistema

Definiremos un criptosistema como una quíntupla (M; C; K; E; D), donde (F. Rodríguez-Henríquez, 2006):

- M representa el conjunto de todos los mensajes sin cifrar (lo que se denomina texto claro, o *plaintext*) que pueden ser enviados.
- C representa el conjunto de todos los posibles mensajes cifrados, o criptogramas.
- K representa el conjunto de claves que se pueden emplear en el criptosistema.
- E es el conjunto de transformaciones de cifrado o familia de funciones que se aplica a cada elemento de M para obtener un elemento de C. Existe una transformación diferente E_k para cada valor posible de la clave k.
- D es el conjunto de transformaciones de descifrado, análogo a E.

Todo criptosistema ha de cumplir la siguiente condición:

$$D_k\big(E_k(m)\big) = m$$

Es decir, que si se tiene un mensaje m, se cifra empleando la clave k y luego se descifra empleando la misma clave, se obtiene de nuevo el mensaje original m.

Existen dos tipos fundamentales de criptosistemas:

- Criptosistemas simétricos o de clave privada. Son aquellos que emplean la misma clave k tanto para cifrar como para descifrar. Presentan el inconveniente de que para ser empleados en comunicaciones la clave k debe estar tanto en el emisor como en el receptor, lo cual nos lleva a la pregunta ¿cómo transmitir la clave de forma segura?
- Criptosistemas asimétricos o de llave pública, que emplean una doble clave (k_p;k_P). k_p se conoce como clave privada y $k_P$se conoce como clave pública. Una de ellas sirve para la transformación E de cifrado y la otra para la transformación D descifrado. En muchos casos son intercambiables, esto es, si se emplea una para cifrar la otra sirve para descifrar y viceversa. Estos criptosistemas deben cumplir, además, que el conocimiento de la clave pública k_P no permita calcular la clave privada k_p. Ofrecen un abanico superior de posibilidades, pudiendo emplearse para establecer comunicaciones seguras por canales inseguros — puesto que únicamente viaja por el canal la clave pública— o para llevar a cabo autentificaciones.

En la práctica se emplea una combinación de estos dos tipos de criptosistemas, puesto que los segundos presentan el inconveniente de ser computacionalmente mucho más costosos que los primeros. En el mundo real se codifican los mensajes (largos) mediante algoritmos simétricos, que suelen ser muy eficientes, y luego se hace uso de la Criptografía Asimétrica para codificar las claves simétricas (cortas).

1.1.2 Claves débiles

En la inmensa mayoría de los casos los conjuntos M y C definidos anteriormente son iguales. Esto quiere decir que tanto los textos claros como los textos cifrados se representan empleando el mismo alfabeto —por ejemplo, cuando se usa el algoritmo DES, ambos son cadenas de 64 bits—. Por esta razón puede darse la posibilidad de que exista $k \in K$ tal que $E_k(M) = M$, lo cual sería fatal para la seguridad de los textos cifrados, puesto que el empleo de esas claves dejaría todos los mensajes... ¡sin codificar!

También puede darse el caso de que ciertas claves concretas generen textos cifrados de poca calidad. Una posibilidad bastante común en ciertos algoritmos es que algunas claves tengan la siguiente propiedad: $E_k\big(E_k(M)\big) = M$ lo cual quiere decir que basta con

volver a codificar el criptograma para recupera el texto claro original. Estas circunstancias podrían llegar a simplificar enormemente un intento de violar el sistema, por lo que también deben ser evitadas.

La existencia de claves con estas características, como es natural, depende en gran medida de las peculiaridades de cada algoritmo en concreto y en muchos casos, también, de los parámetros escogidos a la hora de aplicarlo. Llamaremos en general a las claves que no codifican correctamente los mensajes claves débiles (*weak keys* en inglés). Normalmente en un buen criptosistema la cantidad de claves débiles es cero o muy pequeña en comparación con el número total de claves posibles. No obstante, conviene conocer esta circunstancia para poder evitar en la medida de lo posible sus consecuencias (Menezes, 1997).

1.1.3 Criptoanálisis

El criptoanálisis consiste en comprometer la seguridad de un criptosistema. Esto se puede hacer descifrando un mensaje sin conocer la llave, o bien obteniendo a partir de uno o más criptogramas la clave que ha sido empleada en su codificación. No se considera criptoanálisis el descubrimiento de un algoritmo secreto de cifrado; se debe suponer por el contrario que los algoritmos siempre son conocidos.

En general el criptoanálisis se suele llevar a cabo estudiando grandes cantidades de pares de mensajes –criptogramas generados con la misma clave–. El mecanismo que se emplee para obtenerlos es indiferente, y puede ser resultado de escuchar un canal de comunicaciones o de la posibilidad de que el objeto de nuestro ataque responda con un criptograma cuando le enviemos un mensaje. Obviamente, cuanto mayor sea la cantidad de pares, más probabilidades de éxito tendrá el criptoanálisis.

Uno de los tipos de análisis más interesantes es el de texto claro escogido, que parte de que se conocen una serie de pares de textos claros —elegidos por nosotros— y sus criptogramas correspondientes. Esta situación se suele dar cuando se tiene acceso al dispositivo de cifrado y este nos permite efectuar operaciones, pero no permite leer su clave —por ejemplo, las tarjetas de los teléfonos móviles GSM—. El número de pares

necesarios para obtener la clave desciende entonces significativamente. Cuando el sistema es débil, pueden ser suficientes unos cientos de mensajes para obtener información que permita deducir la clave empleada.

También se puede tratar de criptoanalizar un sistema aplicando el algoritmo de descifrado, con todas y cada una de las claves, a un mensaje codificado que se posee y comprobar cuáles de las salidas que se alcanzan tienen sentido como posible texto claro. En general, todas las técnicas que buscan exhaustivamente por el espacio de claves K se denominan de fuerza bruta, y no suelen considerarse como auténticas técnicas de Criptoanálisis, reservándose este término para aquellos mecanismos que explotan posibles debilidades intrínsecas en el algoritmo de cifrado. Generalmente, se denomina ataque a cualquier técnica que permita recuperar un mensaje cifrado empleando menos esfuerzo computacional que el que se usaría por la fuerza bruta. Se da por sentado que el espacio de claves para cualquier criptosistema digno de interés ha de ser suficientemente grande como para que los métodos basados en la fuerza bruta sean inviables. Hemos de tener en cuenta no obstante que la capacidad de cálculo de las computadoras crece a gran velocidad, por lo que algoritmos que hace unos años eran resistentes a la fuerza bruta hoy pueden resultar inseguros, como es el caso de DES, motivo por el cual fue sustituido por AES (NIST, 2001).

Como se puede apreciar, la gran variedad de sistemas criptográficos produce necesariamente gran variedad de técnicas de Criptoanálisis, cada una de ellas adaptada a un algoritmo o familia de ellos. Con toda seguridad, cuando en el futuro aparezcan nuevos mecanismos de protección de la información, surgirán con ellos nuevos métodos de Criptoanálisis. De hecho, la investigación en este campo es tan importante como el desarrollo de algoritmos criptográficos, y esto es debido a que mientras que la presencia de fallos en un sistema es posible demostrarla, su ausencia es por definición indemostrable.

1.2 Teoría de grupos

La mayor parte del contenido a exponer en este epígrafe se encuentra en (Herstein, 1996), (Hungerford, 2000) y (Lang, 2000). En estos libros se pueden encontrar las definiciones, teoremas y propiedades que se enuncian, igualmente se pueden encontrar las demostraciones.

1.2.1 Semigrupos, Monoides y Grupos

Si G es un conjunto no vacío, una operación binaria en G es una función $G \times G \to G$. Hay varias notaciones comúnmente usadas para la imagen de (a, b) bajo la operación binaria: ab (notación multiplicativa), $a + b$ (notación aditiva), $a \cdot b, a * b,$ etc. Por conveniencia generalmente se usará la notación multiplicativa durante el texto y se referirá ab como el producto de a y b.

Definición 1.1 Un semigrupo es un conjunto no vacío G junto con una operación binaria en G que cumple

i) Asociatividad: $a(bc) = (ab)c, \ \ para\ todo\ a, b, c \in G$.

Un monoide es un semigrupo G que contiene

ii) Un elemento neutro $e \in G$, tal que $ae = ea = a, para\ todo\ a \in G$.

Un grupo es un monoide G tal que

iii) Para todo elemento $a \in G$, existe un elemento inverso $a^{-1} \in G$ tal que $aa^{-1} = a^{-1}a = e$.

Un semigrupo G es llamado *abeliano* o *conmutativo* si su operación binaria es

iv) Conmutativa: $ab = ba, para\ todo\ a, b \in G$.

El principal interés de este trabajo es los grupos. Además, semigrupos y monoides son convenientes para poner algunos teoremas de una forma más general. El *orden* de un grupo G es el número cardinal de este y se representa por $|G|$.

Teorema 1.2 Si G es un monoide, entonces el elemento neutro es único. Si G es un grupo, entonces

i) $c \in G$ y $cc = c \Rightarrow c = e$;

ii) para todo $a, b, c \in G$ $ab = ac \Rightarrow b = c$ y $ba = ca \Rightarrow b = c$ (cancelación derecha e izquierda);

iii) para cada $a \in G$, el elemento inverso a^{-1} es único;

iv) para cada $a \in G$, $(a^{-1})^{-1} = a$;

v) para $a, b \in G, (ab)^{-1} = b^{-1}a^{-1}$;

vi) para $a, b \in G$ las ecuaciones $ax = b$ y $ya = b$ tienen soluciones únicas en G: $x = a^{-1}b$ y $y = ba^{-1}$.

Definición 1.3 Un grupo *G* es llamado cíclico, si existe un elemento $a \in G$, tal que todo elemento $x \in G$ se puede representar como una potencia de *a*. Se puede denotar como $G =< a >$.

Ejemplos: Los enteros *Z,* los números racionales *Q* y los números reales *R* son cada uno grupos abelianos infinitos bajo la operación de adición ordinaria. Estos son monoides bajo la operación de multiplicación, pero no grupos (el 0 no tiene inverso). Sin embargo, los elementos diferentes del 0 de *Q* y *R* respectivamente forman un grupo abeliano bajo la operación de multiplicación. Los enteros pares forman un semigrupo que no es monoide bajo la operación de multiplicación.

1.2.2 Subgrupos

Definición 1.3 Un subconjunto no vacío, *H,* de *G* es llamado un *subgrupo* de *G*, si relativo a la operación de *G*, *H* por sí mismo es un grupo.

Todo grupo *G* tiene automáticamente dos subgrupos llamados *subgrupos triviales,* el mismo grupo *G* y el formado solamente por el elemento neutro *e*. De mayor interés son los demás subgrupos llamados *subgrupos propios*.

Lema 1.4 Un subconjunto $H \subset G$ es un subgrupo de *G* si y solo si *H* es cerrado respecto a la operación de *G*, y para todo $a \in H$ se cumple que $a^{-1} \in H$.

Muchas son las reducciones y resultados que se pueden obtener cuando el subgrupo *H* sea finito.

Lema 1.5 Supón que *G* es un grupo y que H es un subconjunto finito no vacío, cerrado bajo la operación de *G.* Entonces *H* es un subgrupo de *G*.

Corolario 1.6 Si *G* es un grupo finito y *H* es un subconjunto no vacío de *G* cerrado bajo la operación *G*, entonces *H* es un subgrupo de *G.*

Justo como el concepto de *función* está presente en la mayoría de las ramas de la matemática, algo similar ocurre con el concepto de relación.

Definición 1.7 Una relación *R* en un conjunto *S* es llamada una *relación de equivalencia* si para todo $a, b, c \in S$ se cumple:

a) reflexividad: $a\ R\ a$
b) simetría: $aRb \Rightarrow bRa$
c) transitividad: $aRb \wedge bRc \Rightarrow aRc$

Las relaciones de equivalencia son capaces de agrupar los elementos de un conjunto de una forma natural, llamada *clase de equivalencia.*

Definición 1.8 Si *R* es una relación de equivalencia en un conjunto *S*, entonces $\bar{a}$, la clase de equivalencia de *a*, se define como el conjunto de elementos relacionados con *a* mediante la relación *R*.

Las relaciones de equivalencia tienen la importante influencia sobre los conjuntos de romperlos y realizar una *partición* sobre estos, en subconjuntos disjuntos.

Teorema 1.9 Si *R* es una relación de equivalencia sobre el conjunto *S*, entonces $S = \cup\, \bar{a}$ y donde $\bar{a} \neq \bar{b}$ implica que $\bar{a} \cap \bar{b} = \emptyset$. Esto es, *R* particiona al conjunto *S* en clases de equivalencia.

Dado un grupo *G* y un subgrupo *H* de *G*. Se puede definir la relación de equivalencia $aRb \Leftrightarrow ab^{-1} \in H$. Entonces aRb si $ab^{-1} \in H$, esto es, si $ab^{-1} = h$, para algún $h \in H$. Entonces aRb implica que $a = hb$. Por el otro lado, si $a = kb$ donde $k \in H$, implica que

$ab^{-1} = kbb^{-1} \Rightarrow ab^{-1} = k \Rightarrow ab^{-1} \in H \Rightarrow aRb$. Por tanto aRb si y solo si $a \in Hb = \{hb: h \in H\}$. Entonces $\bar{b} = Hb$, según la relación inducida por el subgrupo H.

El conjunto Hb es llamado *coseto derecho* y similarmente se define el *coseto izquierdo* $bH = \{bh: h \in H\}$ que también genera una relación de equivalencia de la misma forma que el otro. Estos dos conjuntos forman un papel clave a lo largo del trabajo.

Ahora están las condiciones para exponer un famoso resultado de Lagrange que trasciende al álgebra y la teoría de números.

Teorema 1.10 (Lagrange) Si *G* es un grupo finito y *H* es un subgrupo de *G*, entonces el orden de *H* divide al orden de *G.*

Si *G* es finito, el número de cosetos derechos de *H* en *G*, llamado $|G|/|H|$, es el *índice* de *H* en *G* y es escrito como $i_G(H)$.

Teorema 1.11 Un grupo *G* de orden primo es un grupo cíclico.

Este teorema tiene la gran implicación de que existe un único grupo con orden primo *p* *($|G| = p$),* salvo isomorfismos (concepto que se explicará en la próxima sección).

Definición 1.12 Si *G* es finito, el *orden de a*, escrito como $o(a)$, es el menor entero positivo *m* tal que $a^m = e$.

Supón que $a \in G$ tiene orden *m.* El conjunto $A = \{e, a, a^2, \dots, a^{m-1}\}$ es un subgrupo de *G* y los *m* elementos listados en *A* son diferentes. Entonces $|A| = m = o(a)$. Como $|A| \mid |G|$ se tiene:

Teorema 1.13 Si *G* es finito y $a \in G$, entonces $o(a) \mid |G|$.

Por lo tanto hemos probado:

Teorema 1.14 Si *G* es un grupo finito de orden *n*, entonces $a^n = e$ para todo $a \in G$.

Cuando aplicamos este teorema a algunos grupos especiales presentes en la *teoría de números (Moser, 2004),* se pueden obtener algunos resultados clásicos los de *Euler* y de *Fermat (Vinogradov, 1954) y (Castillo, 1999)*.

1.2.3 Homomorfismos y subgrupos normales

En cierto sentido la teoría de grupos es construida sobre tres conceptos básicos: homomorfismos, subgrupo normal y grupo factor o cociente por un subgrupo normal de un grupo (Mullen and Panario, 2013).

Definición 1.15 Sean *G* y *H* dos grupos; entonces la aplicación $\varphi: G \to H$ es un homomorfismo si $\varphi(ab) = \varphi(a)\varphi(b)$ para todo $a, b \in G$.

En esta definición el producto de la izquierda —en $\varphi(ab)$ —, es en *G* mientras el producto de la derecha $\varphi(a)\varphi(b)$ es en H. Una corta descripción es que el homomorfismo preservar la operación de los grupos.

La siguiente construcción general conlleva a la demostración del muy conocido teorema de *Cayley*. Sea *G* cualquier grupo y $S(G)$ el *grupo de permutaciones de G*, el cual contiene a todas las biyecciones de *G* en *G* y como operación la composición de funciones (Castillo, 2001). Se define $T_a: G \to G$ por $T_a(x) = ax$ para cada $x \in G$. El producto de T_a y T_b se define como:

$$(T_a T_b)(x) = T_a\big(T_b(x)\big) = a(bx) = (ab)x = T_{ab}$$

Entonces se tiene que $T_a T_b = T_{ab}$.

Al definir la aplicación $\varphi: G \to S(G)$ por $\varphi(a) = T_a$ para todo $a \in G$. Se tiene que $\varphi(ab) = T_{ab} = T_a T_b = \varphi(a)\varphi(b)$, por tanto φ es un homomorfismo de *G* en $S(G)$. Tenemos que φ es inyectiva pues $\varphi(a) = \varphi(b) \Rightarrow T_a = T_b \Rightarrow ax = bx \; para \; todo \; x \in G \Rightarrow a = b$. No es sobreyectiva en general pues *G* tiene orden *n* y $S(G)$ orden $n!$ y $n! > n$ para $n > 2$. Es fácil verificar que la imagen de φ, definida como $\varphi(G) = \{T_a: a \in G\}$, es un subgrupo de $S(G)$.

Definición 1.16 El homomorfismo $\varphi: G \to H$ es llamado *monomorfismo* si φ es inyectiva. Un *homomorfismo* que sea a su vez sobreyectivo es llamado *isomorfismo*. Un isomorfismo de *G* a *G* es llamado *automorfismo*.

Definición 1.17 Dos grupos *G* y *H* son isomorfos si existe un isomorfismo de *G* a *H*.

Denotaremos *G* isomorfo a *H*, escribiéndolo de la forma $G \simeq H$.

Ahora están las condiciones para presentar el teorema de *Cayley*, de gran importancia teórica.

Teorema 1.18 Todo grupo finito *G* es isomorfo a algún subgrupo de $S(A)$, para algún *A* escogido.

El conjunto escogido para *A* puede ser el *G,* ejemplo para el cual se demostró anteriormente. Pero pueden existir mejores opciones para *A.*

Este es un buen lugar para discutir la importancia de un *"isomorfismo".* Sea φ un isomorfismo de *G* en *H.* Podemos ver H como una sustitución idéntica de *G*, utilizando $\varphi(x)$ para un valor x. Como $\varphi(xy) = \varphi(x)\varphi(y)$, vemos que xy es sustituido por $\varphi(x)\varphi(y)$, entonces el renombrado que se hace de los elementos es consistente con el producto en *G*. Por tanto, dos grupos que sean isomorfos, en cierto sentido son iguales, como se describió anteriormente. Muchas veces es deseable encontrar un isomorfismo entre un grupo y un grupo del cual ya se conocen sus características.

Es tiempo de hacer una pequeña investigación sobre los homomorfismos.

Lema 1.19 Si φ es un homomorfismo de *G* en *H* entonces:

a) $\varphi(e) = e'$, es el elemento neutro de H.
b) $\varphi(a^{-1}) = \varphi(a)^{-1}$, para todo $a \in G$.

Definición 1.20 La imagen φ, es $\varphi(G) = \{\varphi(a): a \in G\}$.

Lema 1.21 Si φ es un homomorfismo de G en H, entonces la imagen de φ, es un subgrupo de H.

Si se quiere medir cuán lejos está un *homomorfismo* de ser un *monomorfismo*. Esto nos acerca a

Definición 1.22 Si φ es un homomorfismo de G en H, entonces el *núcleo* de φ, se define por $Ker\ \varphi = \{a \in G: \varphi(a) = e'\}$.

El núcleo $Ker\ \varphi$ mide claramente la falta de inyectividad en el punto e'. Asombrosamente esto se generaliza uniformemente para todos los puntos de la imagen.

Lema 1.23 Si $w \in H$ es de forma $\varphi(x) = w$, entonces $\{y \in G: \varphi(y) = w\} = (Ker\ \varphi)\ x$.

Ahora se pueden estudiar algunas propiedades básicas de los núcleos de los homomorfismos.

Teorema 1.24 Si φ es un homomorfismo de *G* en *H*, entonces

a) $Ker\ \varphi$ es un subgrupo de *G.*
b) Dado $a \in G$, se tiene $a^{-1}(Ker\ \varphi)a \subset Ker\ \varphi$.

Corolario 1.25 Si φ es un homomorfismo de *G* en *H*, entonces φ es un monomomorfismo si y sólo si $Ker\ \varphi = \{e\}$.

La propiedad *b)* $Ker\ \varphi$ en el *Teorema 1.24* es interesante para un subgrupo. Se usará para definir una ultra-importante clase de subgrupos de un grupo.

Definición 1.26 El subgrupo *N* de *G* es llamado un *subgrupo normal* de *G* si $a^{-1}Na \subset N$ para todo $a \in G$.

Por supuesto, $Ker\ \varphi$ para cualquier homomorfismo es un subgrupo normal de *G*.

Aunque se define un subgrupo normal vía $a^{-1}Na \subset N$, realmente se tiene $a^{-1}Na = N$. Porque si $a^{-1}Na \subset N$ para todo $a \in G$, entonces $N = a(a^{-1}Na)a^{-1} \subset aNa^{-1} \subset (a^{-1})^{-1}Na^{-1} \subset N$. Entonces $a^{-1}Na = N$ para todo $a \in G$. Trasponiendo, se obtiene $aN = Na$; esto es, todo coseto izquierdo de *N* en *G* es un coseto derecho de *N* en *G*.

Por otro lado, si todo coseto izquierdo de N en G es un coseto derecho, entonces el coseto izquierdo aN, que contiene a, debe ser igual al coseto derecho que contiene a, denotado como Na. Entonces $aN = Na$ y $a^{-1}Na = N$ para todo $a \in G$, que quiere decir que *N* es normal en *G*.

Se escribe *"N es un subgrupo normal de G"* por el símbolo abreviado $N \lhd G$.

Teorema 1.27 $N \lhd G$ si y solo si todo coseto izquierdo de *N* en *G* es un coseto derecho de *N* en *G*.

1.2.4 Grupo cociente

En la demostración del teorema de Lagrange, se usa para un subgrupo arbitrario *H* la relación de equivalencia aRb si $ab^{-1} \in H$. Trataremos esto cuando el subgrupo es normal y veremos si podemos decir algo más que con el viejo subgrupo arbitrario.

Entonces, sea aRb si $ab^{-1} \in N$ y $\bar{a} = \{x \in G : xRa\}$. Como se vio anteriormente $\bar{a} = aN$, el coseto derecho que contiene *a,* pero en el caso de un subgrupo normal también coincide con el coseto izquierdo aN.

Sea $M = \{\bar{a} : a \in G\}$, donde se define un producto vía $\overline{ab} = \bar{a}\bar{b}$. Se demostrará que *M* es un grupo bajo este producto. Pero primero que todo se debe demostrar que este producto en *M* está bien definido; que *el producto de clases no depende de los representantes de cada clase*, o equivalentemente que *la relación R sea compatible con la operación de G.*

Si se supone que $\bar{a} = \overline{a'}$ y $\bar{b} = \overline{b'}$. Mediante la definición de la relación de equivalencia, se obtiene $a' = na$, donde $n \in N$; similarmente $b' = mb$, donde $m \in N$. Entonces $a'b' = namb = n(ana^{-1})ab$, por $N \lhd G$ se tiene $ana^{-1} \in N$ y por tanto $n_1 = nana^{-1} \in N$. Entonces, $n_1 \in G$ y $a'b' = n_1ab$. Esto nos dice $a'b' \in Nab$, entonces $a'b'Rab$, de donde se obtiene que $\overline{a'b'} = \overline{ab}$, justo lo que se necesitaba para que el producto en *M* estuviera bien definido.

Ahora se verifican los axiomas para que *M* sea un grupo. La clausura se obtiene de la misma definición del producto. Si $\bar{a}, \bar{b}, \bar{c} \in M$, entonces $(\bar{a}\bar{b})\bar{c} = \overline{ab}\bar{c} = \overline{abc} = \overline{a(bc)} = \bar{a}\overline{bc} = \bar{a}(\overline{bc})$, con lo que se demuestra la asociatividad. El elemento neutro es $\bar{e}$, pues $\bar{e}\bar{a} = \overline{ea} = \bar{a} = \overline{ae} = \bar{a}\bar{e}$. Para $\bar{a}$, tenemos como inverso $\overline{a^{-1}}$, debido a que $\bar{a}\overline{a^{-1}} = \overline{aa^{-1}} = \bar{e} = \overline{a^{-1}a} = \overline{a^{-1}}\bar{a}$.

Se quiere darle un nombre a *M* y mucho mejor un símbolo que denote la dependencia de *N* y *G*. El símbolo que se usará para *M* será G/N y es llamado *grupo cociente o grupo factor* de *G* por *N*.

Teorema 1.28 Si $N \lhd G$ y $G/N = \{\bar{a}: a \in G\} = \{Na: a \in G\}$, entonces G/N es un grupo relativo a la operación $\overline{ab} = \bar{a}\bar{b}$.

Una observación que debe ser hecha inmediatamente es

Teorema 1.29 Si $N \lhd G$, entonces existe un homomorfismo ψ de G en G/N, tal que $Ker\,\psi = N$.

Este homomorfismo se define como $\psi(a) = \bar{a} = Na$.

Este último teorema, trae el notable resultado: *todo homomorfismo tiene como núcleo un grupo normal y todo grupo normal es núcleo de algún homomorfismo.*

Cuando *G* es un grupo finito y $N \lhd G$, entonces por el teorema de Lagrange se tiene que el número de cosetos derechos es $i_G(N) = |G|/|N|$. Más precisamente se puede definir como

Teorema 1.30 Si G es un grupo finito y $N \lhd G$, entonces $|G/N| = |G|/|N|$.

La noción de grupo cociente es muy sutil y una de las más grandes en la materia. La construcción de un nuevo conjunto, desde un conjunto viejo utilizando como elementos de este nuevo conjunto subconjuntos del conjunto viejo, es extraño para un aficionado verlo por primera vez.

1.2.5 Los teoremas de homomorfismo

Sea G un grupo y φ un homomorfismo de G en G'. Si *K* es el núcleo de φ, entonces *K* es un subgrupo normal de *G* y se puede formar el grupo cociente G/K. Es completamente

natural esperar que exista una relación entre G/K y G'. El *Primer Teorema de Homomorfismo* explica esta relación en detalle.

Teorema 1.31 (Primer Teorema de Homomorfismo) Sea φ un homomorfismo de G en G' con núcleo *K*. Entonces $G/K \simeq G'$, el isomorfismo entre estos dos grupos puede ser efectuado por la aplicación $\psi: G/K \to G'$ definida por $\psi(aK) = \varphi(a)$.

El siguiente resultado, es una extensión del *Primer Teorema de Homomorfismo* y es llamado tradicionalmente *Teorema de Correspondencia.*

Teorema 1.32 (Teorema de Correspondencia) Sea $\varphi: G \to G'$ un homomorfismo de G en G' con núcleo *K.* Si H' es un subgrupo de G' y si $H = \{a: \varphi(a) \in H'\}$, entonces *H* es un subgrupo de *G,* $H \supset K$ *y* $H/K \simeq H'$. Finalmente, si $H' \lhd G'$, entonces $H \lhd G$.

Es valioso notar que si *K* es cualquier subgrupo normal de *G* y φ es el homomorfismo natural de G en G/K, entonces el teorema nos da una correspondencia biunívoca entre todos los subgrupos H' de G/K y aquellos subgrupos de G que contienen *K*. Además, esta correspondencia preserva la normalidad en el sentido de H' es normal en G/K si y sólo si H es normal en G.

Ahora se propone el *Segundo Teorema de Homomorfismo.*

Teorema 1.33 (Segundo Teorema de Homomorfismo) Sea *H* un subgrupo de *G* y sea *N* un subgrupo normal de *G*. Entonces $HN = \{hn: h \in H, n \in N\}$ es un subgrupo de *G*, $H \cap N$ es un subgrupo normal de *H*, y $H/(H \cap N) \simeq (HN)/N$.

Finalmente el *Tercer Teorema de Homomorfismo.*

Teorema 1.34 (Tercer Teorema de Homomorfismo) Si el isomorfismo $\varphi: G \to G'$ es un homomorfismo de G en G' con núcleo *K* entonces, si $N' \lhd G'$ y $N = \{a \in G: \varphi(a) \in N'\}$, concluimos que $G/N \simeq G'/N'$. Equivalente a, $G/N \simeq (G/K)/(N/K)$ de forma más elegante.

Esta última igualdad es realmente sugerente, es como si *se cancelara* el subgrupo *K* del denominador y el numerador.

1.3 La estructura de grupos

En este epígrafe se tratan contenidos de un nivel de abstracción un poco superior a los presentados en el epígrafe anterior. Basado en los fundamentos de la teoría de grupos expuestos previamente, se podrá definir de forma rigurosa la presentación de un grupo, las definiciones, teoremas y demostraciones aquí utilizadas puede ser encontradas explícitamente (Hungerford, 2000) y (Castillo, 2005); también de una forma más sutil en (Castillo, 2002).

1.3.1 Teoría de categorías

Es un momento apropiado para introducir el concepto de categoría. Categorías servirán como un útil lenguaje y proveerán un contexto general para tratar diferentes situaciones matemáticas.

La idea intuitiva bajo la definición de una categoría es la de varios objetos matemáticos ya introducidos (conjuntos, grupos, monoides) junto con los mapeos apropiados para estos objetos (funciones para conjuntos, homomorfismos para grupos) tienen un número de propiedades formales en común. Estas nociones son formalizadas en:

Definición 1.35 Una categoría es una clase $\mathcal{C}$ de objetos (denotada *A, B, C…*) junto con

i) Una clase de conjuntos disjuntos, denotada como *hom(A, B),* una para cada par de objetos en $\mathcal{C}$ (un elemento *f* de *hom(A, B)* es llamado un morfismo de *A* en *B* y es denotado por $f: A \to B$).

ii) Para cada tripleta *(A, B, C)* de objetos de $\mathcal{C}$ un función

$$\hom(B, C) \times \hom(A, B) \to \hom(A, C)$$

para morfismo $f: A \to B$ y $g: B \to C$, esta función es escrita $g \circ f: A \to C$ y llamada **compuesta de f y g.** La cual está sujeta a los dos axiomas:

I. (Asociatividad) Si $f: A \to B$, $g: B \to C$ y $h: C \to D$ son morfismos de $\mathcal{C}$ entonces $h \circ (g \circ f) = (h \circ g) \circ f$.

II. (Identidad) Para cada objeto *B* de $\mathcal{C}$, existe un morfismo $1_B: B \to B$, tal que para cualquier $f: A \to B$ y $g: B \to C$,

$$f \circ 1_B = f, \quad 1_B \circ g = g$$

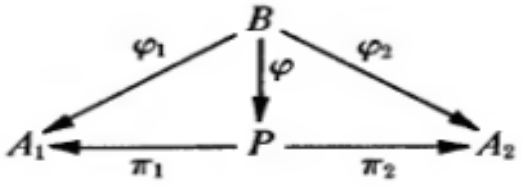
En una categoría $\mathcal{C}$ un morfismo $f: A \to B$ es llamado una **equivalencia** si existe en $\mathcal{C}$ un morfismo $g: B \to A$ tal que $g \circ f = 1_A$ y $f \circ g = 1_B$. La composición de dos equivalencias es una equivalencia. Si $f: A \to B$ es una equivalencia, entonces *A* y *B* se dice que son **equivalentes**.

Ejemplo Sea $\mathcal{S}$ la clase de todos los conjuntos; para $A, B \in \mathcal{S}$, *hom(A, B)* es el conjunto de todas las funcionas $f: A \to B$. Entonces $\mathcal{S}$ puede ser vista fácilmente como una categoría. Un morfismo f de $\mathcal{S}$ es una equivalencia si y solo si es una biyección.

Ejemplo Sea $\mathcal{G}$ la categoría cuyos objetos son los grupos; *hom(A, B)* es el conjunto de todos los homomorfismos $f: A \to B$. Un morfismo f es una equivalencia si y solo si f es isomorfismo. La categoría $\mathcal{A}$ de los grupos abelianos puede ser definida de igual forma.

Definición 1.36 Sea $\mathcal{C}$ una categoría y $\{A_i: i \in I\}$ una familia de objetos de $\mathcal{C}$. Un **producto** para la familia $\{A_i: i \in I\}$ es un objeto P perteneciente a $\mathcal{C}$ junto con una familia de morfismos $\{\pi_i: P \to A_i: i \in I\}$ tal que para cualquier objeto B y familia de morfismos $\{\varphi_i: B \to A_i: i \in I\}$, existe un único morfismo $\varphi: B \to P$ tal que $\varphi_i \circ \varphi = \pi_i$ para todo $i \in I$.

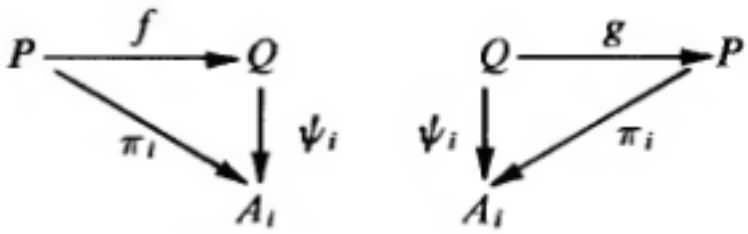
Un producto P de $\{A_i: i \in I\}$ es usualmente denotado como $\prod_{i \in I} A_i$. Algunas veces útil describir un producto en términos de diagramas conmutativos, especialmente para el caso $I = \{1,2\}$. Un producto para $\{A_1, A_2\}$ es un diagrama (de objetos y morfismos) $A_1 \overset{\pi_1}{\leftarrow} P \overset{\pi_2}{\rightarrow} A_2$ tal que: para cualquier diagrama de la forma $A_1 \overset{\varphi_1}{\leftarrow} B \overset{\varphi_2}{\rightarrow} A_2$, existe un único morfismo $\varphi: B \to P$ tal que el siguiente diagrama es conmutativo:



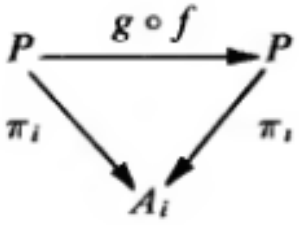
En la categoría de los conjuntos el producto Cartesiano $\prod_{i\in I} A_i$ es un producto para la familia de conjuntos $\{A_i : i \in I\}$.

Teorema 1.37 Si (P, φ_i) y (Q, ψ_i) son ambos productos de la familia $\{A_i : i \in I\}$ de objetos de una categoría $\mathcal{C}$, entonces P y Q son equivalentes.

Demostración: Como P y Q son ambos productos, existen los morfismos $f : P \to Q$ y $g : Q \to P$ tal que los siguientes diagramas son conmutativos para cada $i \in I$:



Componiendo estos dos diagramas para cada $i \in I$ se obtiene el siguiente diagrama conmutativo:



Entonces $g \circ f : P \to P$ es un morfismo tal que $\pi_i \circ (g \circ f) = \pi_i$, para todo $i \in I$. Pero por la definición de producto existe un único morfismo con esta propiedad. Como el mapeo $1_P : P \to P$ también cumple la propiedad $\pi_i \circ 1_P = \pi_i$ para todo $i \in I$, entonces se tiene $g \circ f = 1_P$ por unicidad. Similarmente, usando el hecho de que Q es un producto, se

muestra que $f \circ g = 1_Q$. Entonces $f: P \to Q$ es una equivalencia, con lo que queda demostrado el teorema■.

Definición 1.38 Un **categoría concreta** es una categoría $\mathcal{C}$ junto con una función σ que asigna a cada objeto *A* de $\mathcal{C}$ un conjunto $\sigma(A)$ (llamado conjunto subyacente de *A*) en la siguiente forma:

i) Todo morfismo $A \to B$ de $\mathcal{C}$ es una función en los conjuntos subyacentes $\sigma(A) \to \sigma(B)$.

ii) El homomorfismo identidad para cada objeto *A* de $\mathcal{C}$ es la función identidad en el conjunto subyacente $\sigma(A)$.

iii) La composición de morfismos en $\mathcal{C}$ coincide con la composición de funciones en los conjuntos subyacentes.

Ejemplo La categoría de grupos, equipado con la función que asigna a cada grupo su conjunto subyacente en el sentido usual, es una categoría concreta. Similarmente las categorías de grupos abelianos y conjuntos parcialmente ordenados, con sus conjuntos subyacentes obvios, son categorías concretas.

Categorías concretas son útiles frecuentemente ya que no solo tienes disponible las propiedades de una categoría, si no también ciertas propiedades de conjuntos, subconjuntos, etc. Como en toda categoría virtual el principal interés es en la función σ que asigna a un objeto su conjunto subyacente en el sentido usual (como en el ejemplo anterior), se denotarán ambos el objeto y su conjunto subyacente por el mismo símbolo y omitiremos cualquier referencia explícita a σ.

Definición 1.39 Sea F un objeto en una categoría concreta $\mathcal{C}$, *X* un conjunto no vacío, y $i: X \to F$ un mapeo (de conjuntos). *F* es llamado **objeto libre del conjunto X** si para cualquier objeto *A* de $\mathcal{C}$ y mapeo (de conjuntos) $f: X \to A$, existe un único morfismo de $\mathcal{C}$, definido $\bar{f}: F \to A$, tal que $\bar{f} i = f$ (como un mapa de conjuntos $X \to A$).

El hecho esencial acerca de un objeto libre *F* es para definir un morfismo con dominio *F*, es suficiente con especificar la imagen del subconjunto $i(X)$ como puede ser visto en el siguiente ejemplo.

Ejemplo Sea *G* cualquier grupo y $g \epsilon G$. Entonces el mapeo $\bar{f}: Z \to G$ definido por $\bar{f}(n) = g^n$ es fácilmente ver que es el único homomorfismo $Z \to G$ tal que $1 \vdash g$. Consecuentemente, si $X = \{1\}$ y $i: X \to Z$ es el mapeo de inclusión, entonces *Z* es libre sobre *X* en la categoría de grupos (dado $f: X \to G$, sea $g = f(1)$ y define $\bar{f}$ como fue previamente). En otras palabras, para determinar un único homomorfismo de *Z* a *G* solo es necesario especificar la imagen de $1 \in Z$ (que es, la imagen de $i(X)$).

Como se ha visto la teoría de categorías es una poderosa herramienta que permite niveles de abstracción incluso superior a la de un conjunto. Lo cual brinda una forma fácil y a la vez rigurosa de definir un objeto libre y sus principales características. Esto será particularmente útil para luego introducir el concepto de grupo libre sobre otro grupo en la categoría de los grupos, este trabalenguas sería imposible entenderlo sin una clara definición de los conceptos anteriores.

1.3.2 Grupos libres, generadores y relaciones

Se debe demostrar que objetos libres (grupos libres) existen en la categoría (concreta) de grupos, y se usará para desarrollar un método de describir grupos en términos de "generadores y relaciones". Estos conceptos están enmarcados dentro de la Teoría de Grupos Combinatoria (Wilhelm Magnus, 1966) y (Roger C. Lyndon, 1977)

Dado un conjunto *X* se construye un grupo *F* que es libre en el conjunto *X* en el sentido de la **Definición 1.39.** Si $X = \emptyset$, *F* es el grupo trivial $< e >$. Si $X \neq \emptyset$, sea X^{-1} un conjunto disjunto a *X* tal que $|X| = |X^{-1}|$. Se escoge una biyección $X \to X^{-1}$ y se denota la imagen de $x \in X$ por x^{-1}. Finalmente se escoge un conjunto que es disjunto de $X \cup X^{-1}$ y contiene un solo elemento denotado por 1 (neutro). Una **palabra** en *X* es una secuencia $(a_1, a_2, a_3, \dots)$ con $a_i \in X \cup X^{-1} \cup \{1\}$ tal que para algún $n \in N$, se cumple $a_k = 1$ para todo $k \geq n$. La secuencia constante $(1,1, \dots)$ es llamada la **palabra vacía** y es denotada

por 1. (Esta notación ambigua no causará confusión)**.** Una palabra $(a_1, a_2, a_3, \dots)$ en *X* es **reducida** cuando:

i) Para todo $x \in X$, los elementos x y x^{-1} no son adyacentes (esto es, $a_i = x \Rightarrow a_{i+1} \neq x^{-1}$ para toda $i \in N$ y $x \in X$).

ii) $a_k = 1$ implica que $a_i = 1$ para toda $i \geq k$.

En particular la palabra 1 es reducida.

Toda palabra no vacía y reducida es de la forma $({x_1}^{\lambda_1}, {x_2}^{\lambda_2}, \dots, {x_n}^{\lambda_n}, \dots)$, donde $n \in N, x \in X \; y \; \lambda_i = \mp 1$ (por convención x^1 denota para todo $x \in X$). De ahora en adelante se denotará esta palabra por ${x_1}^{\lambda_1}, {x_2}^{\lambda_2}, \dots, {x_n}^{\lambda_n}$. Esta nueva notación es ambas más tratable y sugerente. Observe que la definición de igualdad de las secuencias muestra que dos palabras reducidas ${x_1}^{\lambda_1}, {x_2}^{\lambda_2}, \dots, {x_m}^{\lambda_m}$ y ${y_1}^{\delta_1}, {y_2}^{\delta_2}, \dots, {y_n}^{\delta_n}$, con $x_i, y_i \in X \; y \; \lambda_i, \delta_i = \mp 1$ son iguales si y solo si ambas son *1* o $m = n$ y $x_i, = y_i, \lambda_i = \delta_i$ para cada $i = 1,2, \dots, n$. Consecuentemente el mapeo de *X* dentro del conjunto $F(X)$ de todas las palabras reducidas en *X* dado por $x \vdash x^1 = x$ es inyectivo. Identificando *X* con su imagen y considerando *X* como un subconjunto de $F(X)$.

Luego se define una operación binaria en el conjunto $F = F(X)$ de todas las palabras reducidas en *X*. La palabra vacía actúa como elemento identidad ($w1 = 1w = w$ para toda $w \in F$). Informalmente, se quiere un producto de palabras no vacías y reducidas definido por yuxtaposición, esto es:

$$\left({x_1}^{\lambda_1}, {x_2}^{\lambda_2}, \dots, {x_m}^{\lambda_m}\right)\left({y_1}^{\delta_1}, {y_2}^{\delta_2}, \dots, {y_n}^{\delta_n}\right) = {x_1}^{\lambda_1}, {x_2}^{\lambda_2}, \dots, {x_m}^{\lambda_m} {y_1}^{\delta_1}, {y_2}^{\delta_2}, \dots, {y_n}^{\delta_n}.$$

Desafortunadamente la palabra del lado derecho de la ecuación puede no ser reducida (por ejemplo ${x_m}^{\lambda_m} = {y_1}^{-\delta_1}$). Por tanto, se define el producto dado por la yuxtaposición y (si es necesario) la cancelación de términos adyacentes de la forma xx^{-1} o $x^{-1}x$; por ejemplo $({x_1}^1 {x_2}^{-1} {x_3}^1)({x_3}^{-1} {x_2}^1 {x_4}^1) = {x_1}^1 {x_4}^1$. De forma más precisa, si ${x_1}^{\lambda_1}, {x_2}^{\lambda_2}, \dots, {x_m}^{\lambda_m}$ y ${y_1}^{\delta_1}, {y_2}^{\delta_2}, \dots, {y_n}^{\delta_n}$ son palabras no vacías en *X* con $m \leq n$, sea *k* el entero más largo ($0 \leq k \leq m$) tal que ${x_{m-j}}^{\lambda_{m-j}} = {y_{j+1}}^{-\delta_{j+1}}$ para $j = 0,1,2 \dots k-1$. Entonces se define:

$$(x_1{}^{\lambda_1}\cdots x_m{}^{\lambda_m})(y_1{}^{\delta_1}\cdots y_n{}^{\delta_n}) = \begin{cases} x_1{}^{\lambda_1}\cdots x_{m-k}{}^{\lambda_{m-k}} y_{k+1}{}^{\delta_{k+1}}\cdots y_n{}^{\delta_n} \text{ if } k < m; \\ y_{m+1}{}^{\delta_{m+1}}\cdots y_n{}^{\delta_n} \quad \text{if} \quad k = m < n; \\ 1 \quad \text{if} \quad k = m = n. \end{cases}$$

Si $m > n$, el producto es definido de forma análoga. La definición asegura que el producto de palabras reducidas sea una palabra reducida.

Teorema 1.40 Si *X* es un grupo no vacío y $F = F(X)$ es el conjunto de todas las palabras reducidas de *X*, entonces *F* es un grupo bajo la operación binaria arriba y $F =< X >$.
El grupo $F = F(X)$ es llamado **el grupo libre en el conjunto *X***. (La terminología "libre" será demostrada en el próximo teorema).

Demostración: Como *1* es un elemento identidad y ${x_1}^{\lambda_1}, {x_2}^{\lambda_2}, \dots, {x_m}^{\lambda_m}$ tiene como inverso ${x_1}^{-\lambda_1}, {x_2}^{-\lambda_2}, \dots, {x_m}^{-\lambda_m}$, se necesita solo verificar la asociatividad. Esto puede ser demostrado por inducción y una tediosa examinación de casos o por el siguiente medio más elegante. Para cada $x \in X$ y $\delta = \mp 1$ sea $|x^\delta|$ el mapeo $F \to F$ dado por $1 \vdash x^\delta$ y

$$x_1{}^{\delta_1}\cdots x_n{}^{\delta_n} \vdash\!\!\to \begin{cases} x^\delta x_1{}^{\delta_1}\cdots x_n{}^{\delta_n} & \text{if} \quad x^\delta \neq x_1{}^{-\delta_1}; \\ x_2{}^{\delta_2}\cdots x_n{}^{\delta_n} & \text{if} \quad x^\delta = x_1{}^{-\delta_1} \ (= 1 \text{ if } n = 1). \end{cases}$$

Como $|x||x^{-1}| = 1_F = |x^{-1}||x|$, cualquier $\left|x^\delta\right|$ es una permutación (biyección) de *F* (con inversa $\left|x^{-\delta}\right|$). Sea $S(F)$ el grupo de todas las permutaciones de *F* (esto fue definido anteriormente) y F_0 el subgrupo de *F* generado por $\{|x|: x \in X\}$. El mapeo $\varphi: F \to F_0$ dado por $1 \vdash 1_F$ y ${x_1}^{\lambda_1}, {x_2}^{\lambda_2}, \dots, {x_m}^{\lambda_m} \vdash \left|{x_1}^{\lambda_1}\right|\left|{x_2}^{\lambda_2}\right| \dots \left|{x_m}^{\lambda_m}\right|$ es claramente una sobreyección (debido al propia definición de F_0) tal que $\varphi(w_1 w_2) = \varphi(w_1)\varphi(w_2)$ para todo $w_i \in F$. Como $1 \vdash {x_1}^{\lambda_1}, {x_2}^{\lambda_2}, \dots, {x_m}^{\lambda_m}$ bajo el mapa $\left|{x_1}^{\lambda_1}\right|\left|{x_2}^{\lambda_2}\right| \dots \left|{x_m}^{\lambda_m}\right|$, se obtiene $Ker\ \varphi = \{1_F\}$, por lo que φ es inyectiva. El hecho de que F_0 sea un grupo implica que la asociatividad se cumple en *F* y que φ es un isomorfismo de grupos. Obviamente $F =< X >$ ■. (La demostración de este teorema es un caso particular del teorema de Cayley, se logra un isomorfismo entre *F* y un subgrupo del grupo de permutaciones $S(F)$.)

Ciertas propiedades de los grupos libres son fácilmente derivadas. Por ejemplo si $|X| \geq 2$, entonces el grupo libre sobre *X* es no abeliano ($x, y \in X$ y $x \neq y \Rightarrow x^{-1}y^{-1}xy\ es\ reducida \Rightarrow x^{-1}y^{-1}xy \neq 1 \Rightarrow xy \neq yx$).

Teorema 1.41 Sea *F* el grupo libre sobre un conjunto *X* y $i: X \to F$ el mapa de inclusión. Si *G* es un grupo y $f: X \to G$ un mapeo de conjuntos, entonces un único homomorfismo de grupos $\bar{f}: F \to G$ tal que $\bar{f}i = f$. En otras palabras, *F* es un objeto libre del conjunto *X* en la categoría de los grupos.

Demostración: Se define $\bar{f}(1) = e$ y si ${x_1}^{\lambda_1}, {x_2}^{\lambda_2}, \dots, {x_m}^{\lambda_m}$ es una palabra no vacía y reducida sobre *X*, se define $\bar{f}\left({x_1}^{\lambda_1}, {x_2}^{\lambda_2}, \dots, {x_m}^{\lambda_m}\right) = f(x_1)^{\lambda_1} f(x_2)^{\lambda_2} \dots f(x_m)^{\lambda_m}$. Como *G* es un grupo y $\lambda_i = \overline{+}1$, el producto $f(x_1)^{\lambda_1} f(x_2)^{\lambda_2} \dots f(x_m)^{\lambda_m}$ está bien definido en *G.* Se verifica que $\bar{f}$ es un homomorfismo tal que $\bar{f}i = f$. Si $g: F \to G$ es cualquier homomorfismo tal que $gi = f$, entonces $g\left({x_1}^{\lambda_1}, {x_2}^{\lambda_2}, \dots, {x_m}^{\lambda_m}\right) = g(x_1)^{\lambda_1} g(x_2)^{\lambda_2} \dots g(x_m)^{\lambda_m} = gi(x_1)^{\lambda_1} gi(x_2)^{\lambda_2} \dots gi(x_m)^{\lambda_m} = f(x_1)^{\lambda_1} f(x_2)^{\lambda_2} \dots f(x_m)^{\lambda_m} = \bar{f}\left({x_1}^{\lambda_1}, {x_2}^{\lambda_2}, \dots, {x_m}^{\lambda_m}\right)$. Por tanto $\bar{f}$ es única■.

Corolario 1.42 Todo grupo *G* es la imagen homomorfa de un grupo libre.

Demostración: Sea *X* un conjunto de generadores de *G* y sea *F* el grupo libre sobre el conjunto *X.* Por el teorema anterior, el mapeo de inclusión $X \to G$ induce un homomorfismo $\bar{f}: F \to G$ tal que $x \vdash x \in G$. Como $G =< X >$, la prueba del teorema muestra que $\bar{f}$ es un epimorfismo■.

Una consecuencia inmediata del Corolario 1.42 y el Primer Teorema de Homomorfismo es que cualquier grupo *G* es isomorfo a un grupo cociente F/N, donde $G =< X >$, *F* es el grupo libre sobre *X* y *N* es el núcleo del epimorfismo $F \to G$ de Corolario 1.42. Por tanto en orden para describir *G* mediante un isomorfismo se necesita solamente especificar *X, F* y *N.* Pero *F* es determinado por *X* y *N* es determinado por cualquier subconjunto que lo genere como un subgrupo de *F*. Ahora si $w = {x_1}^{\lambda_1}\, {x_2}^{\lambda_2} \dots {x_m}^{\lambda_m} \in F$ es un generador de *N*, entonces bajo el epimorfismo $F \to G$, se tiene $w \vdash {x_1}^{\lambda_1}\, {x_2}^{\lambda_2} \dots {x_m}^{\lambda_m} = e \in G$. La

ecuación $x_1{}^{\lambda_1}\, x_2{}^{\lambda_2}\, \ldots x_m{}^{\lambda_m} = e$ en *G* es llamada una **relación** sobre los generadores x_i. Claramente un grupo dado *G* puede ser completamente descrito especificando un conjunto *X* de generadores de *G* y un conjunto *R* de relaciones validas sobre estos generadores. Esta descripción no es única ya que hay varias elecciones posibles de *X* y *R* para un grupo *G* dado.

Por el contrario, suponga que son dados un conjunto *X* y un conjunto *Y* de palabras (reducidas) sobre elementos de *X*. Pregunta: ¿Existirá un grupo *G* tal que *G* es generado por *X* y todas las relaciones $w = e\ (w \in Y)$ son válidas (donde $w = x_1{}^{\lambda_1}\, x_2{}^{\lambda_2}\, \ldots x_m{}^{\lambda_m}$ ahora denota el producto en *G*)? Veremos que la respuesta es sí.

Dado un conjunto de "generadores" *X* y un conjunto *Y* de palabras (reducidas) sobre los elementos de *X*, se construye el grupo de la siguiente forma. Sea *F* el grupo libre generado sobre *X* y *N* el subgrupo normal de *F* generado por *Y* (el subgrupo normal generado por un conjunto $S \in F$ es la intersección de todos los subgrupos normales de *F* que contienen a *S*). Sea *G* el grupo cociente F/N, y *X* se identifica con su imagen en F/N bajo el mapeo $X \subset F \to F/N$, como se denotó arriba. Entonces *G* es un grupo generado por *X* y por construcción todas las relaciones $w = e\ (w \in Y)$ son satisfechas.

Definición 1.43 Sea X un conjunto y Y un conjunto de palabras (reducidas) en X. Un grupo *G* es llamado **grupo definido por los generadores $x \in X$ y relaciones $w = e\ (w \in Y)$** denotado por $G \cong F/N$, donde *F* es el grupo libre sobre *X* y *N* el subgrupo normal generado por *Y*. Se dice que $(X|Y)$ es una presentación de *G*.

La discusión precedente muestra que el grupo definido por sus generadores y relaciones siempre existen. Además es el grupo más largo posible en el siguiente sentido.

Teorema 1.44 (Van Dyck) Sea X un conjunto, Y un conjunto de palabras (reducidas) sobre *X* y *G* el grupo definido por los generadores $x \in X$ y las relaciones $w = e\ (w \in Y)$. Si *H* es un grupo cualquiera tal que $H =< X >$ y H satisface las relaciones $w = e\ (w \in Y)$, entonces existe un epimorfismo $G \to H$.

Ejemplo El grupo definido por un generador *b* y una simple relación $b^m = e\ (m \in N)$ es el grupo Z_m.

Ejemplo El grupo definido por los generadores a, b y las relaciones $a^n = e\ (3 \leq n \in N), b^2 = e\ y\ abab = e$ es el grupo dihedral D_n.

1.4 Presentaciones y sistemas de reescrituras de palabras.

Una presentación de un grupo G es una especificación abstracta de G en términos de generadores y relaciones entre ellos. Tales especificaciones son convenientes para algunos usos por su carácter abstracto, y por ser relativamente compactas en algunos casos. Sin embargo, plantean una serie de problemas computacionales de difícil solución.

Sea (X, R) la presentación de un grupo G. El problema de las palabras para esta presentación se define de la forma siguiente:

Definición 1.45 (Problema de las palabras general) Sea (X, R) una presentación de un grupo G:

Instancia: Dos palabras $\alpha, \beta \in X^*$.

Pregunta: ¿$\alpha \equiv_G \beta$?

El *problema de las palabras* para un grupo $G = (X, R)$, fue introducido por Max Dehn en 1911 junto con los problemas de *búsqueda del conjugado y el isomorfismo*. El *problema de las palabras* plantea el siguiente problema de decisión: ¿para una palabra w arbitraria sobre $X \cup X^{-1}$, es w equivalente a la palabra vacía? Claramente este problema de decisión sirve para resolver el problema general de equivalencia entre las palabras, para probar la equivalencia entre a y b, es necesario solamente resolver el problema de las palabras con la palabra ab^{-1}.

En la década de 1950, Novikov y Boone independientemente demostraron que existen grupos finitamente presentados para los cuales el problema de las palabras no es resoluble. Es un hecho importante que la posibilidad de resolver el problema de las

palabras de un grupo finitamente presentado depende solamente del grupo, y no de la presentación escogida. En otras palabras, si el grupo G tiene un problema de las palabras soluble para la presentación (X, R), entonces G tiene un problema de las palabras soluble para cualquiera de sus posibles presentaciones.

La idea de considerar las presentaciones de grupos como un **Sistema de Reescritura de Palabras** resulta bastante natural, de hecho a partir de 1940 la *Combinatoria de Grupos* y los *Sistemas de Reescritura* se encontraron, debido fundamentalmente a los trabajos (Thue, 1914), (A. Church, 1939) y (Markov, 1947), entre otros.

Formalmente, un *sistema de reescritura de palabras* sobre un alfabeto X se define de igual manera que una presentación de monoide o grupo; es decir, como un par$< X: R >$, donde X es un alfabeto finito y R es una relación binaria sobre X^*. La diferencia estriba en el uso que se hace de estos dos objetos. En el caso de la *presentación de grupo* siempre trabajamos con la congruencia generada por R, que es simétrica; o sea, utilizamos las parejas de palabras $(u, w) \in R$ como reglas de transformación bidireccionales. En cambio, en un *sistema de reescritura* las reglas se utilizan en una sola dirección; se trabaja con la clausura reflexiva y transitiva de R, pero no con su clausura simétrica. La ventaja de utilizar las reglas en una sola dirección radica en que se obtiene un enfoque algorítmico uniforme (un modelo de computación) para resolver problemas en la estructura algebraica asociada.

Definición 1.46 Dado un sistema de reescritura $< X: R >$, si existe $(l, r) \in R$, y existen $(x, y) \in X^*$ tales que $u = xly$ y $v = xry$, se dice que u **se reduce** a v (en un paso), y se escribe $u \rightarrow_R v$. La clausura reflexiva y transitiva de $\rightarrow_R$ se llama **relación de reducción** (inducida por R), y se denota por $\overset{*}{\rightarrow}$. Si $u \overset{*}{\rightarrow} v$, se dice que u se reduce a v, y u se llama reducible. $\leftrightarrow$ y $\overset{*}{\leftrightarrow}$ denotan respectivamente, la clausura simétrica, y la clausura reflexiva, simétrica y transitiva de $\rightarrow$ (esta última coincide con $\equiv_R$).

Un sistema de reescritura permite aplicar transformaciones sucesivas a una palabra, hasta llegar (idealmente) a una palabra irreducible. Ahora bien, si en un momento dado

hay más de una regla aplicable, en general, el resultado final de este proceso no será único. Se desea imponer ciertas condiciones a los sistemas de reescritura que permitan garantizar que el resultado final sea calculable y único.

Definición 1.47 Un *sistema de reescritura* R se dice de **terminación, bien fundado** o **noetheriano** si no existe una sucesión infinita $u_1 \to u_2 \to \cdots \to u_n \to \cdots$. Se dice que R posee la **propiedad Church-Rosser** si $u \equiv_R v$ implica que existe w tal que $u \overset{*}{\leftrightarrow} w$ y $v \overset{*}{\leftrightarrow} w$. Si R es a la vez de terminación y Church-Rosser, se dice que es **completo**, **canónico**, o **convergente.**

De esta manera, si se utiliza un sistema de reescritura completo, cada palabra irreducible puede tomarse como representante de su clase de equivalencia. Si nuestro sistema de reescritura completo se utiliza como presentación de un grupo G, poseerá una correspondencia entre la palabras irreducibles y los elementos de G. A una tal presentación de G se le llamará **presentación completa** de G, y permite resolver el problema de las palabras. En efecto, se denota por $FN_R(w)$ a la única palabra irreducible que se encuentra en la clase de equivalencia de w, y sean $\alpha, \beta \in X^*$; entonces, $\alpha \equiv_R \beta$ si, y solo si, $FN_R(\alpha) = FN_R(\beta)$.

Dos sistemas de reescritura de R y S sobre un mismo alfabeto X se dice **equivalentes** si $\equiv_R = \equiv_S$. Dado un sistema arbitrario R, se puede construir un sistema completo $\bar{R}$ equivalente a R, por medio del conocido procedimiento de Knuth-Bendix (si este termina) (D.E Knuth, 1970). El procedimiento utiliza de manera esencial un **ordenamiento de términos** sobre X^*.

Definición 1.48 (Ordenamiento de términos) Sea < un orden parcial estricto bien fundado sobre X^*, y R un sistema de reescritura sobre X. Se dice que < es admisible si $u < v$ implica $xuy < xvy$, para todo $u, v, x, y \in X^*$. Los ordenamientos bien fundados y admisibles se denominan **ordenamientos de términos**. Se dice que < es compatible con R si $\beta < \alpha$ para todas las reglas de reescritura de $\alpha \to \beta$ en R.

Uno de los ordenamientos de términos más utilizados es el llamado “Short-Lex” o “Deg-Lex”, donde la comparación entre dos palabras se resuelve primero por su longitud, y los empates se resuelven lexicográficamente.

Conclusiones parciales del capítulo

Se han expuesto los conceptos básicos de algunos principios criptográficos, teoría de grupos y los *sistemas de reescritura*. Los conceptos criptográficos que se expusieron fueron completamente introductorios para que el lector conociera términos básicos como *sistema de llave pública* o *sistema de llave privada*, cualquier interesado en profundizar un poco más en el tema puede consultar (Konheim, 2007). El material tratado en las secciones de teoría de grupo es un poco complicado, y el camino recorrido para poder llegar a la simple definición de *presentación de un grupo* fue largo y complicado. La sección dedicada a *los sistemas de reescritura*, contiene la teoría necesaria para trabajar con grupos con métodos computacionales. En general se realizó un análisis y una recopilación de la bibliografía para exponer y explicar las diferentes teorías y temáticas abordadas en el capítulo, que servirán como sustento del trabajo.

2 Diseño e implementación de los algoritmos criptográficos

El capítulo 2 comienza con las siguientes preguntas que relacionan la **Teoría de Grupos Combinatoria** y los **Criptosistemas de clave pública:**

Pregunta 1: ¿Existe un grupo, o una clase de grupos, donde el protocolo de intercambio de clave mutua sugerido en (Anshel I., 1999) utilizando el problema de la búsqueda del conjugado, sea lo suficientemente seguro para aplicaciones en la vida real?

Pregunta 2: ¿Existe otro problema "difícil" en la *Teoría de Grupos Combinatoria* que pueda ser utilizado, en lugar del problema de la búsqueda del conjugado, en un protocolo de intercambio de clave mutua?

Sin una respuesta positiva para al menos una de estas preguntas, es poco probable que la *Teoría de Grupos Combinatoria* tenga un impacto significante en la Criptografía de clave pública, la cual es dominada en la actualidad por la *Teoría de Números* y *Curvas Elípticas*.

Se apunta a otra pregunta más, la cual no ha recibido la suficiente atención hasta ahora, pero es probable que sea el centro de la investigación en la Criptografía de clave pública basada en computación simbólica (Shpilrain Vladimir, 2006):

Pregunta 3: ¿Se puede disfrazar eficientemente un elemento de un grupo utilizando sus relaciones definitorias?

Disfrazar un elemento antes de transmitirlo es a menudo llamado "difusión" (Garrett, 2001). La importancia de esto es obvia: si, por ejemplo, se transmite un conjugado axa^{-1} de un elemento público a, "tal cual y como es", entonces el oponente mediante simple inspección puede determinar el elemento privado x. Problemas similares aparecen en cualquier otro *protocolo de intercambio de clave mutua.* En protocolos basados en *teoría de números,* la difusión es usualmente facilitada "automáticamente", debido a varias

propiedades de los sistemas numéricos utilizados. Por ejemplo, en el producto $7 * 3 = 21$, los factores 7 y 3 no pueden ser determinados por simple inspección; esto es provisto simplemente por la forma que multiplicamos enteros en el sistema decimal, o, equivalentemente, por la existencia de una sencilla “forma normal” para los enteros.

Tomando como guía la **Pregunta 3,** en este capítulo se realiza con detalles la descripción del nuevo algoritmo criptográfico basado en la presentación de grupos seguido con la programación de las herramientas en el lenguaje orientado a objetos Java.

2.1 Estructuras de datos y algoritmos para el trabajo con las palabras de un grupo

Como se dijo en el *epígrafe 1.4* los sistemas de reescritura son idóneos para el trabajo con grupos mediante sus presentaciones. Recordando, un sistema de reescritura es un par $< X, R >$, donde X es el alfabeto (conjunto de generadores en caso de un grupo) y R el conjunto de sustituciones (conjunto de relaciones definitorias en caso de un grupo). Además se vio la posibilidad tratar la presentación de un grupo (X, R), como un sistema reescritura **convergente** $< X, \bar{R} >$, pues se puede construir la función $FN_R(w)$ que lleva cada palabra a su único representante canónico.

Este epígrafe está compuesto de dos secciones. La primera correspondiente a la implementación del algoritmo de Knuth-Bendix (D.E Knuth, 1970), el cual permite mediante la presentación (X, R) de un grupo obtener un sistema de reescritura $< X, \bar{R} >$ que sea **convergente**. La segunda expone el árbol de Aho-Korasick (Aho, 1975) capaz de hacer en tiempo lineal el macheo de varios patrones, lo cual servirá para una implementación eficiente del algoritmo criptográfico.

2.1.1 Algoritmo de completamiento de Knuth-Bendix

Para la implementación de este algoritmo se utilizaron las ideas expuestas en (Huet, 1981), el cual contiene un demostración completa y rigurosa del procedimiento (algoritmo en caso de que termine). El teorema de Knuth-Bendix (HUET, 1980), (D.E Knuth, 1970)

brinda un proceso de decisión para la **confluencia de sistemas de reescritura noetherianos.** La idea básica es considerar los casos donde dos lados izquierdos de reglas en un sistema de reescritura $< X,\ R >$ se superponen de una forma no trivial para crear una ambigüedad de la forma $M \rightarrow_R N_1$ y $M \rightarrow_R N_2$ (Se dice que (N_1, N_2) es un *par crítico*). El sistema $< X,\ R >$ es no confluente si, y solo si, para algún par como el anterior, se tiene que N_1 y N_2 se reducen a representantes canónicos diferentes P_1 y P_2 respectivamente, mediante R.

El algoritmo de completamiento de Knuth-Bendix consiste en intentar completar un sistema *no confluente* en uno *confluente* mediante la adición de nuevas reglas, tales como $P_1 \rightarrow P_2$. Esto debe ser hecho de una forma que el sistema continúe siendo *noetheriano*. Por supuesto, una sola ronda de completamiento no es suficiente en general, ya que nuevas ambigüedades pueden ser creadas. Durante el proceso de completamiento alguna regla recién introducida puede simplificar una regla antigua en su lado izquierdo o derecho. Es esencial tanto para la eficiencia como para la elegancia, mantener la reglas inter reducidas tanto como se pueda. Pero entonces surge la pregunta de cómo el proceso puede ser realizado de una forma incremental; esto significa, que no se quiere re computar pares críticos entre reglas que hayan sido consideradas previamente. Como sea, las reglas que hayan sido utilizadas para resolver ambigüedades pueden ya no existir más, por lo que este paso debe ser justificado cuidadosamente. Cuando un conjunto de ecuaciones puede ser orientado de forma tal que el proceso de completamiento termina, el *sistema de reescritura de términos canónico* define una función $FN_R(w)$, capaz de llevar una palabra a su único representante canónico.

Se presenta el algoritmo de completamiento de Knuth-Bendix de una forma incremental en el cómputo de los pares críticos. A continuación, E_i es un conjunto finito de ecuaciones, y R_i es un sistema de reescritura finito, para cada $i \in N$. Cada regla de reescritura en R_i tiene un etiqueta, el cual es un entero único. Se denota por $k{:}\ \lambda \rightarrow \rho$ la regla de reescritura $\lambda \rightarrow \rho$ con etiqueta k. Finalmente cada regla en R_i puede estar marcada o no.

Entrada : Un conjunto finito de ecuaciones E y un ordenamiento de términos $<$. En nuestro caso $E = R$, que proviene de $< X, R >$ y $< = \ ShortLex$
Salida : Un conjunto de reglas que representa un sistema de reescritura canónico y equivalente al de entrada.
Inicialización : $E_0 := E$; $R_0 := \varnothing$; $i := 0$; $p := 0$

1: **loop**
2: **while** $E_i \neq \varnothing$ **do**
3: *Reducir ecuación :* Seleccionar una ecuación $M = N$ en E_i.
4: Sean $M\downarrow$ y $N\downarrow$ obtenidas mediante la aplicación sucesiva de reglas de R_i, hasta que no se pueda aplicar ninguna más.
5: **if** $M\downarrow = N\downarrow$ **then**
6: $E_{i+1} := E_i - \{M = N\}$; $R_{i+1} := R_i$; $i := i + 1$;
7: **else if** $M\downarrow < N\downarrow$ **or** $M\downarrow > N\downarrow$ **then**
8: $\lambda := max(M\downarrow, N\downarrow)$; $\rho := min(M\downarrow, N\downarrow)$;
9: *Agregar nueva regla :* Sea K el conjunto de etiquetas k de reglas de R_i, tal que su lado izquierdo λ_k es reducible por $\lambda \to \rho$, digamos a λ_k'.
10: $E_{i+1} := E_i - \{M = N\} \cup \{\lambda_k' = \rho_k | k : \lambda_k \to \rho_k \in R_i \ donde \ k \in K\}$;
11: $p := p + 1$;
12: $R_{i+1} := \{j : \lambda_j \to \rho_k' | j : \lambda_j \to \rho_k \in R_i \ con \ j \notin K\} \cup \{p : \lambda \to \rho\}$
13: Las reglas provenientes de R_i son marcadas o no marcadas según como eran R_i. La nueva regla $\lambda \to \rho$ es no marcada.
14: $i := i + 1$
15: **else**
16: **return** (FALLO: $M\downarrow$ y $M\downarrow$ no son comparables)
17: **end if**
18: **end while**
19: *Computar pares críticos :*
20: **if** todas las reglas en R_i están marcadas **then**
21: **return** (R_i sistema de reescritura canónico)
22: **else**
23: Seleccionar una regla en R_i, digamos con etiqueta k. Sea E_{i+1} el conjunto de todos los pares críticos computados entre la regla k y cualquier regla de R_i con etiqueta menor que k. Sea R_{i+1} el mismo que R_i, excepto que ahora la regla k está marcada.
24: $i := i + 1$
25: **end if**
26: **end loop**

Algoritmo 2.1: Implementación incremental del algoritmo de Knuth-Bendix.

Capítulo 2

Dado un conjunto de ecuaciones E y un ordenamiento de términos $<$, el algoritmo puede terminar con éxito, parar con fallo, o iterar por siempre. En este caso E será igual a R y $<$ será Short-Lex.

En la *línea 23* del algoritmo anterior se plantea la búsqueda de posibles pares críticos entre la regla k de R_i y cualquier regla con etiqueta menor que k en R_i. Ahora se describirá como resolver el problema de encontrar pares críticos entre dos reglas.

Suponga que existen dos reglas $\lambda_a \to \rho_a$ y $\lambda_b \to \rho_b$, con $a \neq b$ y que sus partes izquierdas se superponen de forma no trivial, para esto existe dos casos:

1. $\lambda_a = B$ y $\lambda_b = ABC$ (o $\lambda_b = B$ y $\lambda_a = ABC$); en este caso λ_a es subpalabra de λ_b (o λ_b es subpalabra de λ_a).
2. $\lambda_a = AB$ y $\lambda_b = BC$ (o $\lambda_b = AB$ y $\lambda_a = BC$); en este caso un sufijo de λ_a es igual a un prefijo de λ_b (o un sufijo de λ_b es igual a un prefijo de λ_a).

En cualquiera de los dos casos se reduce la palabra ABC usando $\lambda_a \to \rho_a$ y $\lambda_b \to \rho_b$, a los resultados se les llama r_1 y r_2 respectivamente. Si se tiene que $r_1 \not\equiv_{R_i} r_2$, entonces existe un **par crítico** entre las reglas $\lambda_i \to \rho_i$ y $\lambda_j \to \rho_j$. Por lo que se debe agregar la ecuación $r_1 = r_2$, al conjunto E_{i+1}.

Para resolver la problemática del *Caso 1,* se puede utilizar el algoritmo de macheo de cadenas conocido como Knuth-Morris-Pratt en la literatura (DONALD E. KNUTH, 1977), (Thomas H. Cormen, 2001). Este algoritmo permite dados un texto T y un patrón P, encontrar todas la ocurrencias de P en T, con una complejidad temporal de $O(|T| + |P|)$. KMP es óptimo en el sentido de que no existe ningún algoritmo de macheo de cadenas con complejidad inferior a este (Maxime Crochemore, 2007).

Suponiendo que el largo de las palabras es de $O(N)$, la problemática del *Caso 2* se puede resolver de forma ingenua con una complejidad cuadrática $O(N^2)$, al ir comparando iterativamente los sufijos de una palabra contra prefijos de la otra; pero esta complejidad

no es computacionalmente aceptable, pues el algoritmo de Knuth-Bendix hace un uso intensivo de la búsqueda de *pares críticos*.

A continuación se muestra una forma mucho más complicada que la anterior estrategia ingenua, pero sin embargo con un complejidad de sólo $O(N * Log(N))$, lo cual reduce considerablemente el tiempo de cómputo del algoritmo de completamiento de Knuth-Bendix.

Para esto se utilizará la estructura de datos **Arreglo de Sufijos *(Suffix Array)*** (Udi Manber, 1991), (Adrian Vladu, 2005). El *Arreglo de Sufijos* de la cadena s, se denota por $SA(s)$, o simplemente SA y contiene ordenados lexicográficamente todos los sufijos de s. Cada sufijo es representado por su posición en s, el sufijo $i-ésimo\ (Suff_i)$ es la subcadena $s[i \ldots length(s)-1]$ de s. Se tiene que, $SA[i] = j$ si, y solo si, $Suff_j$ es el $i-ésimo$ sufijo lexicográficamente más pequeño de s. El arreglo de sufijos es normalmente utilizado en conjunción con un arreglo llamado Lcp, conteniendo el largo del prefijo común más largo (*Longest Common Prefix*) entre cada par consecutivo de sufijos en SA. Se utiliza $lcp(\alpha, \beta)$ para denotar el largo del prefijo común más largo entre la cadena α y β. Se tiene que, $Lcp[i]$ contiene el largo del prefijo común más largo entre la cadena $Suff_{SA[i]}$ y $Suff_{SA[i+1]}$, o sea $Lcp[i] = lcp(Suff_{SA[i]}, Suff_{SA[i+1]})$.

Lema 2.1 Para todo $0 \leq i < j < length(s)$, se tiene que $lcp\left(Suff_{SA[i]}, Suff_{SA[j]}\right) = min_{k=i}^{j-1} Lcp[k]$ (Dinesh P. Mehta, 2005).

Por tanto se puede calcular el prefijo común más largo entre cualquier par de sufijos utilizando la fórmula del lema anterior y teniendo de antemano el arreglo Lcp. El cálculo de $min_{k=i}^{j-1} Lcp[k]$ se puede lograr con una complejidad de $O(Log(N))$ utilizando una estructura de datos del tipo *Range Minimum Query (RMQ) (TopCoder, 2003),* en particular se utilizó un *tabla esparcida (sparse table),* que logra esta complejidad y es de gran sencillez en su implementación, aunque también se puede utilizar un *árbol de segmentos (segment tree).*

Capítulo 2

El seudocódigo a continuación muestra de forma muy general como se realiza el proceso de inicialización de las estructuras de datos antes mencionadas y su posterior utilización con el propósito de dar solución al *Caso 2.* De hecho, se asumirá que se cuenta con las siguientes funciones: $\boldsymbol{build\_suffix\_array}(\boldsymbol{s})$ la cual al pasarle un cadena s devuelve su arreglo de sufijos $SA[s]$ con una complejidad de $O(N * Log(N))$, $\boldsymbol{build\_lcp\_array}(\boldsymbol{SA})$ la cual al pasarle un arreglo de su sufijos SA devuelve su correspondiente arreglo Lcp en tiempo $O(N)$, explicación de cómo implementar estas dos funciones y mucho más sobre trabajo con cadenas se puede encontrar en (Udi Manber, 1991), (Maxime Crochemore, 2007), (Adrian Vladu, 2005) y (Maxime Crochemore, 1997), por último se cuenta con la función $\boldsymbol{build\_sparse\_table}(\boldsymbol{A})$ la cual dado un arreglo A devuelve una *tabla dispersa* en tiempo $O(N * Log(N))$ capaz de responder las preguntas del tipo $min_{k=i}^{j} A[k]$ con una complejidad $O(Log(N))$, como implementar esta potente estructura de datos puede ser encontrado en (TopCoder, 2003), se supone que esta estructura de datos contiene una función $\min(i, j)$ la cual puede ser llamada y retorna el valor $min_{k=i}^{j} A[k]$.

Entrada : Dos cadenas A y B.
Salida : El largo de la mayor cadena posible Y, tal que $A = XY$ y $B = YZ$.

```
1: Construir una nueva cadena S = AB, al concatenar A y B.
2: SA := build_suffix_array(S);
3: Lcp := build_lcp_array(SA);
4: ST := build_sparse_table(Lcp);
5: n := |A|; m := |B|;
6: Declarar un arreglo Buckt, que será la permutación inversa de SA, o sea
   Buckt[i] = j indica que SA[j] = i.
7: for i := 0 to n + m − 1 do
8:     Buckt[SA[i]] := i;
9: end for
10: largo := 0;
11: for i := 1 to min(n, m) do
12:     a := min(Buckt[n − i], Buckt[n]);
13:     b := max(Buckt[n − i], Buckt[n]);
14:     k := ST.min(a, b − 1);
15:     if k ≥ i then
16:         largo := i
17:     end if
18: end for
19: return largo
```

Algoritmo 2.2: Algoritmo para hallar el mayor solapamiento.

En el algoritmo anterior se utiliza el arreglo $Buckt$ para saber la posición que ocupa un sufijo en el arreglo de sufijos y de esa forma poder calcular el ***lcp*** entre cualquier par de sufijos utilizando la tabla esparcida. El bucle ***for*** de la línea 11 a la 18, analiza el ***lcp*** entre los sufijos de las posiciones $n - i$ y n, los cuales coinciden el sufijo de largo i de la cadena A y el comienzo de la cadena B, respectivamente. Al tener que él ***lcp*** entre estos sufijos es mayor que i, se puede asegurar que el sufijo de largo i de la cadena A y el prefijo de la largo i de la cadena B son iguales.

Si se hace $N = n + m$ (el largo de la cadena S), se obtiene $O(N) = O(\max(n, m))$ y que el proceso de inicialización de todas las estructuras de datos y el arreglo $Buckt$ es de $O(N * Log(N) + N + N * Log(N) + N) = O(N * Log(N))$. El bucle ***for*** de la línea 11 a la

18 itera $\min(n,m)$ veces, como $\min(n,m) \leq \max(n,m)$ se tiene que $\min(n,m) \in O(N)$, además cada iteración consume un tiempo de $O(Log(N))$, por tanto la complejidad de este bucle es $O(N * Log(N))$. Finalmente la complejidad total del algoritmo completo es de $O(N * Log(N))$, tal y como se quería.

2.1.2 Implementación de un sistema de reescritura utilizando un árbol de Aho-Korasick

En el *epígrafe 1.4* se vio que un sistema de reescritura es un par $< X, R >$, suponiendo que $R = \{\lambda_1 \to \rho_1,\ \lambda_2 \to \rho_2, \dots, \lambda_n \to \rho_n\}$, o sea que está compuesto por n reglas. La forma de reducir una palabra w mediante un sistema de reescritura es ir aplicando reducciones mediante estas reglas hasta que no se pueda continuar.

El **Algoritmo 2.3** muestra cómo resolver este problema utilizando un algoritmo de macheo de cadenas de tiempo lineal como Knuth-Morris-Pratt, el cual permitirá saber si una cadena α es subcadena de otra β devolviéndolo en un valor booleano $kmp(\alpha, \beta)$ y se asume que se cuenta con una función $apply(w, \lambda \to \rho)$ que devuelve la cadena w al aplicar una sustitución mediante la regla $\lambda \to \rho$.

Cada iteración del bucle ***loop*** es un paso de sustitución por una regla y la complejidad del bucle ***for*** es de $O(\sum_{i=1}^{n}(|\lambda_i| + |w|)) = O(\sum_{i=1}^{n}(|\lambda_i|) + n * |w|)$. Como se verá más adelante esto se puede reducir dramáticamente a tan solo $O(\sum_{i=1}^{n}(|\lambda_i|) + |w|)$, mediante el uso del algoritmo de macheo de Aho-Korasick.

Entrada : Un sistema de reescritura $< X, R >, R = \{\lambda_1 \rightarrow \rho_1, \lambda_2 \rightarrow \rho_2, ..., \lambda_n \rightarrow \rho_n\}$ y un cadena w a reducir.
Salida : La cadena w reducida.

```
1: loop
2:     id := −1;
3:     for i := 1 to n do
4:         if kmp(λ_i, w) then %(λ_i es una subcadena de w)%
5:             id := i;
6:             break;
7:         end if
8:     end for
9:     if id = −1 then
10:        break;
11:    else
12:        w := apply(w, λ_id → ρ_id);
13:    end if
14: end loop
```

Algoritmo 2.3: Reducción de una cadena mediante un sistema de reescritura.

En el mundo de la computación, un árbol de *Aho-Korasick* (Aho, 1975), (Kilpelainen, 2003) es una estructura de datos que sirve como un algoritmo de macheo de cadenas. Puede ser visto como un algoritmo del tipo diccionario-macheo que ubica elementos de un conjunto finito de cadenas (el "diccionario") dentro de un texto de entrada. Este realiza un macheo simultáneo de todos los patrones. La complejidad del algoritmo es lineal en el largo de los patrones, más el largo del texto buscado, más el número de macheos contra el texto, pero en este caso cuando se encuentre cualquier macheo se puede parar significando que una regla es aplicable, por tanto la complejidad de aplicar el algoritmo de Aho-Korasick en el macheo del diccionario $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$ contra el texto w, sería exactamente de $O(\sum_{i=1}^{n}(|\lambda_i|) + |w|)$.

Informalmente, el algoritmo construye una máquina de estado finito parecida a un ***trie*** *(Briandais, 1959), (Fredkin, 1960)*, con enlaces adicionales a diversos nodos internos. Estos enlaces adicionales permiten transiciones rápidas de macheos de patrones fallidos, hacia otras ramas del *trie* que comparten un prefijo común. Esto le permite al

autómata hacer transiciones entre macheos de patrones sin realizar vueltas atrás (*backtracking*).

El algoritmo de Aho-Korasick fue la base original del comando de *Unix* ***fgrep***.

A continuación se mostrará un algoritmo para implementar un sistema de reescritura utilizando Aho-Korasick, esta optimización de tiempo es increíblemente grande, pues en el *Algoritmo 2.3* hay que recorrer las palabras w una cantidad de n veces, mientras que en el siguiente algoritmo una sola vez. Se asumirá que se cuenta con una función $\boldsymbol{build\_Aho\_Korasick(D)}$, la cual al pasarle un diccionario (conjunto de palabras) retorna un árbol de Aho-Korasick y este tiene implementada una función $match(w)$ que retorna el *id* de un patrón perteneciente al diccionario D y que machea con el texto w, en caso de no machear ninguno devuelve -1.

Entrada : Un sistema de reescritura $< X, R >, R = \{\lambda_1 \to \rho_1, \lambda_2 \to \rho_2, ..., \lambda_n \to \rho_n\}$ y un cadena w a reducir.
Salida : La cadena w reducida.

```
1: Matcher := build_Aho_Korasick({λ1, λ2, ..., λn});
2: loop
3:     id := Matcher.match(w);
4:     if id = −1 then
5:         break;
6:     else
7:         w := apply(w, λid → ρid);
8:     end if
9: end loop
```

Algoritmo 2.4: Reducción de una cadena mediante un sistema de reescritura implementado con un árbol de Aho-Korasick.

Esta implementación de un sistema de reescritura se puede decir que es eficiente ya que cada paso necesita un tiempo lineal respecto al largo de la palabra a reducir y téngase en cuenta que para reducir una palabra se necesita al menos recorrerla, lo que ya de por si es una complejidad lineal.

2.2 Generación de los grupos utilizados como base para el criptosistema

Se comienza con un grupo G finitamente presentado $G = \{x_1, x_2, \dots, x_n \mid \lambda_1 = \rho_1,\ \lambda_2 = \rho_2, \dots, \lambda_m = \rho_m\}$ y se agregan otras relaciones adicionales $\{\alpha_1 = \beta_1,\ \alpha_2 = \beta_2, \dots, \alpha_p = \beta_p\}$, para obtener otro grupo G'. En este trabajo el grupo G formará parte de la clave pública y G' de la clave privada.

Para la generación de las presentaciones de estos grupos, se creó un algoritmo que de forma aleatoria las genera. Para no caer en ciertas ambigüedades y ser precisos, se explicarán los parámetros exactos que se utilizaron en la implementación actual del criptosistema, aunque perfectamente se pueden utilizar otros parámetros, de hecho se puede ver como una metodología general. En lo adelante se utilizará varias veces la función $\boldsymbol{rnd(n)}$, la cual devuelve un entero aleatorio en el intervalo $[1 \dots n]$ con distribución uniforme, además que entre las palabras existe el operador $+$, de concatenación.

Entrada : Ninguna
Salida : Una palabra aleatoria sobre un conjunto de 50 generadores (alfabeto) y con un largo aleatorio de hasta 200 caracteres.

1: $w := ""$;
2: $largo := rnd(200)$;
3: **for** $i := 1$ **to** $largo$ **do**
4: $\quad id := rnd(50)$;
5: $\quad w := w + x_{id}$;
6: **end for**

Algoritmo 2.5 (RndWord): Genera una palabra aleatoria.

Existe una relación especial entre G y G'. Formalmente en Teoría de Grupos (vea epígrafes 1.2 y 1.3) si N es el subgrupo normal de G generado por $\{\alpha_1{\beta_1}^{-1} = e,\ \alpha_2{\beta_2}^{-1} = e, \dots, \alpha_s{\beta_s}^{-1} = e\}$, entonces se cumple que G' es el grupo cociente $G' = G/N$. Existe una función natural (llamada mapeo cociente) $\varphi: G \to G'$, definida como sigue. Sea x un

elemento de G que representa la relación de equivalencia de la palabra w en G, entonces $\varphi(x)$ es el elemento de G' que representa la relación de equivalencia de la misma palabra w en G'. Con un abuso de notación, también se utilizará la función con palabras, por ejemplo $\varphi(w)$ coincidiría con el representante canónico de w en G', o sea $\varphi(w) = FN_{G'}(w)$.

En este trabajo es muy importante el siguiente lema, que puede ser demostrado simplemente con la propia de definición de grupo cociente (ver subepígrafes 1.2.4 y 1.2.5).

Lema 2.2 Si $w_1 \equiv_G w_2$, entonces $w_1 \equiv_{G'} w_2$ y $\varphi(w_1) = \varphi(w_2)$.

2.2.1 Construcción del grupo G

Para la construcción del grupo $G = (X, R)$, que formará parte de la clave pública, se toma como conjunto generador $X = A \cup B \cup C$, donde los tres conjuntos A, B y C son disjuntos. Las siguientes reglas de conmutatividad $\{ab = ba : a \in A,\ b \in B\}$, junto con las básicas $\{xx^{-1} = e\}$ y con un conjunto de reglas generadas aleatoriamente $\{r_1, r_2, \dots, r_m\}$, formarán las reglas definitorias R. Entonces G es un grupo donde los elementos generados por A conmutan con los elemento generados por B.

Luego, a esta presentación, sistema de reescritura, se le aplica el algoritmo de Knuth-Bendix hasta que se hayan creado una cantidad de reglas suficientemente grande (normalmente 10 000 en la implementación actual del criptosistema). La idea es que este nuevo sistema de reescritura no llegue a ser confluente, pero sí sirva para disfrazar cualquier palabra w, mediante sustituciones de este sistema de reescritura pero tomadas de forma contraria, o sea, en este sistema de reescritura las reglas $(v \to w)$ deben de ser aplicadas de la forma $(w \to v)$.

Es muy importante aplicar el algoritmo de Knuth-Bendix a la presentación (X, R) (sistema de reescritura $< X, R >$), para que tenga la capacidad de poder disfrazar gran cantidad de palabras, idealmente todas, ya que el sistema de reescritura inicialmente creado solamente se puede aplicar a las palabras generadas mayormente. O sea, se aplica

Knuth-Bendix para llegar lo más cercano posible a $\bar{R}$ clausura reflexiva, simétrica y transitiva de R y de hecho se logra que siempre se pueda "variar" un poco una palabra, ya que las reglas triviales $e \to xx^{-1}$ (recuérdese que se revierten las reglas) son aplicables a cualquier palabra.

A continuación se describirá a manera de seudocódigo el algoritmo para generar la presentación (sistema de reescritura) de G.

Entrada : Ninguna.
Salida : Un sistema de reescritura, que sirve para disfrazar los elementos del grupo $G = (X, R)$. Donde $|X| = 50$.

1: Inicializar un sistema de reescritura vacío $< X, R >$, con $X = \{x_1, x_2, ..., x_{50}\}$, $X = A \bigcup B \bigcup C$, $A = \{x_1, x_2, ..., x_{16}\}$, $B = \{x_{16}, x_{17}, ..., x_{30}\}$, $C = \{x_{31}, x_{32}, ..., x_{50}\}$ y $R = \{\}$;
2: **for** $i := 1$ **to** 50 **do**
3: $\quad R := R \bigcup \{x_i x_i^{-1} \to e\}$;
4: **end for**
5: **for** $i := 1$ **to** 15 **do**
6: $\quad$ **for** $j := 16$ **to** 30 **do**
7: $\quad\quad R := R \bigcup \{x_i x_j \to x_j x_i\}$;
8: $\quad$ **end for**
9: **end for**
10: **for** $i := 1$ **to** 250 **do**
11: $\quad \lambda := RndWord()$;
12: $\quad \rho := RndWord()$;
13: $\quad R := R \bigcup \{\lambda \to \rho\}$;
14: **end for**
15: $< X, R >:= Knuth\_Bendix(< X, R >, 10000)$; % Aplicar Knuth-Bendix hasta tener al menos 10000 reglas%
16: Revertir todas las reglas de $< X, R >$.
17: **return** <X, R>;

Algoritmo 2.6: Generación del grupo G.

2.2.2 Construcción del grupo G'

Para la generación del grupo G', se toma como base la presentación de G y a esta se le agrega un nuevo conjunto de relaciones $S = \{\alpha_1 = \beta_1,\ \alpha_2 = \beta_2, \dots, \alpha_p = \beta_p\}$, con lo que queda $G' = (X, R \cup S)$ y $G' = G/S$.

Ahora la idea es que G' tenga un problema de las palabras relativamente sencillo, computacionalmente pueda ser resuelto de forma rápida, y que su presentación como sistema de reescritura sea **canónico**.

Para esto se agregan las siguientes reglas mientras se aplica el algoritmo de Knuth-Bendix, hasta obtener una presentación para G', cuyo sistema de reescritura sea canónico y lo "suficientemente sencillo":

- (Eliminación de un generador) $x_i = e$ para algún i.
- (Colapsar dos generadores en uno) $x_i = x_j$ para algún $i \neq j$.
- (Conmutar dos generadores) $x_i x_j = x_j x_i$ para algún $i \neq j$.

Estas relaciones extras pueden simplificar grandemente el grupo G'.

A continuación se presenta en forma de seudocódigo el algoritmo para generar la presentación (sistema de reescritura) de G'. Este algoritmo toma como entrada la presentación original de G (sin aplicarle Knuth-Bendix) y con las reglas adicionales S lo reduce a un nuevo sistema de reescritura canónico con alrededor de 200 reglas. Este algoritmo de generación puede tomar grandes cantidades de tiempo, pues quizás tenga que intentarlo con varios conjuntos para S, debido a que algunas veces este proceso falla y termina en una presentación trivial $G' = \{e\}$, o durante Knuth-Bendix se crea una cantidad muy grande de reglas, normalmente se toma como fallo cuando se generan 10000 o más reglas. Este proceso puede ser visto como el intento de hacer converger el grupo G hacia un grupo G' que cumpla con las condiciones anteriores.

Entrada : La presentación (X, R) de G.
Salida : Un sistema de reescritura canónico que sirve para resolver el problema de las palabras en el grupo $G^{'}$.

1: $NumIntentos := 100$;
2: **while** $NumIntentos > 0$ **do**
3: $R^{'} := R$;
4: $NumEliminaciones := 30$;
5: **loop**
6: **for** $i := 1$ **to** NumEliminaciones **do**
7: Generar una regla $r \notin R^{'}$, de los tipos $x_i = e$, $x_i = x_j$, o $x_i x_j = x_j x_i$.
8: $R^{'} := R^{'} \bigcup r$;
9: **end for**
10: $< X, T >:= Knuth\_Bendix(< X, R^{'} >, 200, 10000)$ %Aplicar Knuth-Bendix mientras que exista una cantidad de reglas de hasta 10000 y parar cuando existan menos de 200.%
11: **if** $0 < |T|$ **and** $|T| < 200$ **then**
12: **return** $< X, T >$;
13: **end if**
14: $NumEliminaciones := NumEliminaciones + 1$;
15: **end loop**
16: $NumIntentos := NumIntentos - 1$;
17: **end while**
18: **return null**; %No se pudo crear el grupo $G^{'}$%

Algoritmo 2.7: Generación del grupo G'.

2.3 Protocolo de intercambio de clave mutua

Un protocolo es un algoritmo multiparte, definido por una secuencia de pasos, especificando las acciones requeridas por las dos partes en orden de alcanzar el objetivo requerido. Además, un *protocolo de intercambio (o establecimiento) de clave mutua* es un protocolo mediante el cual las dos partes pueden de forma segura establecer una clave común para luego ser utilizada con fines criptográficos.

Se presenta un compacto protocolo de establecimiento puramente algebraico, para el establecimiento de una clave privada entre dos individuos, donde los cuales sólo cuentan

con un canal público de comunicación. La base del método cae sobre la dificultad de resolver ecuaciones sobre estructuras algebraicas, en particular en grupos. El protocolo requiere que ambas partes realicen cómputo algebraico (varias multiplicaciones seguidas de reescrituras en un grupo). Los resultados de la computación son entonces intercambiados entre ambas partes sobre el canal público y una *clave secreta común* es obtenida en cada parte, luego de que una segunda computación es realizada. La segunda computación requiere un algoritmo para resolver el problema de las palabras dentro un grupo.

Aquí se realiza una aplicación del protocolo generalizado de Ko-Lee (K. H. Ko, 2000). El protocolo Ko-Lee se define sobre un grupo G (con eficiente solución para el problema de las palabras) y dos subgrupos A y B de G, tal que conmuten entre sí ($ab = ba \mid a \in A, b \in B$). Los grupos A, B y G, junto con un elemento $w \in G$ son públicos. Para que Alice y Bob puedan establecer una clave mutua mediante este protocolo, deben realizar los siguientes pasos:

1. Alice selecciona un elemento privado $a \in A$ y envía a Bob el elemento $a^{-1}wa$.
2. Bob selecciona un elemento privado $b \in B$ y envía a Alice el elemento $b^{-1}wb$.
3. Alice computa $K_A = a^{-1}b^{-1}wba$ y Bob computa $K_B = b^{-1}a^{-1}wab$. Como se tiene que $ab = ba$ ($a^{-1}b^{-1} = b^{-1}a^{-1}$) en G, por tanto $K_A \equiv_G K_B$ (como un elemento de G, no necesariamente como palabras). Entonces Alice y Bob comparten la misma clave, para esto pueden resolver el problema de las palabras con K_A y K_B respectivamente obteniendo el mismo elemento canónico K, al cual luego pudieran aplicarle una *función hash*.

Se tiene que el protocolo del logaritmo discreto de Diffie-Hellman (Diffie W. , 1976) es una instancia particular del protocolo Ko-Lee.

Ahora se muestra la variación del protocolo Ko-Lee presentada en este trabajo, la cual funciona con dos grupos en lugar de uno solo. Aquí se trabajará con los grupos previamente definidos y generados en el epígrafe anterior, el grupo G (subepígrafe 2.2.1) formará parte de la clave pública y el grupo G' (subepígrafe 2.2.2).

Capítulo 2

Para Alice y Bob establecer una clave privada común mediante el nuevo protocolo descrito en este trabajo, los grupos A, B y G junto con el elemento $w \in G$ son públicos (al igual que en Ko-Lee), pero se tiene la restricción adicional de que Alice y Bob tengan previamente un grupo G' privado y compartido solamente entre ellos dos. Como se verá más adelante esta nueva restricción eleva grandemente la fortaleza del protocolo. No deja de ser un protocolo de intercambio de clave mutua, pues una misma instancia de (A, B, G, G') puede ser utilizada varias veces en el proceso de establecimiento de una nueva clave K; por ejemplo, si Alice y Bob son dos entidades diferentes que se comunican diariamente por un canal público, cada mes ellos pueden generar el cuarteto (A, B, G, G') y acordarlo privadamente (fuera del canal), el cual se podrá utilizar durante todo ese mes. Entonces esta restricción tiene el inconveniente adicional de tener que compartir un elemento secreto entre ambas partes, pero en cambio brinda un protocolo mucho más seguro mientras se realice una correcta aplicación de este, cada un tiempo cambiando el cuarteto (A, B, G, G') y brindando una seguridad alta al grupo G'.

El nuevo protocolo utilizará la función φ definida en el epígrafe anterior, la cual al aplicarla en una palabra w devuelve su representante canónico en G' (coincide con $FN_{G'}(w)$), para implementar esta función simplemente se puede aplicar el *Algoritmo 2.4.* Además se utilizará una relación δ que en se encargara de disfrazar aleatoriamente una palabra w mediante el sistema de reescritura que se generó para el grupo G (*Algoritmo 2.6*), la implementación de esta relación será descrita en el *Algoritmo 2.8.*

Además se usará una función hash $H: X^* \to Z$, la cual permitirá dar un uso criptográfico a la clave común K mediante la operación $\boldsymbol{xor}(\oplus)$**.** Para la implementación de esta función se escoge un número primo p y la evaluación de H en una palabra $w = [w_0 w_1 \dots w_{m-1}]$ es igual a $H(w) = w_0 + w_1 * p + w_2 * p^2 + \cdots + w_{m-1} * p^{m-1}$. Por tanto Alice y Bob pueden obtener exactamente la misma clave a utilizar con $\oplus$**,** mediante $H(\varphi(K_A))$ y $H(\varphi(K_B))$ respectivamente.

Entrada : Una palabra w.
Salida : El resultado de $\delta(w)$. La palabra w luego de aplicarle desordenamientos de forma aleatoria.

1: Suponiendo que se cuenta con $Matcher := build\_Aho\_Korasick(G =< X, R >)$ y que esta implementación del sistema de reescritura de G cuenta con una función $matcAll(w)$ que devuelve todas la reglas que machean con la palabra w. (Recuerdese que las reglas en $Matcher$ están en sentido contrario).
2: $NumIteraciones := rnd(25)$;
3: **for** $i := 1$ **to** NumIteraciones **do**
4: $\quad ReglasMacheadas := Matcher.matchAll(w)$;
5: $\quad k := rnd(|ReglasMacheadas|)$;
6: $\quad w := apply(w, ReglasMacheadas[k])$;
7: **end for**
8: **return** w;

Algoritmo 2.8: Implementación de la relación δ, para disfrazar palabras.

Finalmente, Alice y Bob contando el cuarteto (A, B, G, G') pueden establecer una clave privada común entre ellos con la aplicación del nuevo protocolo, aplicando los siguientes pasos:

1. Alice selecciona un elemento privado $a \in A$ y envía a Bob el elemento $a_0 = \delta(a^{-1}wa)$ (disfraza el elemento $a^{-1}wa$ mediante la relación δ y luego lo envía). El elemento enviado por Alice se denota por $a_0 (= \delta(a^{-1}wa))$, pues sería un error de notación volver a utilizar $\delta(a^{-1}wa)$, recuérdese que δ evaluada en una misma palabra puede retornar valores diferentes.
2. Bob selecciona un elemento privado $b \in B$ y envía a Alice el elemento $b_0 = \delta(b^{-1}wb)$ (disfraza el elemento $b^{-1}wb$ mediante la relación δ y luego lo envía). El elemento enviado por Bob se denota por $b_0 (= \delta(b^{-1}wb))$, pues sería un error de notación volver a utilizar $\delta(b^{-1}wb)$.
3. Alice computa $K_A = a^{-1}b_0a$ y Bob computa $K_B = b^{-1}a_0b$. Como se tiene que $ab = ba$ $(a^{-1}b^{-1} = b^{-1}a^{-1})$ en G, $a^{-1}wa \equiv_G a_0$ y $b^{-1}wb \equiv_G b_0$, por tanto $K_A \equiv_G K_B$. Utilizando el *Lema 2.2* se tiene que $K_A \equiv_{G'} K_B$ y $\varphi(K_A) = \varphi(K_B)$. Entonces Alice y

Bob calculando $H(\varphi(K_A))$ y $H(\varphi(K_B))$ respectivamente, obtienen exactamente la misma clave $K = H\big(\varphi(K_A)\big) = H(\varphi(K_B))$.

Luego de que Alice y Bob hayan obtenido la misma clave K mediante el nuevo protocolo, para que Alice le envíe un mensaje m a Bob mediante el canal público, se deben aplicar los dos siguientes pasos:

1. Alice computa $M = m \oplus K$ y se lo envía a Bob mediante el canal.
2. Bob recibe M y computa $M \oplus K = m \oplus K \oplus K = m$, con lo que obtiene el mensaje claro.

En este epígrafe se ha discutido una posible implementación de un sistema criptográfico con un protocolo público de establecimiento de clave pública, que de hecho se encuentra implementado en el criptosistema actual.

2.4 Criptosistema público: *algebraico combinatoriamente*

Detrás de un temprano fallo del criptosistema de clave pública, basado en el problema de la mochila, una gran variedad de explicaciones concernientes a este fallo comenzaron a aparecer en el "folclor" de la Criptografía. El punto de muchas de estas discusiones parecen argumentar que muchos problemas NP-completos (como la mochila) no son apropiados como una base para sistemas criptográficos, o sea, que la Criptografía *debe* estar basada en problemas de "dificultad intermedia", tales como factorizar o el logaritmo discreto.

En (Michael R. Fellows, 1993) se describe una forma completamente general en la cual problemas NP pueden servir como la base de criptosistemas de clave pública, para los cuales un ataque criptoanalítico eficiente es desconocido en la actualidad. Estos criptosistemas algebraicos combinatoriamente (*combinatorially algebraic (CA)*) parecen ser de gran interés matemático y de una alta dificultad para ser quebrados por un atacante. ¿Exactamente, que significa combinatoriamente en el contexto de Criptografía? En (Michael R. Fellows, 1993) se plantea que la distinción entre "combinatorio" y "algebraico" (o "número teórico") es artificial e improductiva. De hecho,

estos criptosistemas representan una algebratización de combinatorica, que es similar a nuevos y productivos métodos en complejidad y combinatorica (tales como (N. Alon, 2000) y (C. Lund, 1990))

A continuación se muestran la idea general de estos sistemas CA que será variada para trabajar con nuestras presentaciones de grupos. Los mecanismos de los criptosistemas CA están concernidos a ideales sobre anillos de polinomios, donde el conjunto de polinomios generando el ideal es escogido combinatoriamente.

Sea X tu problema NP-completo favorito. Para ser un poco más concretos, supongamos que se ha escogido la **3-Coloración de un Grafo** para X. Además pongamos atención en el campo F_2 de dos elementos, sobre el cual consideramos varios ideales polinomiales. Supongamos que nuestro mensaje es un simple bit (un elemento de F_2).

Ahora se brinda una visión general de cómo implementar un criptosistema de clave pública utilizando el problema X.

La *clave pública* es el grafo $G = (V, E)$.

La *clave privada* es una 3-coloración propia del grafo G.

La encriptación probabilística del mensaje $m \in F_2$ consiste en crear un polinomio q con las siguientes propiedades:

1. Si se evalúa q en una sustitución t de las variables de q, que corresponde con una solución propia de la 3-coloración del grafo G, entonces $q(t) = m$.
2. Si se evalúa q en una sustitución s de las variables de q, que no corresponde con una solución propia de la 3-coloración del grafo G, entonces el resultado de $q(s)$ es distribuido aleatoriamente.

El método de encriptación está basado en el conjunto de polinomios base $B = \{q_i\}$ (asociados canónicamente al grafo G), sobre el cual se construye el ideal $J(B)$. Cada uno de estos polinomios q_i debe cumplir que: si t corresponde con una 3-coloración propia

del grafo G entonces $q_i(t) = 0$. Por tanto, todo polinomio $q \in J(B)$, cumple la propiedad anterior, al igual que los polinomios de la base B. Para enviar un mensaje m, aleatoriamente se construye un polinomio $p \in J(B)$ y el correspondiente mensaje encriptado sería el polinomio $p + m$. El receptor del mensaje $p + m$, al evaluarlo con su clave privada t, obtiene el mensaje enviado $(p + m)(t) = p(t) + m = m$.

Para culminar con este ejemplo que servirá como punto de partida para el nuevo criptosistema de clave pública, se presentará un conjunto de polinomios que sirven como base para el problema de 3-coloración.

Sea $G = (V, E)$ la clave pública. Los polinomios de la base B están definidos sobre las variables $T = \{t_{v,i}: v \in V, 1 \leq i \leq 3\}$. Se tiene que $B = B_1 \cup B_2 \cup B_3$ donde

$$B_1 = \{t_{u,1} + t_{u,2} + t_{u,3} + 1 : u \in V\};$$

$$B_2 = \{t_{u,1}t_{u,2} + t_{u,1}t_{u,3} + t_{u,2}t_{u,3} : u \in V\};$$

$$B_3 = \{t_{u,1}t_{v,1} + t_{u,2}t_{v,2} + t_{u,3}t_{v,3} : (u, v) \in V\}$$

Se puede chequear fácilmente que los polinomios de B_1 y B_2 se encargan de la restricción de que un vértice tenga un único color y B_3 de que dos vértices adyacentes tengan colores diferentes.

A continuación se mostrará un nuevo criptosistema que será llamado del tipo *algebraico combinatoriamente*, por su gran parecido con estos. Su diferencia será que en lugar de utilizar un conjunto de polinomios, se utilizará un conjunto de elementos de un grupo y la estructura base será un subgrupo en vez de ser un ideal de polinomios.

La ***clave pública*** estará compuesta por el grupo G (subepígrafe 2.2.1) y un conjunto de palabras (representan elementos de G) $B = \{w_1, w_2, \dots, w_p\}$ que será utilizado como la base.

La ***clave privada*** estará compuesta por el grupo G' (subepígrafe 2.2.2) y un conjunto de palabras (representantes canónicos de G') $M = \{m_1, m_2, \dots, m_q\}$ que será utilizado para el envío de mensajes.

El conjunto de palabras B se escogerá de forma que todas representen el elemento neutro de G ($w_i \equiv_G e$) y se utilizará como base el subgrupo $< B >$ generado por B. Entonces el subgrupo $< B >$ puede ser caracterizado por ser un conjunto de palabras que representan el elemento neutro en G ($w \equiv_G e : w \in < B >$), además por el *Lema 2.2* se tiene que también representan el elemento neutro en G' ($w \equiv_{G'} e : w \in < B >$).

El conjunto de palabras M es escogido de forma tal que todas sean representantes canónicos diferentes de G' ($m_i \not\equiv_{G'} m_j : i \neq j$).

El siguiente algoritmo muestra la construcción de estos dos conjuntos:

Entrada : Ninguna.
Salida : Los conjuntos de palabras $B = \{w_1, w_2, ..., w_p\}$ y $M = \{m_1, m_2, ..., m_q\}$. Con $|B| = 100$ y $|M| = 16$.

```
1:  B := ∅;
2:  for i := 1 to 100 do
3:      loop
4:          w := RndWord();
5:          if φ(w) = e then % representa la palabra vacía%
6:              B := B ⋃ w;
7:              break;
8:          end if
9:      end loop
10: end for
11: M := ∅;
12: for i := 1 to 16 do
13:     loop
14:         w := RndWord();
15:         if φ(w) ∉ M then % su representante canónico pertenece a M%
16:             B := B ⋃ φ(w);
17:             break;
18:         end if
19:     end loop
20: end for
21: return (B, M);
```

Algoritmo 2.9: Construcción de los conjuntos B y M.

La encriptación probabilística de un mensaje $m_i \in M$ consiste en crear una nueva palabra m' que sea equivalente a m_i en G' ($m' \equiv_{G'} m_i$), con la propiedad de que al evaluarla en la función φ (definida por ser $FN_{G'}$) se obtenga el propio m_i ($\varphi(m') = m_i$).

El mensaje a ser enviado por este nuevo criptosistema es un número $i(1 \leq i \leq q)$ en lugar de un simple *bit,* normalmente se toma *q* como una potencia de 2 ($q = 2^k$) por razones de comodidad para el envío de información y en el criptosistema actual se tiene $q = 16$ (*cuatro bits*).

Para enviar el mensaje i, se escoge la palabra m_i y se le agrega al principio y al final palabras pertenecientes al subgrupo $< B >$ y luego se le aplica la relación δ (*Algoritmo 2.8*) para disfrazarla, o sea $m' = \delta(aw_ib)$ con $a, b \in < B >$. Esta relación será denotada por ρ_B y el siguiente algoritmo muestra una forma de implementarla:

Entrada : Un representante canónico m a encriptar y el conjunto $B = \{w_1, w_2, ..., w_p\}$
Salida : Una palabra m' equivalente al representante canónico m.

```
1: m' := m;
2: for i := 1 to 10 do
3:     id := rnd(p);
4:     m' := w_id + m';
5: end for
6: for i := 1 to 10 do
7:     id := rnd(p);
8:     m' := m' + w_id;
9: end for
10: m' := δ(m');
11: return m';
```

Algoritmo 2.10: Implementación de la relación ρ_B.

Se cumple que $\varphi\big(\rho_B(m_i)\big) = m_i$, ya que m_i es un representante canónico, $am_ib \equiv_{G'} m_i$ (por $a, b \equiv_{G'} e$) y $w \equiv_{G'} \delta(w)$. Por tanto la función φ junto con la relación ρ_B ("puerta de trampa (*trapdoor*)") sirve para la construcción de un criptosistema de clave pública de la siguiente forma:

- Si Alice desea enviarle a Bob el mensaje i, entonces envía la palabra $\rho_B(m_i)$ por el canal público.
- Para Bob desencriptar un mensaje recibido m' simplemente debe aplicar la función $\varphi(m')$ y con esto recuperar el representante canónico m_i y por tanto el mensaje i enviado por Alice.

En este nuevo criptosistema un mismo mensaje lo puede enviar de muchas formas por el uso de la relación δ al igual que el protocolo presentado en el epígrafe 2.3, pero en este caso se puede asegurar una cantidad infinita debido a que se puede generar cualquier cantidad de palabras mediante B.

Conclusiones parciales del capítulo

En este capítulo se presentaron dos formas posibles de utilizar la Teoría de Grupos Combinatoria en la Criptografía de Clave Pública, mediante un Protocolo de Intercambio de Clave Mutua (epígrafe 2.3) y un Criptosistema Combinatorio Algebraicamente (epígrafe 2.4).

El nuevo protocolo es una variación del protocolo Ko-Lee (K. H. Ko, 2000) para *grupos de cuerdas* (B_n), en el sentido de utilizar el mismo protocolo de mensajes entre las partes, pero téngase en cuenta que la base algebraica es completamente diferente pues inclusive se utilizan dos grupos y la encriptación es aleatoria.

El nuevo criptosistema está basado en las mismas ideas de los *criptosistemas algebraicos combinatoriamente* expuestos en (Michael R. Fellows, 1993), sin embargo se utilizan subgrupos de palabras nulas en lugar de ideales de polinomios.

La implementación de estos fue descrita de forma específica, ya que todos los algoritmos en seudocódigo tenían los parámetros exactos del criptosistema implementado actualmente. Aunque téngase en cuenta que ambos pueden ser vistos como métodos generales.

3 Resultados y discusión

En este capítulo se analizarán los resultados fundamentales de este trabajo que son el Protocolo de Clave Mutua (CAKE) y el Criptosistema Algebraico Combinatoriamente (CA). A partir de ahora estos dos serán referidos por sus siglas en inglés CAKE y CA, por cuestiones de estética y comodidad.

Se presentarán diversas razones por las cuales, los resultados obtenidos en el trabajo pueden ser considerados de importancia teórica y demuestran que la idea general para un criptosistema de clave pública propuesto por Wagner y Magyarik (Neal R. Wagner, 1985) es un tema interesante de investigación.

Luego, será presentada la actual versión de un software que tiene implementado los dos nuevos métodos criptográficos expuestos en el *Capítulo 2.* Este software esta implementado en Java y tiene un chat junto con la posibilidad de enviar archivos por la red, ambas tareas se pueden realizar con cualquiera de los métodos criptográficos: CAKE o CA.

Finalmente, se analizarán las fortalezas y debilidades de ambos métodos, así como una estimación teórica de la complejidad de diferentes ataques.

3.1 Importancia teórica de los resultados

Como se ha planteado anteriormente, la Teoría de Grupos Combinatoria no ha logrado tener un impacto decisivo en la Criptografía de Clave Pública, la cual es basada fundamentalmente en la Teoría de Números y Curvas Elípticas. Sin embargo este trabajo ha demostrado que en esta dirección el camino no ha terminado, ya que se obtuvieron dos criptosistemas utilizando el problema de las palabras en diferentes presentaciones de grupos.

Ambos criptosistemas usan la idea general de utilizar dos grupos, uno parte de la clave pública (G) y otro de la privada (G'), donde G' es una imagen homomorfa de G. Esta idea

fue propuesta por primera vez en (Neal R. Wagner, 1985) y es la base algebraica sobre la cual ambos criptosistemas se sustentan. El hecho de utilizarla, lejos de restar importancia teórica la aumenta, ya que la propuesta original es muy general y está lejos de poder ser llamada un criptosistema, y en este trabajo se logró utilizar de una forma concreta y específica. Además, muestra que simples ideas como la de Wagner y Magyarik (Neal R. Wagner, 1985) son un excelente camino para futuras investigaciones de la Teoría de Grupos Combinatoria en la Criptografía.

El nuevo protocolo CAKE presentado en el epígrafe 2.3 puede ser considerado como una extensión del protocolo Ko-Lee para *grupos de cuerdas* (K. H. Ko, 2000), hacia la aplicación sobre grupos generados mediante una presentación arbitraria. De hecho ambos protocolos (Ko-Lee y CAKE) pueden ser caracterizados por ser una "variación" del protocolo Diffie-Hellman (Diffie W. , 1976) basado en el problema de los logaritmos discretos. La nueva propuesta de protocolo brinda niveles de seguridad superiores al protocolo de Ko-Lee, debido a la presencia del grupo G', con el cual resolver el problema de búsqueda del conjugado no sería suficiente como en Ko-Lee, pues todavía se tendría que encontrar cuales son las imágenes homomorfas en G' mediante φ y recuérdese que la presentación de G' se mantiene de forma privada.

El criptosistema CA presentado en el epígrafe 2.4 puede ser llamado del tipo *algebraico combinatoriamente* al igual que los expuestos en (Michael R. Fellows, 1993) ya que la base algebraica es una estructura generada combinatoriamente. Resulta de gran importancia que este algoritmo devuelve salidas aleatorias para un mismo mensaje gracias a la relación δ (para disfrazar) y que con la inserción de las palabras pertenecientes a $< B >$ se puede lograr un factor de crecimiento entre el mensaje claro y el mensaje encriptado tanto como se quiera, inclusive se puede lograr una cantidad infinita de salidas para un mismo mensaje. Este criptosistema logra adaptar exitosamente la idea original de ideales de polinomios, hacia un subgrupo generado por palabras que representan el elemento nulo y además agrega la idea de usar los dos grupos G y G'.

En este trabajo se ha constatado en la bibliografía consultada, que el uso de la Teoría de Grupos Combinatoria en la Criptografía es un tema de investigación que está en alza en la actualidad por las diversas aplicaciones que puede tener. De hecho, en la investigación se lograron aplicar conocimientos de: Criptografía, Teoría de Grupos (Álgebra), Problemas Combinatorios (Matemática Discreta), Sistemas de Reescritura, Procesamiento de Cadenas, Análisis de la Complejidad y varios Algoritmos y Estructuras de Datos, por lo que se puede definir como una línea de investigación multidisciplinaria que utiliza conocimientos abstractos del Álgebra y la Combinatoria aplicándolos mediante novedosas técnicas dentro de las Ciencias de la Computación.

3.2 Explicación general del software implementado

Para la utilización práctica de estos dos nuevos criptosistemas se creó una aplicación visual en Java llamada “PKC” (Public Key Cryptosystem). La actual implementación de este *software* será presentada en esta sección junto con sus funcionalidades y posibles aplicaciones. La aplicación utiliza la API (Application Programming Interface) de Java que brinda cierto nivel de abstracción al programador en el uso del protocolo TCP/IP, conocida como *sockets (Kenneth L. Calvert, 2008)*.

Los principales tipos de *sockets* en TCP/IP en la actualidad son los *sockets de flujo (stream sockets)* y los *sockets de datagramas (datagrams sockets)*. Los *sockets de flujo,* popularmente conocidos como sockets TCP, utilizan TCP como la conexión punto a punto (con un IP por debajo) y proveen una conexión confiable y constante. Los *sockets de datagramas*, popularmente conocidos como sockets UDP, utilizan UDP (también con IP por debajo) y está dedicado de la mejor forma al envío de mensajes de hasta 65,500 bytes de largo.

En este software el envío de mensajes por la red se realizó utilizando el protocolo orientado a la conexión de *sockets TCP*. Gracias a esto, el software permite las funcionalidades de brindar un chat y de enviar un archivo entre dos partes (un servidor y un cliente), y lo más importante es que ambas tareas se realizan utilizando cualquiera de los dos criptosistemas presentados en este trabajo.

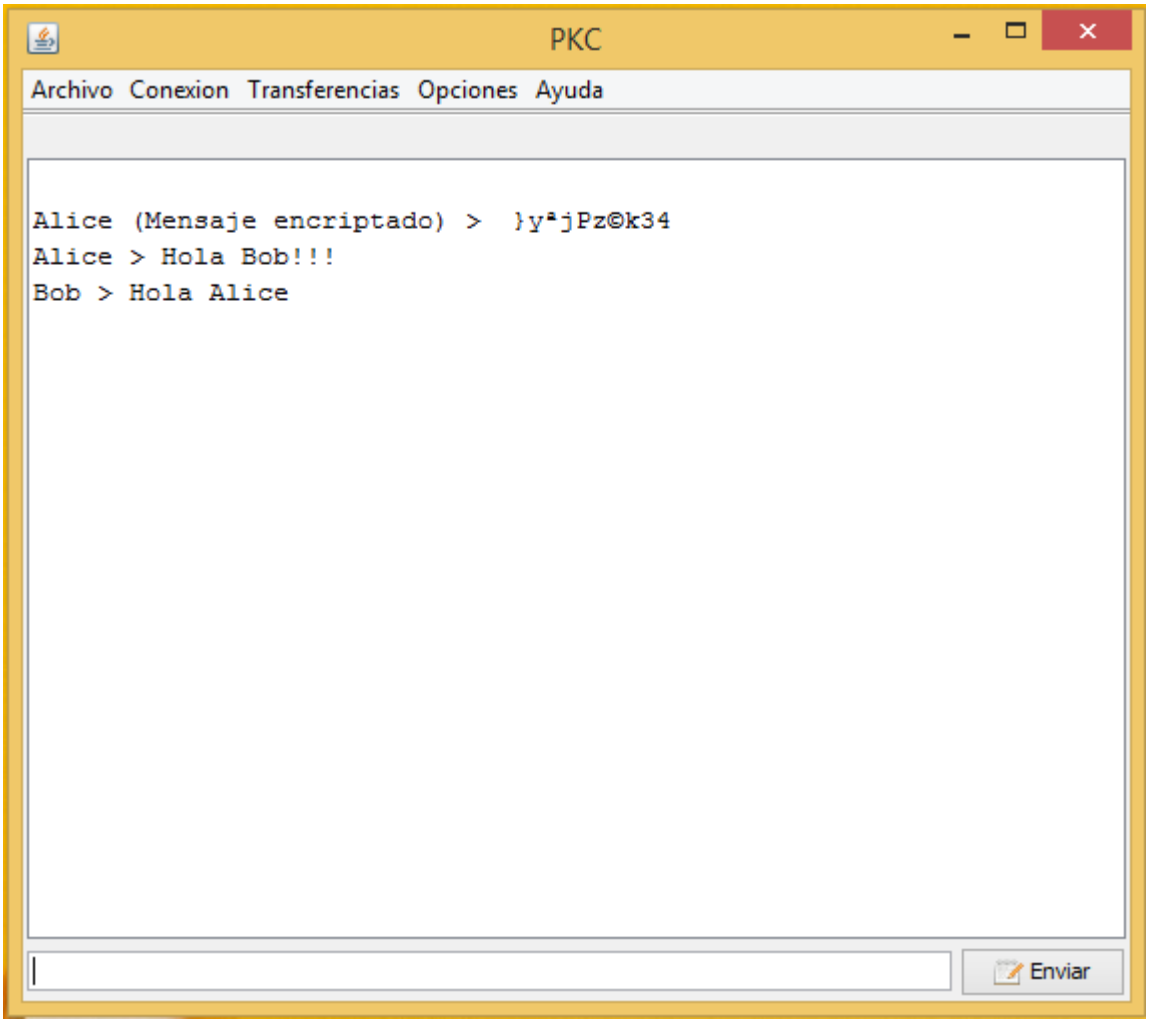


Figura 3.1: Vista general del software.

En la figura anterior se muestra una visión general del software implementado, donde están conectados dos usuarios Alice y Bob. Como se puede ver el diseño de la aplicación es sencillo, práctico y funcional, el cual consta con un área principal para el texto del chat y un serie de menús que serán explicados.

Menú Archivo: este menú está dedicado al trabajo con las claves, con una instancia de uno de los criptosistemas, y a encriptar y desencriptar archivos en la misma computadora. Las opciones que provee este menú son las siguientes:

- Cargar la clave pública que se utilizará en la conexión con la otra parte. En realidad solo se carga la presentación del grupo G, pues los otros elementos se generan al comenzar la conexión.
- Cargar la clave privada que se utilizará en la conexión con la otra parte. Es cargada solamente la presentación privada del grupo G', ya que los otros elementos son generados al principio de la conexión junto con los de la clave pública. Es de vital importancia que solo las dos partes que se comunicarán tengan acceso a la presentación del grupo G'.
- Generar nuevas claves para ser utilizadas en el criptosistema. Esta funcionalidad se encarga de la generación del par de grupos G y G'. Este es un proceso que consume bastante tiempo, ya que la generación de un par de grupos G y G' necesita como promedio 3 minutos.
- Luego de establecerse una conexión segura entre el servidor y el cliente, ambos tienen una "instancia" del criptosistema CAKE o del CA. Esta instancia puede ser guardada en la computadora, para después utilizarla como una herramienta para encriptar y desencriptar archivos sin la necesidad de existir una conexión. O sea esta funcionalidad de forma "completa" uno de los criptosistemas. En el caso de CAKE además de guardar G y G', también se guarda la clave común K, y con CA se guardan la base B y el conjunto de mensajes M junto con G y G'.
- Cargar un protocolo que se puede utilizar para la encriptación y desencriptación de archivos. Esta operación se puede realizar luego de que se haya guardado la instancia de uno de los criptosistemas con la operación anterior.
- Encriptar un archivo en la misma computadora luego de haberse cargado una instancia de un criptosistema.
- Desencriptar un archivo en la misma computadora luego de haberse cargado una instancia de un criptosistema.

Menú Conexión: este menú está dedicado al manejo de las conexiones mediante la red.

- Crear en una de las partes un servidor que será el encargado de proveer la conexión mediante el protocolo TCP con un puerto prefijado y su dirección IP. Este

servidor tiene la capacidad de esperar hasta que el cliente solicite la conexión, además de inicializar el criptosistema que se haya prefijado por el cliente.

- Mediante el IP y el puerto utilizado por el servidor el cliente puede solicitar una conexión segura mediante de uno de los criptosistemas:
    a) Solicitar una conexión mediante el Protocolo de Acuerdo de Clave Mutua (CAKE), para esto se establece la conexión, el cliente envía un paquete especial indicando que se utilizará CAKE, el servidor envía el elemento público w y finalmente se establece la clave mutua mediante CAKE.
    b) Solicitar una conexión mediante el Criptosistema Algebraico Combinatoriamente (CA), para esto se establece la conexión, el cliente envía un paquete especial indicando que se utilizará CA y el servidor genera y envía la base B.
- Terminar la conexión con la otra parte.

Menú Transferencias: este menú se encarga del envío de archivos por cualquiera de las dos partes presentes en la conexión. Cuando una de las partes solicita una transferencia, la otra parte tiene que aceptarla para que se lleve a cabo.

Menú Opciones: este menú es el encargado de manejar las preferencias de los usuarios.

- En el chat se permite mostrar el mensaje encriptado que se recibe, además del texto limpio.
- Si brinda la posibilidad de guardar el archivo encriptado recibido por la red, además de guardarse el archivo original.

Menú Ayuda: Cuenta con un "*Acerca de*" en el cual se muestra la versión actual del software y el autor.

3.3 Análisis de los criptosistemas

En esta sección se hará un análisis de los posibles ataques que se le puede hacer a los sistemas criptográficos presentados en este trabajo, su resistencia y fortaleza. Este

análisis no puede ser tomado como una validación de los criptosistemas, ya que en ambos casos se demostrará la imposibilidad de realizar un ataque por fuerza bruta. La existencia de un ataque criptográfico eficiente se va fuera de los propósitos y objetivos de este trabajo, ya que en muchos casos puede resultar ser un trabajo más complejo que crear un nuevo criptosistema. Además desde el punto de vista teórico sería difícil demostrar la no existencia de un ataque "rápido", que resulta ser una proposición bastante fuerte.

En el protocolo CAKE (epígrafe 2.3) parte de su seguridad recae sobre la dificultad de resolver el problema de la búsqueda del conjugado. Si fuera posible resolver el problema de la búsqueda del conjugado de forma eficiente entonces se pueden encontrar los elementos privados a y b, y por tanto la clave común K.

Ahora supongamos que un adversario encuentra $a_1, a_2 \in A$ tal que $a_1 w a_2 = a^{-1} w a$ y $b_1, b_2 \in B$ tal que $b_1 w b_2 = b^{-1} w b$. Entonces por las propias definiciones de A y B, se tiene que a_1, a_2 conmutan con cualquier $b \in B$, por tanto:

$$a_1 b_1 w b_2 a_2 = a_1 b^{-1} w b a_2 = b^{-1} a_1 w a_2 b = b^{-1} a^{-1} w a b = a^{-1} b^{-1} w b a = K.$$

Al enfatizar que a_1, a_2 y b_1, b_2 no tienen nada que ver con los elementos privados a y b enviados, se simplifica sustancialmente la búsqueda. En otras palabras, para encontrar la clave K, el adversario no tiene que resolver el problema de la búsqueda del conjugado, en su lugar es suficiente resolver el problema aparentemente más fácil según algunos autores, llamado *problema de descomposición:*

Dados dos elementos w y $x * w * y$ en un grupo G, encontrar cualquier elementos x' y y' que pertenezcan a un subconjunto $A \subset G$, y tal que $x * w * y = x' * w * y'$.

La afirmación de que el *problema de descomposición* debe ser más sencillo que el *problema de búsqueda del conjugado* es intuitivamente clara, ya que generalmente es más sencillo resolver una ecuación con dos incógnitas que un caso especial de la misma ecuación con una sola incógnita.

Sin embargo la dificultad de tratar de resolver el problema de descomposición en el grupo *G* es un problema que recae sobre la capacidad de desordenación de la relación δ y la incapacidad de "deshacer esta desordenación" por parte de un atacante. Ambas condiciones son alcanzadas ya que el grupo G cuenta con cerca de 10000 reglas, con la posibilidad de aplicar al menos una regla cada palabra w y con una aplicación aleatoria de estas es posible dar una gran cantidad de palabras para la misma w (incluso infinita). Por último el sistema de reescritura generado por la presentación de G al aplicarle Knuth-Bendix se trata de dejarlo lo suficientemente "complicado" como para que no sea canónico (confluente y noetheriano) y que ninguna o pocas características sobre la estructura del grupo G puedan ser develadas mediante su compleja presentación.

El análisis anterior de la relación δ también se aplica a su utilización en el criptosistema CA. Por tanto, se puede llegar a la conclusión que tratar revertir la difusión hecha por δ en cualquiera de los dos criptosistemas CAKE y CA, mediante fuerza bruta parece ser vía sin esperanzas para el atacante. Básicamente esto puede ser visto como un símil al problema de un cubo de Rubik, pues luego de hacerle unas pocas desordenaciones resulta increíblemente difícil llevarlo a su estado inicial.

Para un ataque al criptosistema CA también hay que tener en cuenta la información "basura" generada por el conjunto de palabras base B, que también se realiza de una forma aleatoria, con lo que se aumenta la capacidad de disfrazar los elementos. El criptosistema CA actual deja factores de crecimiento de 1 a 4000, por ejemplo un texto de solamente 13 bytes como "Hola Mundo!!!" se convierte en un mensaje de cerca de 66.7KB= 68300.8 bytes. Entonces por el carácter aleatorio de la encriptación (difusión), un mensaje de largo de n-bits puede tener una cantidad gigantesca de posibles salidas, idealmente en el orden de $O(2^{4000*n})$. La cota anterior para las posibles salidas nunca se alcanzará, pero si sirve como una posible estimación del tamaño y se pudiera esperar exponencial de acuerdo a n.

Los anteriores análisis tanto a CAKE como a CA, han estado sobre la base de la existencia únicamente del grupo G, o sea se ha analizado la fortaleza de los

criptosistemas al aplicar el proceso de difusión o desordenación mediante la relación δ. Sin embargo la mayor dificultad para romper uno de los sistemas criptográfico proviene de la función φ y la presentación privada del grupo G'. Téngase en cuenta que en ambos criptosistemas se trabaja con el valor retornado por la función φ para resolver el problema de las palabras, pero un atacante en orden de poder alcanzar este valor debe conocer la presentación privada del grupo G'.

La presentación del grupo G' recuérdese que es un cociente de la presentación de G, ya que al conjunto original R se le agrega un conjunto adicional de relaciones S, con lo que $G' = G/S$. Un ataque de fuerza bruta para encontrar este conjunto adicional S tendría muy pocas esperanzas de triunfar, pues este conjunto puede ser completamente arbitrario y de hecho es grande el sentido de que reduce una presentación no canónica con cerca de 10000 reglas, a una canónica con solamente alrededor de 200 reglas.

La implementación actual del software da la posibilidad de generar aleatoriamente presentaciones de G y G', para ser utilizadas como un par de llaves pública y privada, respectivamente. Un posible método de utilización de estos criptosistemas sería cambiar periódicamente las claves (G y G'), con lo que se puede alcanzar grados de seguridad altos ya que un atacante estaría obligado a realizar el Criptoanálisis con poco tiempo. Por ejemplo, supongamos que Alice y Bob se comunican todos los días, entonces estos semanalmente pueden acordar generar unas nuevas claves y utilizarlas. El tipo de conexión determina el tiempo de cambio de las claves, por ejemplo, para conexiones que requieran una seguridad alta pudiera utilizarse una clave distinta en cada conexión.

3.4 Conclusiones parciales del capítulo

Se expusieron y analizaron los criptosistemas creados y el software implementado como los principales resultados del trabajo. Se discutió la importancia teórica y la fortaleza de los dos criptosistemas propuestos, y el sistema experimental parece ser consistente y resistente a ataques criptoanalíticos iniciales. Con los resultados expuestos en este capítulo se logra dar una respuesta positiva a la aplicabilidad de la Teoría de Grupos Combinatoria en la Criptografía.

Conclusiones

Con los resultados de este trabajo se puede llegar a las siguientes conclusiones:

- La Teoría de Grupos Combinatoria puede ser una potente base para la creación de nuevos algoritmos criptográficos y el proceso de aplicar la “difusión” a los elementos de un grupo parece ser la forma idónea.
- Con la creación de los dos nuevos algoritmos criptográficos se demuestra que la idea original de Wagner y Magyarik (Neal R. Wagner, 1985) puede ser utilizada como una base algebraica en la Criptografía de Clave Pública, aunque por si sola es muy general y dista de poder ser utilizada como un Criptosistema.
- Existen varios problemas en la Teoría de Grupos Combinatoria que pueden ser utilizados como fuertes candidatos en la construcción de nuevos Criptosistemas, pero no todos pueden ser implementados.
- La aplicación implementada recoge las funcionalidades de los dos nuevos criptosistemas y permite la generación de nuevas presentaciones de grupos.
- Se logró demostrar que ataques de fuerza bruta son impensables para los dos criptosistemas implementados.

Recomendaciones

- Adaptar el Criptosistema Algebraico Combinatoriamente para que se pueda utilizar como un Sistema de Firma Digital.
- Utilizar el Protocolo de Intercambio de Clave Mutua para establecer la clave privada en Criptosistemas Simétricos como: AES o DES.
- Realizar un análisis riguroso sobre la fortaleza de los dos Criptosistemas implementados que abarque más allá de la fuerza bruta. Ambiciosamente, pudiera ser demostrar que los ataques serían problemas NP-completos.
- Ampliar el uso de las ideas expuestas en este trabajo, adaptando Criptosistemas para que usen los grupos G y G' como base algebraica. Por ejemplo, una versión de RSA pudiera ser desarrollada con el cálculo del orden de cada palabra w en el grupo G'.

Referencias bibliográficas

A. CHURCH, J. R. 1939. Some properties of conversion. *Trans. Am. Math. Soc. 39* 472-482.

ADRIAN VLADU, C. N. 2005. Suffix arrays : a programming contest approach. *GInfo 15/7*.

AHO, A. V. M. J. C. 1975. Efficient string matching: An aid to bibliographic search. *Communications of the ACM 18* 333–340.

ANSHEL I., A. M., GOLDFELD D. 1999. An algebraic method for public-key cryptography. *Math. Res. Lett. 6***,** 287-291.

BIRGET, J.-C., MAGLIVERAS, S. S. & SRAMKA, M. 2005. On public-key cryptosystems based on combinatorial group theory. *IACR Cryptology ePrint Archive,* 2005**,** 70.

BRIANDAIS, R. D. L. 1959. File searching using variable length keys. *In Proceedings of the Western Joint Computer Conference***,** 295-298.

C. LUND, L. F., H. KARLOFF, N. NISAN 1990. Algebraic methods for interactive proof systems. *IEEE Symposium on Foundations of Computer Science***,** 2-10.

CASTILLO, C. I. 1999. Teoria de Numeros.

CASTILLO, C. I. 2001. Algebra.

CASTILLO, C. I. 2002. Algebra Homologica y Algebra Conmutativa.

CASTILLO, C. I. 2005. Representacion de grupos.

D.E KNUTH, P. B. B. 1970. Simple Word Problems in Universal Algebras. *Computational Problems in Abstract Algebra. Pergamon Press.***,** 263-297.

DIFFIE W. , M. E. H. 1976. New directions in cryptography. *IEEE Transactions on Information Theory 22***,** 644-654.

DINESH P. MEHTA, S. S. 2005. *Handbook of DATA STRUCTURES and APPLICATIONS*.

DONALD E. KNUTH, J. H. M., JR., VAUGHAN R. PRATT 1977. FAST PATTERN MATCHING IN STRINGS. *SIAM J. COMPUT.,* 6.

F. RODRÍGUEZ-HENRÍQUEZ, N. A. S., A. DÍAZ PÉREZ Y Ç. K. KOÇ. 2006. Cryptographic Algorithms on Reconfigurable Hardware. *Springer First Edition* [Online].

FREDKIN, E. 1960. Trie Memory. *Comm. ACM* 490-499.

GARRETT, P. 2001. *Making, breaking codes: an introduction to cryptology*.

HERSTEIN, I. N. 1996. *Abstract Algebra*, Prentice-Hall.

HUET, G. 1980. Abstract properties and applications to term rewriting systems. *J. Assoc. Cornput. Mach.* **,** 797-821.

HUET, G. 1981. A Complete Proof of Correctness of the Knuth-Bendix Completion Algorithm. *Inria, Domaine de Voluceau-Rocquencourt, Le Chesnay France*.

HUNGERFORD, T. W. 2000. *Algebra*, Springer-Verlag.

INSTITUTE FOR STUDIES IN THEORETICAL PHYSICS AND MATHEMATICS, I. 2010. Iran. Available: http://math.ipm.ac.ir.

K. H. KO, S. J. L., J. H. CHEON, J. W. HAN, J. KANG, C. PARK 2000. New public-key cryptosystem using braid group. *Advances in cryptology - CRYPTO 2000 LNCS Springer***,** 166-183.

KENNETH L. CALVERT, M. J. D. 2008. *TCP/IP Sockets in Java, A practical guide for programmers.*

KILPELAINEN, P. 2003. Set Matching and Aho-Corasick Algorithm. Kuopio, Finlandia: University of Kuopio.

KONHEIM, A. G. 2007. *Computer Security and Cryptography*, John Wiley & Sons.

LANG, S. 2000. *Algebra*, Springer-Verlag.

M.I. GONZÁLEZ VASCO, C. M., R. STEINWANDT 2002. Toward a uniform description of several group based cryptographic primitives. *Cryptography ePrint Archive, Report 2002/048.*

MARKOV, A. A. 1947. On the possibility of certain algorithms in the theory of associative systems. *Dokl. Akad. Nauk. SSSR, 55***,** 583-586.

MAXIME CROCHEMORE, C. H., THIERRY LECROQ 2007. *Algorithms on Strings*.

MAXIME CROCHEMORE, W. R. 1997. Text Algorithms

MENEZES, A. J. V. O., P. C. VANSTONE, S. A. 1997. Handbook of Applied Cryptography. CRC Press.

MICHAEL R. FELLOWS, N. K. 1993. COMBINATORIAL CRYPTOSYSTEMS GALORE. *AMS Contemporary Mathematics Series*.

MOSER, L. 2004. *An Introduction to the Thory of Numbers*, Trillia Group.

MULLEN, G. L. & PANARIO, D. 2013. Handbook of Finite Fields. *Discrete Mathematics and its Applications. Chapman & Hall/CRC, Boca Raton (to appear)*.

N. ALON, M. T. 2000. Chromatic numbers and orientations of graphs. *Combinatorica*.

NEAL R. WAGNER, M. R. M. 1985. A public-key cryptosystem based on the word problem. *Proc. Advances in Cryptology - CRYPTO'84, LNCS 196, Springer-Verlag (1985)***,** 19-36.

NIST 2001. Announcing the Advanced Encryption Standard (AES). Federal Information Standards Publication.

ROGER C. LYNDON, P. E. S. 1977. *Combinatorial group theory*, New York, Springer-Verlag.

SHPILRAIN VLADIMIR, Z. G. 2006. Combinatorial group theory and public key cryptography. *Applicable Algebra in Engineering, Communication and Computing,* 17**,** 291-302.

THOMAS H. CORMEN, C. E. L., RONALD E. RIVEST, CLIFFORD STEIN 2001. *Intoduction to algorithms*, The Massachusetts Institute of Technology.

THUE, A. 1914. Probleme über Veranderungen einer Zeichenreihen nach gegebenen Regeln. *Kra. Vidensk. Selsk. Skrifter I. Mat. Nat. Kl. No. 10*.

TOPCODER. 2003. *Range Minimum Query and Lowest Commom Ancestor* [Online]. Available: http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor.

UDI MANBER, G. M. 1991. Suffix arrays: A new method for on-line string searches. *Department of Computer Science, University of Arizona*.

VASCO, M. I. G. 2003. *Criptosistemas Basados en Teoría de Grupos.* Tesis Doctoral, Universidad de Oviedo.

VINOGRADOV, I. M. 1954. *Elements of Number Theory*, Dover Publications, INC.

WILHELM MAGNUS, A. K., DONALD SOLITAR 1966. *Combinatorial group theory presentations of groups in terms of generators and relations,* New York, Interscience Publishers.