CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL INSTITUTO POLITÉCNICO NACIONAL

Unidad Zacatenco
Departamento de Computación

Autenticación de usuarios basada en el problema de homomorfismo de gráficas

Tesis que presenta:
Jesús Bastida Granados

Para obtener el grado de:
Maestro en Ciencias en Computación

Director de la Tesis:
Guillermo Benito Morales Luna

México, D.F. Septiembre, 2014

Dedicado a mis padres

A quienes me han heredado el tesoro más valioso que puede dársele a un hijo: amor.
A quienes sin escatimar esfuerzo alguno, han sacrificado gran parte de su vida para formarme y educarme.
A quienes la ilusión de su vida ha sido convertirme en persona de provecho.
A quienes nunca podré pagar todos sus desvelos, ni aun con las riquezas más grandes del mundo.
Por esto y más... Gracias.

II

Agradecimientos

Doy gracias a mi asesor Dr. Guillermo Benito Morales Luna por aceptarme como su estudiante.

También quiero dar las gracias al CINVESTAV por la oportunidad de ser parte de esta comunidad científica.

Finalmente, quiero agradecer al CONACyT, por el apoyo económico brindado durante los primeros dos años de la maestría.

Resumen

Actualmente se considera que los protocolos criptográficos están basados en problemas matemáticos computacionalmente difíciles, sin embargo, los avances tecnológicos y científicos han permitido contar con recursos de cómputo cada vez mejores, al grado de que ciertas instancias de problemas difíciles pudieran dejar de serlo aún hoy mismo o en un futuro.

En teoría de gráficas, un homomorfismo de una gráfica $G_0(V_0, A_0)$ a otra $G_1(V_1, A_1)$ es una función $f : V_0 \to V_1$ del conjunto de vértices de G_0 al conjunto de vértices de G_1, tal que las aristas bajo el homomorfismo f se preservan, es decir para dos vértices cualesquiera $u, v \in V_0$, si u y v son adyacentes, $[uv] \in A_0$, entonces sus imágenes también lo son, $[f(u)f(v)] \in A_1$. Para dos clases de gráficas $\mathcal{G}$ y $\mathcal{H}$, se denota por HOM($\mathcal{G}$, $\mathcal{H}$) al problema de decidir cuando una gráfica $G \in \mathcal{G}$ es homomorfa a una gráfica $H \in \mathcal{H}$. Este problema se encuentra en la clase completo$-NP$, aun si H es un mero triangulo.

Una función $f : X \to Y$ es de un solo sentido si es fácil calcular $f(x)$ para cualquier $x \in X$, sin embargo, para valores de $y \in Im(f)$ seleccionados de manera aleatoria es computacionalmente inviable encontrar algún elemento $x \in X$ tal que $f(x) = y$. La existencia de funciones de un solo sentido garantiza que cada problema $S \in NP$ tiene una prueba interactiva de conocimiento nulo.

Por lo anterior, en esta tesis se presenta la implementación y desarrollo de los mecanismos necesarios para la autenticación de usuarios basada en HOM($\mathcal{G}$, $\mathcal{H}$) mediante esquemas de conocimiento nulo. El protocolo está basado en la dificultad de resolver HOM($\mathcal{G}$, $\mathcal{H}$).

Palabras clave: homomorfismo de gráficas, gráficas aleatorias, protocolos de autenticación desafío-respuesta, pruebas de conocimiento nulo.

Abstract

Nowadays cryptographic protocols are considered to be based on computational difficult problems, however the technological and scientific advances have increased, giving as a result better computing resources. This made possible that certain problems that are difficult today for certain instances, may not be it in the future.

In graph theory, a homomorphism from a graph $G_0(V_0, A_0)$ to a graph $G_1(V_1, A_1)$ is a mapping $f : V_0 \to V_1$ from the vertex set of G_0 onto the vertex set of G_1 such that for any two adjacent vertices $u, v \in G_0$, their images $f(u)$ and $f(v)$ are adjacent in G_1. We let $\mathcal{G}$ and $\mathcal{H}$ be the graph classes, and denote the corresponding homomorphism decision problem ask whether there is a homomorphism from a graph $G \in \mathcal{G}$ to a graph $H \in \mathcal{H}$ by HOM($\mathcal{G}$, $\mathcal{H}$). This prolem remains $NP-$complete even if H is a triangle.

A function $f : X \to Y$ is called one-way function if $f(x)$ is easy to compute for all $x \in X$ but for essentially all elements $y \in Im(f)$ it is computacionally infeasible to find $x \in X$ such that $f(x) = y$. If one-way function exist, any problem $S \in NP$ has a interactive zero-knowledge proof.

Therefore, in this thesis the implementation and development of the necessary mechanisms for user authentication based on HOM($\mathcal{G}$, $\mathcal{H}$) by zero knowledge schemes is presented. The protocol is based on the difficulty of solving HOM($\mathcal{G}$, $\mathcal{H}$).

Key words: graph homomorphism, random graphs, authentication challenge-response protocols, zero-knowledge proofs.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Las tecnologías de la información y comunicación se han convertido, a una gran velocidad, en parte importante de nuestras vidas. Si se entiende esta última como el conjunto de recursos, procedimientos y técnicas usadas en el procesamiento, almacenamiento y transmisión de la información.

En el ámbito de la seguridad de la información, uno de los problemas fundamentales a solventar es la necesidad de validar o garantizar que la información que enviamos llegue a su destino sin alteraciones y que aquella información que recibimos sea realmente de quien creemos que viene y no de un impostor o suplantador de identidad. Por este motivo, se requieren de mecanismos o servicios que brinden seguridad durante el proceso de comunicación. Este desafío ha generalizado los objetivos de la criptografía moderna, la cual se encarga del estudio de los algoritmos, protocolos, y sistemas de cifrado que se utilizan para proteger la información, dotar de seguridad a las comunicaciones y a las entidades que se comunican. Por otro lado, el criptoanálisis se dedica a encontrar debilidades en los sistemas criptográficos con el fin de elaborar ataques que rompan su seguridad, y la criptología es la conjunción de criptografía y el criptoanálisis.

Este capítulo tiene la finalidad de brindar al lector una breve introducción a conceptos tales como: seguridad, criptografía y protocolos criptográficos.

1.1. Nociones básicas de seguridad y criptografía

La mayor parte de los sistemas informáticos mantienen de alguna manera una relación de identidades personales (usuarios) asociadas normalmente con algún tipo de contraseña. De esta manera, se puede suponer con certeza que quien está accediendo al sistema es quien dice ser y que está autorizado para hacerlo. El proceso de autenticación es un aspecto fundamental dentro de cualquier sistema donde se requiera: un control de acceso, confirmación de procedencia, verificación de identidades, asignación de permisos, etc.

Hasta finales de los años 70, el cifrado de la información se realizaba mediante claves simétricas, es decir, la misma clave era usada para cifrar y descifrar la información.

En un esquema de cifrado simétrico, si dos entidades querían comunicarse secretamente, debían acordar una clave en común de manera segura previamente al intercambio de mensajes. Esto

representa un problema, puesto que al establecer dicha clave por un canal de comunicación inseguro, ésta podía ser hurtada por una tercera entidad. Otro problema que presenta este tipo de cifrado es la administración y distribución de claves conforme se agregan nuevos usuarios al sistema.

Entre los algoritmos de cifrado simétrico se encuentran DES (*Data Encryption Standard*) desarrollado por IBM en 1974 y AES (*Advanced Encryption Standard*) desarrollado por Joan Daemen y Vincent Rijmen adoptado como un estándar efectivo por el gobierno de los Estados Unidos el 26 de mayo de 2002.

Los problemas presentados por los esquemas de cifrado simétrico impulsaron a Diffie y Hellman en 1976 [7], influenciados por el trabajo de Ralph Merkle [8], propusieron un nuevo esquema para el establecimiento de claves, donde la idea principal era usar funciones de un solo sentido. En este tipo de esquemas, la función de cifrado debía ser fácilmente calculable por alguien que conociera la clave pública, mientras que la función de descifrado no podía ser calculada en tiempo razonable a menos que se conociera la clave privada asociada a la clave pública utilizada en el cifrado.

De esta manera surgen los esquemas de cifrado asimétrico, en el cual basta con que un usuario genere una clave privada y una clave pública para poder intercambiar sus mensajes de manera segura.

Los esquemas de cifrado asimétrico permiten cubrir ciertos aspectos de seguridad [1], tales como:

- *Confidencialidad:* Garantiza el acceso a la información sólo por personas autorizadas.
- *Integridad de los datos:* Busca mantener la información libre de modificaciones y es capaz de detectar alteraciones no autorizadas.
- *Autenticación:* Esta función se aplica a ambas entidades que establecen una comunicación al identificarse mutuamente.
- *Vinculación (no repudio):* Permite probar la participación de las partes en una comunicación, de tal manera que impide a una entidad negar compromisos o acciones anteriores.

Entre los algoritmos de cifrado asimétrico se encuentran RSA (Rivest, Shamir y Adleman) desarrollado en 1977 y la criptografía de curva elíptica (*ECC, Elliptic Curve Cryptography*) desarrollado por Neal Koblitz y Victor Miller en 1985.

Actualmente la combinación de estos dos esquemas de cifrado forman la base de los mecanismos de seguridad que se utilizan en cualquier aplicación que requiera de comunicaciones seguras.

1.2. Protocolos criptográficos

Un *protocolo criptográfico* es un algoritmo definido por una secuencia de pasos, que especifican, de manera precisa, las acciones requeridas por dos o mas entidades para llevar a cabo un objetivo de seguridad específico [1]. Con base en la funcionalidad, se pueden clasificar los protocolos dependiendo de los servicios de seguridad que cubran, entre los principales tenemos:

- *Cifrado de mensajes.* Es el proceso por el que una información legible es transformada, mediante un algoritmo, en información ilegible.

- *Firma digital.* Es el método que permite asociar la identidad de una entidad a un documento como autor del mismo.
- *Establecimiento de claves.* Es una secuencia de pasos entre dos o más participantes a través del cual, los participantes se ponen de acuerdo en el valor de una información secreta compartida. A la información secreta se le suele llamar clave.
- *Prueba de conocimiento nulo.* Establece un método interactivo para que una de las partes (probador) pueda demostrar a otra parte (verificador) que una declaración es cierta, sin revelar nada más que la veracidad de la declaración.

1.3. Motivación

Hoy en día se considera que los protocolos criptográficos están basados en problemas matemáticos computacionalmente difíciles, sin embargo, los avances tecnológicos y científicos han permitido contar con recursos de cómputo cada vez mejores, al grado de que ciertas instancias de problemas difíciles pudieran dejar serlo aún hoy mismo o en un futuro.

Por lo anterior, es necesario considerar instancias donde el problema se mantenga difícil o basarse en otro tipo de problemas que garanticen la robustez de los protocolos criptográficos.

Recientemente D. Grigoriev y V. Shpilrain propusieron esquemas de autenticación del tipo desafío-respuesta basados en problemas difíciles-NP de la teoría de graficas, tales como: el problema de homomorfismos de gráficas (*GHP: Graph homomorphism problem*), el problema de subgráficas isomorfas (*SGIP: Subgraph Isomorphism Problem*) y el problema de coloración de gráficas (*GCP: Graph coloring problem*) [9], sin embargo, no se presentaron las especificaciones de los parámetros que requieren, tampoco se mencionan las familias de gráficas a emplear y menos aún el nivel de seguridad en términos de bits que pueden ofrecer en relación a las claves utilizadas.

En esta tesis presentamos el desarrollo e implementación de un protocolo de autenticación del tipo desafío-respuesta basada en el problema de homomorfismo de gráficas. El quebrantamiento del protocolo radica en la dificultad para resolver el problema de homomorfismo de gráficas, el cual pertenece a la clase completo-NP [34].

1.4. Descripción general del problema

En teoría de gráficas, un homomorfismo de una gráfica $G = (V_0, A_0)$ a otra gráfica $H = (V_1, A_1)$ es una función $f : V(G) \to V(H)$ del conjunto de vértices de G al conjunto de vértices de H, tal que para dos vértices cualesquiera $u, v \in V_0$, si $[u, v] \in A_0$ es una arista en G, entonces $[f(u), f(v)] \in A_1$ es una arista en H o bien $f(u) = f(v)$ [2]. Si existe un homomorfismo de gráficas de G a H, entonces se dice que la gráfica G es homomorfa a la gráfica H.

Un protocolo de autenticación del tipo desafío-respuesta *(ZKP: Zero-Knowledge Proof)* presentado por Goldwasser, Micali y Rackoff [10] es un escenario donde una entidad (P), denominada *probador*, intenta probar ante una segunda entidad (V), denominada *verificador*, tener el conocimiento de un cierto secreto sin revelar ninguna información, excepto la validez de su afirmación.

A continuación se muestran las fases de las que consta un protocolo básico de autenticación del tipo desafío-respuesta:

1. *Testimonio*: Una parte P (probador) envía información al V (verificador) como prueba del conocimiento de su clave privada. Esta información y el par de claves en conjunto definen una clase de desafíos matemáticos que el probador supuestamente podría responder siempre y cuando P sea efectivamente el poseedor de tales claves.

2. *Desafío*: La parte V presenta un desafío matemático al azar a P.

3. *Respuesta*: La parte que se quiere autenticar P recibe el desafío y elabora una respuesta que envía a V. La parte V verifica que la respuesta sea correcta y en base a ello decide si P es dueño del par de claves.

Este proceso se puede repetir las veces que sea necesario por parte del verificador para determinar que efectivamente el probador es dueño del par de claves.

Los protocolos de autenticación del tipo desafío-respuesta satisfacen las siguientes propiedades:

1. *Integridad:* Si la afirmación es cierta, el verificador *honesto* estará convencido de este hecho por el probador *honesto*.

2. *Solidez:* Si la afirmación es falsa, ningún probador *tramposo* podrá convencer al verificador *honesto* que es verdad.

3. *Conocimiento nulo:* El verificador *tramposo* no puede obtener ninguna información que le permita revelar el secreto del probador *honesto*, es decir, ningún *conocimiento* se transfiere durante el protocolo.

Del trabajo realizado por Goldreich, Micali y Wigderson [11] se tiene el siguiente teorema:

Si las funciones de un solo sentido existen, entonces cada problema $S \in NP$ tiene una prueba interactiva de conocimiento nulo.

De esta manera se logra construir un protocolo de autenticación del tipo desafío-respuesta, donde cada participante elige una pareja de gráficas G_0 y G_1 como su clave pública, y su clave privada será un homomorfismo de gráficas $\phi : V(G_0) \to V(G_1)$.

Un protocolo de autenticación basado en homomorfismo de gráficas es el siguiente [2, 9]:

Probador	Verificador
1.a Escoge una gráfica (anterior) H_p y un homomorfismo de gráficas $\psi : V(H_p) \to V(G_0)$, y lo guarda para sí. Envía H_p al verificador.	1.b Recibe H_p.
	2.a Escoge aleatoriamente $b \in (0, 1)$ y lo envía al probador.
2.b Recibe b, los desafíos matemáticos en relación al valor de b son los siguientes: Si $b = 0$, entonces hace $W_p = \psi$. Si $b = 1$, entonces hace $W_p = \phi \circ \psi$.	
3.a Envía W_p al verificador.	3.b Recibe W_p y acepta según sea la respuesta: Si $b = 0$, entonces verifica que la respuesta $W_p : V(H_p) \to V(G_0)$ es un homomorfismo de gráficas. Si $b = 1$, entonces verifica que la respuesta $W_p : V(H_p) \to V(G_1)$ es un homomorfismo de gráficas.

Figura 1.1: Protocolo de autenticación básico

1.5. Metodología

La metodología seguida en este trabajo de tesis fue la siguiente:

- Iniciamos con el estudio de algoritmos para la generación de gráficas aleatorias, los cuales sirvieron de base para la construcción de las claves en el protocolo de autenticación propuesto. Los modelos estudiados para la construcción de la gráfica base fueron los siguientes: Erdös y Renyi, gráfica k-regular aleatoria y secuencia de grados aleatoria. Del estudio anterior se eligió el modelo de gráfica k-regular aleatoria.

- Después de contar con la construcción de la gráfica base, se estudiaron diferentes algoritmos que permitieran la construcción de una segunda gráfica a partir de la gráfica base y el homomorfismo de gráficas correspondiente entre éstas. Del estudio anterior, se eligieron los algoritmos de Contracción y Expansión de vértices.

- En seguida se definió la representación de gráficas más adecuada para minimizar la complejidad de almacenamiento y realizar las operaciones involucradas en el protocolo de manera eficiente, así como la codificación de nuestras claves.

- Posteriormente se estudiaron algoritmos que resuelven tipos de homomorfismo de gráficas con el fin de establecer condiciones suficientes para la generación de instancias difíciles ante este problema.

- Luego se investigo sobre los posibles ataques que se le pueden plantear a este tipo de protocolos con la finalidad de que estos ataques no tengan efecto alguno en el protocolo que se propone.

- Considerando lo anterior, se realizó la propuesta del protocolo de autenticación del tipo desafío-respuesta similar a los ya existentes en la literatura especializada.

- Como siguiente punto, se realizó una plataforma experimental que permitió definir los parámetros necesarios para la construcción de instancias difíciles en el problema de homomorfismo de gráficas.
- Finalmente, analizamos diferentes niveles de seguridad que nos brinda el protocolo propuesto en relación al tamaño en bits de las claves utilizadas.

1.6. Objetivos

El objetivo principal de la tesis es la de implementar y desarrollar los mecanismos necesarios para la autenticación de usuarios basada en el problema de homomorfismo de gráficas mediante esquemas del tipo desafío-respuesta.

Los objetivos específicos de este trabajo son:

- Proponer/diseñar un algoritmo para la generación de claves basado en el problema de homomorfismo de gráficas, buscando instancias difíciles.
- Definir la estructura de datos para dichas claves con el propósito de minimizar la complejidad del almacenamiento y realizar las operaciones involucradas en el protocolo de manera eficiente.
- Realizar el protocolo de autenticación propuesto mediante esquemas del tipo desafío-respuesta similar a los existentes en la literatura especializada.
- Plantear/describir los posibles ataques que se le pueden hacer al protocolo.
- Mostrar la seguridad que este protocolo puede ofrecer en relación al tamaño en bits de las claves empleadas.
- Estudiar el comportamiento y propiedades de la generación de gráficas aleatorias.
- Construir una plataforma experimental que permita establecer la longitud de las claves, de tal forma que se garantice la robustez del protocolo.

1.7. Estado de conocimiento

En [3] el autor presenta las bases teóricas del protocolo de autenticación del tipo desafío-respuesta, así como los tipos de problemas que se pueden emplear. Se mencionan aplicaciones/implementaciones eficientes en tarjetas inteligentes y finalmente presenta una serie de ataques que se le pueden plantear a este tipo de esquemas.

Los autores K. Arthi, N. Nandhitha y S. Emalda Roslin [4] realizan una comparación general de los protocolos y métodos de autenticación más usados; mostrando su metodología, métricas, áreas de aplicación, la forma de atacarlos, ventajas y desventajas entre cada uno de ellos.

Los protocolos mencionados son:

1. Protocolo de seguridad de contraseña remota *(SRP: Secure Remote Password Protocol)*.

2. Autenticación y acuerdos de clave en común *(AKA: Authentication and Key Agreement).*
3. Kerberos.
4. Protocolo de autenticación extensible *(EAP: Extensible Authentication Protocol).*

Los métodos mencionados son:

1. Firma digital.
2. Criptografía de clave pública.
3. Esquemas desafío-respuesta.
4. Contraseñas.

En el artículo [5] publicado por G. Hahn y C. Tardif dan una introducción al tema de homomorfismo de gráficas, la cual incluye: definiciones, ejemplos, aplicaciones, resultados de ciertos problemas interesantes, explicación de la coloración de gráficas vista como una generalización del homomorfismos de gráficas, seguido de estudios sobre las gráficas de vértices transitivos y de Cayley; y en cómo éstas se relacionan con algunos aspectos del homomorfismo de gráficas.

Herish O. Abdullah y Mohammad Eftekhari [6] proponen nuevos protocolos de autenticación del tipo desafío-respuesta basados en los problemas de homomorfismo de gráficas, isomorfismo de subgráficas y coloración de gráficas; haciendo un criptoanálisis para cada uno de ellos. Los tipos de ataques que se realizan para el esquema basado en homomorfismo de gráficas fueron estudiados y considerados para la elaboración del presente trabajo.

1.8. Organización de la tesis

El contenido de la tesis ha sido organizado en seis capítulos:

El capítulo 2 retoma los conceptos generales de la teoría de gráficas, los cuales son indispensables para la comprensión de este trabajo. También se mencionan algunos protocolos de autenticación del tipo desafío-respuesta que han sido basados en problemas difíciles de la teoría de gráficas.

El capítulo 3 tiene como propósito general mostrar la manera en que se generan las claves (pública y privada) de nuestro protocolo a partir de una gráfica base, así como la representación y codificación más adecuada para dichas claves. Por último, se mencionan las funciones de la biblioteca `igraph` que se utilizaron para desarrollar el protocolo propuesto.

El capítulo 4 muestra un ejemplo ilustrativo de todas las fases del protocolo, dónde se explica con detalle la obtención de los homomorfismos correspondientes a las gráficas involucradas en nuestro protocolo. También se describen los parámetros de control, la manera de estimar la cantidad de vértices de cada una de las gráficas y la longitud del par de claves.

El capítulo 5 expone algunos posibles ataques que se le pueden plantear a nuestro protocolo, así como el análisis de seguridad. También se sugieren parámetros para obtener diversos niveles

de seguridad. Por último, se muestra la longitud de la clave en bits en relación a los niveles de seguridad que puede ofrecer nuestro protocolo.

Finalmente el capítulo 6 contiene las conclusiones de nuestro trabajo y algunas perspectivas para el trabajo futuro.

Capítulo 2

Teoría de gráficas en criptografía

Este capítulo tiene la finalidad de brindar al lector una breve introducción a conceptos de la teoría de gráficas y en base a ésta, cómo es que se pueden construir protocolos de autenticación del tipo desafío-respuesta.

Una gráfica $G = (V, A)$ consiste de un conjunto de vértices $V \neq \emptyset$ y un conjunto de aristas $A \subseteq \{xy \in V \times V : x \neq y\}$, es decir, una *arista* es un par no ordenado de vértices distintos de V. Usualmente una arista se denota como $[x, y]$ o xy. Un *par no ordenado* es un conjunto de la forma $[x, y]$, de manera que $[x, y] = [y, x]$.

Si $[x, y]$ es una arista, entonces decimos que x y y son *vértices adyacentes* o que y es un *vecino* de x. Usualmente la adyacencia entre vértices se denota como $x \sim y$. Sea una gráfica G, los vecinos de un vértice $u \in V(G)$ es denotado por $Nei(u) = \{v \in V(G) : [u, v] \in A(G)\}$. Un vértice es incidente en una arista siempre y cuando éste sea uno de los vértices que conforman la arista. Una arista que conecta a un vértice consigo mismo se le llama *bucle o lazo*.

Se le llama *orden de una gráfica* a su número de vértices, el cual es denotado por $|V|$. Por otro lado, se le llama *tamaño de una gráfica* a su número de aristas, el cual es denotado por $|A|$. El *grado* de un vértice o *valencia* $x \in V(G)$ de una gráfica es igual al número de aristas que inciden sobre éste. El grado de un vértice se denota por $\deg(x)$. El grado máximo de una gráfica G es denotado por $\Delta(G)$ y el grado mínimo de una gráfica es denotado por $\delta(G)$.

Una gráfica G con n vértices es *completa* si todo par de vértices en G son adyacentes, es decir, cada uno de los vértices tiene grado $n - 1$. La gráfica completa de n vértices es denotada por K_n y su conjunto de aristas $A(K_n) = \frac{n(n-1)}{2}$.

Una gráfica sin aristas pero con al menos un vértice es llamada *gráfica vacía*. La gráfica sin vértices y sin aristas es llamada *gráfica nula*.

El complemento $\overline{G}$ de una gráfica G tiene el mismo conjunto de vértices que G, tal que dos vértices u y v son adyacentes $[u, v] \in A(\overline{G})$ en $\overline{G}$ si y solo si u y v no son adyacentes $[u, v] \notin A(G)$ en G.

Una *familia de conjuntos* es un conjunto F cuyos elementos son conjuntos. Una *partición* de un conjunto A es una familia P de subconjuntos disjuntos no vacíos de A, tal que la unión de los elementos de P es A.

Sea una gráfica $G = (V, A)$, un *conjunto independiente* es un subconjunto $I \subseteq V$ tal que para todo par de vértices $u, v \in I$, estos no son adyacentes $[u, v] \notin A$.

Un *conjunto independiente maximal* es un conjunto independiente que no está contenido en ningún otro conjunto independiente.

Una gráfica G es llamada *bipartita* si su conjunto de vértices puede ser particionado en dos conjuntos de vértices P_1 y P_2 tal que cada arista $[u, v] \in A(G)$ tiene un vértice extremo $u \in P_1$ y el otro $v \in P_2$.

Una *subgráfica* de una gráfica G, es una gráfica Sg tal que: su conjunto de vértices $V(Sg)$ es un subconjunto de vértices de $V(G)$ y su conjunto de aristas $A(Sg)$ es un subconjunto de aristas de $A(G)$, es decir, $V(Sg) \subseteq V(G)$ y $A(Sg) \subseteq A(G)$.

Una subgráfica Sg de una gráfica G, es una *gráfica inducida* si dados dos vértices $u, v \in V(Sg)$ estos son adyacentes $[u, v] \in Sg$ si y solo si son adyacentes en $[u, v] \in A(G)$. Cualquier subgráfica Sg inducida de una gráfica G puede ser obtenida mediante la eliminación de vértices en $V(G)$ junto con cualquier arista que incida en los vértices eliminados. De esta manera, una subgráfica inducida es determinada por el conjunto de vértices seleccionado a partir de una gráfica base.

La vecindad de un vértice $v \in G$, es la subgráfica inducida de G conformada por todos los vértices adyacentes a v y las aristas correspondientes a dichos vértices. La vecindad de un vértice se denota por $N_G(v) = (V, A)$, donde $V = Nei(v)$ y $A = \{[x, y] : x, y \in Nei(v)\ \&\ [x, y] \in A(G)\}$. Si se incluye al vértice v dentro del conjunto V, es decir, $V = Nei(v) \cup \{v\}$, entonces se dice que la vecindad es cerrada. Si $V = Nei(v)$, entonces se dice que la vecindad es abierta.

Una función $f : A \to B$ es una regla que asigna a cada elemento a de A a un único elemento b de B. Si $a \in A$ esta relacionado con $b \in B$, entonces b es llamada la imagen de a, a es llamada la preimagen de b, y es denotada por $f(a) = b$. El conjunto A es conocido como el dominio de la función f y el conjunto B es conocido como el codominio de la función f. La imagen de la función f es denotado por $Im(f) = \{b \in B : \exists a \in A, f(a) = b\}$.

Una función $f : A \to B$ es $1-1$ *(uno a uno) o inyectiva* si cada elemento en B es la imagen de a lo sumo un elemento a en A. Por lo tanto si $f(a_1) = f(a_2) \Rightarrow a_1 = a_2$.

Una función $f : A \to B$ *es sobre o suprayectiva* si cada elemento $b \in B$ es la imagen de al menos un elemento $a \in A$. De manera equivalente, para cada elemento $b \in B$ existe un $a \in A$ con $f(a) = b$, es decir, $Im(f) = B$.

Una función $f : A \to B$ es *biyectiva* si es inyectiva y suprayectiva a la vez. Si f es una biyección entre conjuntos finitos A y B, entonces $|A| = |B|$. Si f es una biyección entre un conjunto A y si mismo, entonces f es una *permutación* de A.

Sean dos funciones $g : A \to B$ y $h : C \to D$, tales que la imagen de la función h está contenida en el dominio de la función g, es decir, $Im(h) \subseteq A$. Entonces la composición de funciones $g \circ h(c) = gh(c) = g(h(c))$, hace actuar primero a la función h sobre un elemento $c \in C$, y luego g sobre la imagen que se obtenga.

La función identidad $f : A \to A$ asocia cada elemento consigo mismo, es decir, cada elemento $a \in A$ le asocia $f(a) = a$. La función identidad de A se denota como id_A. Dada una función $f : A \to B$ se tiene: $f \circ id_A = f$ y $id_B \circ f = f$. Es decir, dado un elemento $a \in A$: $a \overset{id_A}{\longmapsto} a \overset{f}{\mapsto} f(a)$ y $a \overset{f}{\mapsto} f(a) \overset{id_B}{\longmapsto} f(a)$.

Dada una función $f : A \to B$, se dice que $g : B \to A$ es la inversa o recíproca de f si se cumple: $f \circ g = id_B$ y $g \circ f = id_A$. La inversa se denota por $g = f^{-1}$, y tanto f como f^{-1} se dicen invertibles.

Toda función biyectiva f es invertible, y su inversa f^{-1} es biyectiva a su vez. Es decir, toda función invertible f es biyectiva.

Sean G y H dos gráficas. Una función $\phi : V(G) \to V(H)$ es un *homomorfismo* de G a H si este preserva aristas, es decir, si para cualquier arista $[u, v]$ de G, $[\phi(u), \phi(v)]$ es una arista de H. Nosotros simplemente escribiremos $\phi : G \to H$ para referirnos a un homomorfismo de G a H. El problema de homomorfismo de gráficas consiste en decidir si acaso hay un homomorfismo $\phi : G \to H$. Este problema se encuentra en la clase completo$-NP$, aun si H es un mero triangulo.

Si una gráfica G es bipartita, entonces la transformación del conjunto de vértices $V(G)$ a $V(K_2)$ que envía todos los vértices $u \in V_i$ hacia el vértice $i \in \{1, 2\}$ es un homomorfismo de gráficas de G a K_2.

Si $\phi : G \to H$ es un homomorfismo, entonces este produce una relación $\phi_A : A(G) \to A(H)$, $\phi_A([u, v]) = [\phi(u), \phi(v)]$, y podemos definir a la gráfica $\phi(G) = (\phi(V(G)), \phi_A(A(G)))$ como la imagen homomorfa de G en H.

Un homomorfismo suprayectivo es conocido como *epimorfismo*, un homomorfismo inyectivo como *monomorfismo* y un homomorfismo biyectivo como *bimorfismo*. Note que el inverso de un homomorfismo biyectivo no necesita ser un homomorfismo. Un homomorfismo de una gráfica con ella misma es llamado *endomorfismo*. El endomorfismo identidad en una gráfica G es denotada por id_G.

Supongamos que $\phi : G \to H$ es un homomorfismo de gráficas. Entonces ϕ es un *isomorfismo* si y solo si ϕ es una función biyectiva y si su inversa ϕ^{-1} es un homomorfismo también. En particular, si $G = H$ entonces ϕ es un *automorfismo* si y solo si este es una función biyectiva. Por lo tanto, un automorfismo es una permutación de los vértices de $V(G)$ tal que relaciona aristas con aristas. El conjunto de todos los automorfismos de G forman un grupo, el cual es denotado como $Aut(G)$. El grupo simétrico $S(V(G))$ es el grupo de todas las permutaciones del conjunto $V(G)$, entonces $Aut(G)$ es un subgrupo de $S(V(G))$. Si G tiene n vértices, entonces se puede usar indistintamente $S(n)$ o $S(V(G))$.

En general, no es trivial decidir si dos gráficas son isomorfas, es decir, el problema de decisión se encuentra en la clase NP. Por otra parte, instancias genéricas de este problema son sencillas, por que dadas dos gráficas aleatorias típicamente no son isomorfas por razones triviales. Por ejemplo: toda permutación en el conjunto de vértices de una gráfica completa K_n es un automorfismo, entonces $Aut(K_n) \cong S(n)$.

El problema de isomorfismo de subgráficas tiene como instancia un par de gráficas G y H; y ha de decidir si acaso G es isomorfa a una subgráfica de H. El problema de isomorfismo de subgráficas se encuentra en la clase NP. Sin embargo, en la práctica el problema puede ser resuelto de manera eficiente [24].

Una *k-coloración* de una gráfica G es una asignación de colores a los vértices de G tal que dos vértices adyacentes reciben colores diferentes. Una k-coloración apropiada puede verse como una función $f : V \to [k]$, con $[k] = \{0, 1, \ldots, k - 1\}$.

Una gráfica $G = (A, V)$ tiene una k-coloración si y solo si G es homomorfa a la gráfica completa K_k. Los colores muestran una partición del conjunto de vértices de la gráfica G y el homomorfismo $\psi : V \to V(K_k)$ corresponde a la identificación de vértices del mismo color.

El valor más pequeño de k para el cual G tiene una k-coloración, es conocido como el *número cromático* de G, este es denotado por $\chi(G) = min\{k|f : G \to K_k\}$. Si hay un homomorfismo $\phi : G \to H$, entonces $\chi(G) \leq \chi(H)$, es decir, no existe un homomorfismo de gráficas de G a H, donde el número cromático de H es menor que el de G.

2.1. Un esquema de autenticación básico del tipo Fiat-Shamir

Un probador P debe mostrar ante un verificador V que efectivamente ostenta la identidad digital que dice poseer en un protocolo de tres fases con t rondas.

1. *Configuración de una sola vez.*
   - Un centro de seguridad T selecciona y pública los parámetros de un protocolo del tipo RSA, con el módulo $n = pq$ manteniendo en secreto a los primos p y q.
   - Cada probador A selecciona un secreto s coprimo a n, $1 \leq s \leq n-1$, calcula $v = s^2 \ mod \ n$, y registra v con T como su clave pública.
2. *Mensajes en el protocolo.* Cada ronda t tiene tres mensajes con la siguiente forma:
   a) $A \to B : x = r^2 \ mod \ n$
   b) $A \leftarrow B : e \in \{0, 1\}$
   c) $A \to B : y = rs^e \ mod \ n$
3. *Acciones del protocolo.* Los pasos siguientes son repetidos t veces (secuencialmente e independientemente). B acepta la prueba si todas las rondas son exitosas.
   a) A escoge r de manera aleatoria (testimonio), $1 \leq r \leq n-1$, y envía $x = r^2 \ mod \ n$ a B.
   b) B selecciona un (desafío) bit $e = 0$ o $e = 1$, y envía e a A.
   c) A calcula y envía a B (la respuesta) y, si $e = 0$, entonces $y = r$, si $e = 1$, entonces $y = rs \ mod \ n$.
   d) B rechaza la prueba si $y = 0$, en otro caso acepta si $y^2 \equiv xv^e \ mod \ n$. Dependiendo de e, $y^2 = x$ o $y^2 = xv \ mod \ n$, ya que $v = s^2 \ mod \ n$.

El desafío e requiere que A sea capaz de responder a dos preguntas, una de las cuales demuestra que conoce el secreto s, y la otra es una pregunta sencilla (para un probador honesto) para prevenir engaños. Un adversario puede suplantar la identidad de A si trata de engañar al verificador seleccionando cualquier r y calculando $x = \frac{r^2}{v}$, entonces contesta al desafío $e = 1$ con una respuesta "correcta" $y = r$; pero no podría ser capaz de contestar de manera correcta el desafío $e = 0$, el cual requiere el conocimiento de una raíz cuadrada de $x \ mod \ n$. Un probador A que conozca el secreto s puede contestar con éxito ambas preguntas. Por cada ronda sólo se contesta una de las dos preguntas, por lo tanto un adversario tiene la probabilidad de $p_e = \frac{1}{2}$ de superar una ronda del protocolo siguiendo la técnica antes descrita. Para disminuir la probabilidad de engaño $p_e = 2^{-t}$ se puede aumentar la cantidad de rondas t del protocolo.

La respuesta $y = r$ es independiente del secreto s de A, mientras que la respuesta $y = rs \ mod \ n$ no proporciona información sobre s porque la selección aleatoria de r no es conocida por B.

2.2. Esquemas basados en acciones de grupos

Consideraremos problemas basados en funciones de un solo sentido: ellas son fácilmente computables, pero localizar imágenes inversas plantea problemas computacionalmente difíciles.

Sean $\varphi : A \to B$ y $\psi : B \to C$ dos funciones definidas sobre conjuntos, calculables por algoritmos de complejidad temporal polinomial.

- Si ψ es difícil de invertir y el dominio de ψ esta contenido en la imagen de φ, entonces $\varphi \circ \psi = \psi(\varphi)$ es difícil de invertir.
- Si φ es difícil de invertir, ψ es inyectiva, y el dominio de ψ contiene la imagen de φ, entonces $\varphi \circ \psi = \psi(\varphi)$ es difícil de invertir.

Sea S un semigrupo y X un conjunto. Una acción izquierda de S en X es propiamente una función $\Phi : S \times X \to X$, tal que $\forall s, t \in S,\ x \in X : \Phi(st, x) = \Phi(s, \Phi(t, x))$. Con esta acción sobreentendida, se escribe $s(x) = \Phi(s, x), \forall (s, x) \in S \times X$, por lo que se ha de tener $\forall s, t \in S, x \in X : s(t(x)) = (st)(x)$.

Un primer protocolo de autenticación es el siguiente:

Suponga una acción de un semigrupo S en un conjunto X, es decir, para cualquier $s \in S$ y $x \in X$, los elementos $s(x) \in X$ están bien definidos.

1. La clave pública de A consiste de un conjunto X, un semigrupo S y un elemento $x \in X$. La clave privada es un elemento $u = s(x)$ para algún $s \in S$ seleccionado aleatoriamente.
2. A selecciona un elemento (testimonio) $t \in S$ y envía el elemento $v = t(s(x)) \in X$ a B.
3. B selecciona un (desafío) bit $e \in \{0, 1\}$ y lo envía a A.
   - Si $e = 0$, entonces A envía (la respuesta) el elemento t a B, y B verifica si $v = t(u)$ para aceptar la autenticación.
   - Si $e = 1$, entonces A envía (la respuesta) la composición ts a B, y B verifica si $v = t(s(x))$ para aceptar la autenticación.

Un segundo protocolo de autenticación es el siguiente:

1. La clave pública de A consiste de un conjunto X, un semigrupo S y un elemento $x \in X$. La clave privada es un elemento $z = r(x)$ para algún $r \in S$ seleccionado aleatoriamente.
2. A selecciona un elemento (testimonio) $y \in X$, junto con dos elementos $s, t \in S$, tal que $s(x) = y$ y $t(y) = z$. Entonces A envía el elemento y a B.
3. B selecciona un (desafío) bit $e \in \{0, 1\}$ y lo envía a A.
   - Si $e = 0$, entonces A envía (la respuesta) el elemento s a B, y B verifica si $s(x) = y$ para aceptar la autenticación.

- Si $e = 1$, entonces A envía (la respuesta) el elemento t a B, y B verifica si $t(y) = z$ para aceptar la autenticación.

Suponga que después de una cantidad considerable de rondas de los pasos $(2) - (3)$ del segundo protocolo, ambos valores de e son encontrados. Por lo anterior, la clave privada del probador estaría comprometida. Un intruso puede suplantar al probador si y sólo si es capaz de recuperar la clave privada del probador.

Un tercer protocolo de autenticación se puede construir de la siguiente manera:

1. La clave pública de A consiste de un conjunto S que tiene una propiedad Π. La clave privada es una prueba π de que S tiene la propiedad Π. Asumimos que hay una noción de isomorfismo que preserva la propiedad Π.

2. A selecciona un isomorfismo (testimonio) φ que puede ser aplicado a S y envía el conjunto $S_1 = \varphi(S)$ a B.

3. B selecciona un (desafío) bit $e \in \{0, 1\}$ y lo envía a A.

   - Si $e = 0$, entonces A envía (la respuesta) el isomorfismo φ a B, y B verifica si $\varphi(S) = S_1$ y si φ es un isomorfismo.
   - Si $e = 1$, entonces A envía (la respuesta) la prueba π del hecho de que S_1 tiene la propiedad Π a B, y B verifica su validez para aceptar la autenticación.

Para suplantar al probador, un intruso debería ser capaz de producir pruebas de que un conjunto posee la propiedad Π. Si ésta determina un problema completo-NP la suplantación será improcedente desde el punto de vista computacional.

2.3. Protocolos de autenticación basados en gráficas

A continuación se muestra un protocolo basado en el problema de isomorfismo de subgráficas:

1. La clave pública de A consiste de dos gráficas G y G_2. La clave privada es una subgráfica G_1 de G_2 y un isomorfismo $\phi : G \to G_1$.

2. A selecciona una gráfica "intermedia" H entre G y G_2, es decir, hay una inmersión $\psi : G \to H$ y otra $\theta : H \to G_2$, con $\psi(G) = G_1$. Entonces envía la gráfica H a B (testimonio) y guarda $\psi : G \to H$ y $\theta : H \to G_2$ para sí.

3. B selecciona un (desafío) bit $e \in \{0, 1\}$ y lo envía a A.

   - Si $e = 0$, entonces A envía (la respuesta) la inmersión ψ a B, y B verifica si efectivamente ψ es una inmersión entre G y H para aceptar la autenticación.
   - Si $e = 1$, entonces A envía (la respuesta) la inmersión θ a B, y B verifica si efectivamente θ es una inmersión entre H y G_2 para aceptar la autenticación.

Un intruso podrá suplantar al probador exitosamente si y sólo si es capaz de localizar la subgráfica de G_2 que es isomorfa a G.

Como es usual en protocolos del tipo Feige-Fiat-Shamir, los pasos $(2) - (3)$ de este protocolo pueden ser repetidos una cantidad considerable de veces para evitar un falsificación exitosa de un intruso con una probabilidad no despreciable.

Protocolo basado en coloración de gráficas:

1. La clave pública de A consiste de una gráfica G k-colorable. La clave privada es una k-coloración de G, para algún k (público).
2. A selecciona un isomorfismo $\psi : G \to G_1$ (testimonio) y envía G_1 a B.
3. B selecciona un (desafío) bit $e \in \{0, 1\}$ y lo envía a A.
   - Si $e = 0$, entonces A envía (la respuesta) el isomorfismo ψ a B. B verifica si efectivamente ψ es un isomorfismo entre G y G_1 para aceptar la autenticación.
   - Si $e = 1$, entonces A envía (la respuesta) una k-coloración de G_1 a B. B verifica si efectivamente es una k-coloración valida de G_1 para aceptar la autenticación.

Dado que la gráfica G es k-colorable y G_1 es isomorfa a G, entonces G_1 es k-colarable también. Un intruso podrá suplantar al probador de manera exitosa si y sólo si es capaz de encontrar una k-coloración de G.

Protocolo basado en homomorfismo de gráficas:

1. La clave pública de A consiste de un par de gráficas G_0 y G_1. La clave privada es un homomorfismo $\phi : G_0 \to G_1$.
2. A selecciona gráfica H_p (testimonio) junto con un homomorfismo $\psi : H_p \to G_0$ y envía H_p a B y mantiene ψ para sí.
3. B selecciona un (desafío) bit $e \in \{0, 1\}$ y lo envía a A.
   - Si $e = 0$, entonces A envía (la respuesta) el homomorfismo ψ a B. B verifica si efectivamente $\psi(H_p) = G_0$ y si ψ es un homomorfismo de gráficas para aceptar la autenticación.
   - Si $e = 1$, entonces A envía (la respuesta) la composición de homomorfismos $\varphi = \phi \circ \psi$ a B. B verifica si efectivamente $\varphi(H_p) = G_1$ y si φ es un homomorfismo de gráficas para aceptar la autenticación.

El protocolo de autenticación basado en homomorfismo de gráficas puede ser quebrantado si una tercera entidad (intruso), logra construir una gráfica H'_p que sea homomorfa a G_0 y G_1, es decir, hay un homomorfismo de gráficas $\psi' : H'_p \to G_0$ y $\phi' : H'_p \to G_1$. El intruso puede interferir en la fase de testimonio y enviar H'_p al verificador, por lo tanto, puede responder a cualquier desafío por parte del verificador con éxito.

En teoría de gráficas, el producto tensorial $T = G_0 \otimes G_1$ tiene como conjunto de vértices el producto cartesiano $V(G_0) \times V(G_1)$, en donde dos vértices (a, x) y (b, y) son adyacentes en T si y solo si:

- a es adyacente con b en G_0, es decir, $[a, b] \in A(G_0)$.
- x es adyacente con y en G_1, es decir, $[x, y] \in A(G_1)$.

El producto tensorial T cumple con lo siguiente [20]:

- Hay un homomorfismo de $\psi' : V(T) \to V(G_0)$
- Hay un homomorfismo de $\phi' : V(T) \to V(G_1)$

Por lo tanto, un intruso puede suplantar la identidad del probador A con éxito mediante el producto tensorial. Note que el orden de la gráfica generada por el producto tensorial T es $|T| = |G_0| \times |G_1|$. Basta con manejar gráficas con un número de vértices dentro de un rango de valores establecidos para poder identificar un ataque como el del producto tensorial.

En 1980 Babai, Erdös y Selkow mostraron que el problema de ismorfismo de gráficas puede ser resuelto por un algoritmo de complejidad temporal lineal para la mayoría de las gráficas aleatorias [22].

Los algoritmos más eficientes para encontrar isomorfismos entre gráficas aleatorias son los siguientes:

- `Nauty`: algoritmo desarrollado por Brendan Mckay [24] que puede resolver el problema de isomorfismo de gráficas en tiempo polinomial para gráficas aleatorias con miles de vértices.
- `Bliss`: es el famoso sucesor del algoritmo Nauty. Usa las mismas ideas en general, pero con mejores heurísticas y estructuras de datos para la manipulación de gráficas. Bliss supera a Nauty en la mayoría de los casos. Este algoritmo fue desarrollado e implementado por Tommi Junttila y Petteri Kaski [25].
- `Saucy`: es un algoritmo en desarrollo. Las personas involucradas en la investigación sobre este algoritmo son: Paul T. Darga, Hadi Katebi, Mark Liffiton, Igor L. Markov y Karem Sakallah [23].

Así que si queremos usar el problema de isomorfismo de gráficas en criptografía, se debe de trabajar con una familia de gráficas muy reducida, aunque esto no garantiza que un intruso pueda usar alguna combinación de los algoritmos antes descritos para realizar un ataque exitoso.

Capítulo 3

Construcción de la pareja de claves: pública y privada

Este capítulo tiene como propósito general mostrar la manera en que se generan las claves (pública y privada) del protocolo de autenticación de usuarios del tipo desafío-respuesta que se propone, así como la representación más adecuada para dichas claves.

En la criptografía asimétrica, también llamada *criptografía de clave pública* se tiene $\{E_e : e \in K\}$, un conjunto de transformaciones de cifrado, y $\{D_d : d \in K\}$, el correspondiente conjunto de transformaciones de descifrado, donde K es el espacio de claves. Considere cualquier par asociado de transformaciones de cifrado y descifrado (E_e, D_d), y que cada par tiene la propiedad que conociendo E_e es computacionalmente inviable encontrar el mensaje en claro $m \in M$, dado un texto cifrado de manera aleatoria $c \in C$ tal que $E_e(m) = c$. Esta propiedad implica que conociendo la clave de cifrado e no es posible determinar la correspondiente clave de descifrado d.

En este tipo de esquemas de criptografía de clave pública, E_e es una función *de un solo sentido*, siendo d la información necesaria para calcular la función inversa D_d, y por lo tanto permitir el descifrado de cualquier texto cifrado con la clave e. Hablando informalmente, una función de un solo sentido puede definirse de la siguiente manera:

Una función $f : X \to Y$ es *de un solo sentido* si es fácil calcular $f(x)$ para cualquier $x \in X$, sin embargo, para valores de $y \in Im(f)$ seleccionados de manera aleatoria es computacionalmente inviable encontrar algún elemento $x \in X$ tal que $f(x) = y$. El problema del logaritmo discreto (*DLP: Discrete Logarithm Problem*) es un ejemplo de función de un solo sentido.

Bajo estos supuestos, consideremos que dos participantes: Alicia y Beto quieren establecer una comunicación de manera segura. Como primer paso, Beto genera su par de claves (e, d) y envía la clave e (conocida como *la clave pública*) a Alicia manteniendo su clave d (conocida como *la clave privada*) segura y en secreto. Este par de claves permiten asociar una identidad digital a Beto. De esta manera, Alicia puede enviar un mensaje m a Beto aplicando la transformación de cifrado determinada por la clave pública de Beto $c = E_e(m)$. Por último, Beto descifra el texto cifrado c aplicando la transformación inversa determinada únicamente por su cave privada $D_d(c) = m$ y así obtener el mensaje original enviado por Alicia.

Cabe mencionar que la clave pública tiene que distribuirse sobre las entidades que quieran comunicarse de manera confidencial con el dueño de la respectiva clave pública. Por otro lado, la

clave privada debe ser protegida por el propietario de manera que nadie más tenga acceso a ella. De esta manera, basta con que una entidad genere su pareja de claves (pública y privada) para poder intercambiar sus mensajes de manera segura. Este tipo de técnicas criptográficas garantizan que la generación de la pareja de claves (pública y privada) sea única para cada una de las entidades, de modo que hay una probabilidad baja de que se asocie una misma pareja de claves a diferentes entidades.

Cuando el remitente decide cifrar un mensaje usando la clave pública del destinatario, sólo se puede obtener el descifrado de dicha cifra con la correspondiente clave privada del destinatario, por lo que ninguna otra entidad podrá conocer el contenido original del mensaje.

Cuando el propietario de una pareja de claves (pública y privada) decide cifrar un mensaje usando su clave privada, cualquier entidad puede descifrar esta cifra utilizando la correspondiente clave pública. En consecuencia, se puede autenticar e identificar al remitente, puesto que en teoría sólo el pudo cifrar el mensaje con su clave privada. En este caso se dice que se *firmó el mensaje* en lugar de se cifró el mensaje y se denota como *verificar el mensaje* en lugar de descifrar el mensaje. El proceso anterior se le conoce como firma electrónica.

Cabe destacar que si una entidad logra obtener la clave privada de algún tercero, por ejemplo la de Beto, entonces dicha entidad podrá hacerse pasar por Beto. A esta acción se le conoce como *suplantación de identidad*.

Hoy en día, el uso de muchos de los servicios en Internet, tales como envío de correo electrónico, banca en línea o realizar cualquier tipo de trámite a través de Internet, requiere demostrar primero que somos quienes decimos ser. A este proceso de probar nuestra identidad se le conoce como autenticación. La autenticación se realiza mediante el uso de algo que se conoce (como una contraseña), algo que se tiene (como una credencial), o alguna característica única (como un escáner de retina o un lector de huellas dactilares). Una de las formas más comunes de autenticación se lleva a cabo mediante el uso de una cuenta/nombre de usuario y su respectiva contraseña.

El capítulo está organizado de la siguiente manera:

En la sección 3.1 se muestra la especificación de la pareja de claves (pública y privada), así como la derivación de las gráficas principales a partir de una gráfica base.

En la sección 3.2 se menciona como se construye la gráfica base a partir del modelo para la generación de gráficas aleatorias de *Erdös-Rényi (n, p)* y gráficas aleatorias k-regulares.

Los algoritmos claves para la generación de las gráficas principales y sus correspondientes homomorfismos, así como un ejemplo ilustrativo del funcionamiento de cada algoritmo son mostrados en la sección 3.3.

Definir la representación más adecuada para las claves nos permite minimizar la complejidad del almacenamiento y realizar las operaciones involucradas en el protocolo de manera eficiente. Por ello, en la sección 3.4 se habla sobre los tipos de representación de gráficas que se tomaron en consideración para cada una de las fases del protocolo propuesto y la manera en que la clave pública puede ser codificada se menciona en la sección 3.5.

Por último, en la sección 3.6 se mencionan las funciones de la librería `igraph` que se utilizaron para desarrollar el protocolo propuesto.

3.1. Especificación de las claves

En la presente tesis se propone que la generación del par de claves (pública y privada) sea de la siguiente manera:

- Una pareja de gráficas G_0 y G_1 como la clave pública.
- Un homomorfismo de gráficas $\phi : V(G_0) \to V(G_1)$ como la clave privada.

El protocolo de la Figura 1.1 puede ser quebrantado si una tercera entidad (intruso) logra construir una gráfica H'_p que sea homomorfa a G_0 y G_1, es decir, hay un homomorfismo de gráficas $\psi' : V(H'_p) \to V(G_0)$ y $\phi' : V(H'_p) \to (G_1)$. De esta manera, el intruso puede interferir en la fase de testimonio y enviar H'_p al verificador y por lo tanto podrá responder con éxito a cualquier desafío planteado por el verificador. La idea anterior es de vital importancia, ya que con base en ésta, se planteó un ataque exitoso mediante el uso del producto tensorial en gráficas. Para más información puede consultar [6].

Por lo anterior, es necesario contar con una serie de mecanismos que nos permitan construir:

- Para la clave pública: una gráfica base G_b, con la cual se pueda derivar la construcción de las gráficas principales G_0 y G_1, mediante los algoritmos de Contracción y Expansión de vértices.
- Para la clave privada: de manera simultanea a la construcción de la clave pública, construir el homomorfismo correspondiente entre las gráficas empleadas por los algoritmos antes mencionados.
- Para el desafío: dados cualesquiera dos homomorfismos ϕ y ψ en el protocolo, obtener la composición de homomorfismos $\phi \circ \psi$.
- Para la verificación: contar con un mecanismo que nos permita comprobar que efectivamente la información presentada por el probador es un homomorfismo entre las gráficas solicitadas.

Cabe resaltar que hay tres combinaciones posibles para la derivación de las gráficas principales a partir de la gráfica base, las cuales se mencionarán a continuación:

1. $G_b \to G_0 \to G_1$, $\alpha : V(G_b) \to V(G_0)$, $\beta : V(G_0) \to V(G_1)$ y $\phi = \beta$.
2. $G_0 \to G_b \to G_1$, $\alpha : V(G_0) \to V(G_b)$, $\beta : V(G_b) \to V(G_1)$ y $\phi = \beta \circ \alpha$.
3. $G_0 \to G_1 \to G_b$, $\alpha : V(G_0) \to V(G_1)$, $\beta : V(G_1) \to V(G_b)$ y $\phi = \alpha$.

De las posibles opciones antes mencionadas, se tomó en consideración la opción número 2, ya que nos permite generar una clave privada más elaborada que las otras opciones. Ésta consiste de la composición de homomorfismos ($\phi = \beta \circ \alpha$), mientras que en las otras dos opciones solo se cuenta con un homomorfismo simple (α o β). A su vez se pueden emplear los algoritmos de Expansión y Contracción de vértices para la construcción de G_0 y G_1 respectivamente, mientras que en la opción 1 sólo se puede emplear Contracción de vértices y en la opción 3 sólo se puede emplear Expansión de vértices.

Por otro lado, se desea que las gráficas involucradas en el protocolo propuesto sean equiparables, es decir, que el número de vértices y aristas de cada gráfica se mantenga siempre dentro de un rango de valores definido para cada nivel de seguridad correspondiente a la longitud de las claves empleadas. Todo lo anterior con el fin de mantener un control más estricto sobre la creación e identificación de nuestras claves y, en todo caso, poder identificar un posible ataque (*como el del producto tensorial*) de un intruso que haga uso de gráficas fuera del rango de valores permitido.

3.2. Construcción de la gráfica base

El modelo de *Erdös-Rényi (n, p)* fue uno de los modelos elegidos para la generación de gráficas aleatorias, con el cual se puede construir una gráfica G_b que sirva como base. Este modelo nos permite obtener gráficas aleatorias con un número de vértices igual a n y cada arista es incluida en la gráfica con una probabilidad p independientemente de cualquier otra arista.

Sabemos que el número de aristas N_a en una gráfica completa K_n está definido por:

$$N_a = \binom{n}{2} = \frac{n(n-1)}{2}$$

donde n es el número de vértices de la gráfica. Entonces, en el modelo de *Erdös-Rényi (n, p)* dado un número de vértices n determinado y el número de aristas a, se tiene que la expresión analítica de la probabilidad es:

$$p = \frac{a}{N_a} = \frac{2a}{n(n-1)}$$

Despejando el número de aristas se tiene:

$$a = pN_a = \frac{pn(n-1)}{2}$$

El resultado anterior es importante debido a que se puede controlar el crecimiento del número de aristas en relación al número de vértices en una gráfica aleatoria generada por el modelo antes mencionado. Lo anterior es la razón más importante por la que se tomó en cuenta este modelo.

Otro modelo que se considero fue el de generación de gráficas aleatorias k regulares, donde cada vértice tiene la misma cantidad de vecinos, es decir, cada vértice tiene el mismo grado o valencia.

Sabemos por el *lema Quien vive* (*Handshaking Lemma*) que para toda gráfica, la suma de los grados de todos los vértices es igual a dos veces el número de aristas. Por lo tanto, el número de aristas $A(G_k)$ de una gráfica k regular G_k con n vértices está definida por:

$$A(G_k) = \frac{nk}{2}$$

En consecuencia, si la gráfica G es k regular y su número de vértices n es impar, entonces k sólo puede ser par.

Este modelo es aun mejor que el de *Erdös-Rényi (n, p)*, debido a que cada uno de los vértices tiene el mismo grado, el control sobre el número de vértices tras la aplicación del algoritmo Expansión de vértices es un poco más sencilla. La comparación entre los dos métodos será descrita en el siguiente capítulo.

3.3. Algoritmos para la construcción de las claves

A continuación presentamos dos algoritmos que nos permiten la construcción de una gráfica especial a partir de una gráfica base y su correspondiente homomorfismo de gráficas. El algoritmo que expande vértices, nos permite construir una gráfica G_0 y un homomorfismo de gráficas $\alpha : V(G_0) \to V(G_b)$. Mientras que el algoritmo contracción de vértices, nos permite construir una gráfica G_1 y un homomorfismo de gráficas $\beta : V(G_b) \to V(G_1)$. Por lo que el homomorfismo de gráficas requerido $\phi : V(G_0) \to V(G_1)$ se consigue mediante la composición de homomorfismo de $\beta \circ \alpha$.

3.3.1. Expansión de vértices

El funcionamiento del algoritmo que expande vértices se basa principalmente en la selección de una subgráfica inducida a partir de una gráfica base. Dadas la gráfica base G_b y una subgráfica inducida G_0 de G_b, existe un homomorfismo $f : V(G_0) \to V(G_b)$, con $f(x) = x$ para cualquier vértice $x \in V(G_0)$.

Algoritmo: *Expansión de vértices*:

- Entrada: Una gráfica base $G_b(V_b, A_b)$, el número de vértices a generar por cada vértice a expandir nvs y la cantidad de vértices diferentes m que se van a seleccionar de G_b.
- Salida: Una gráfica $G_0(V_0, A_0)$ y un homomorfismo de gráficas $\alpha : V_0 \to V_b$.

1. Seleccionar un subconjunto de vértices $K \subseteq V_b$, con $m = \text{card}(K)$.
2. Para cada uno de los vértices $u_i \in K$, con $1 \leq i \leq m$. Hágase lo siguiente:
   a) Obtener los vecinos del vértice $u_i \in V_b$ con sus correspondientes aristas en G_b y agregarlo a la gráfica G_0.
   b) Agregar un conjunto de nvs_i vértices a la gráfica G_0 y conectarlos aleatoriamente con los vecinos del vértice $u_i \in V_b$.
3. Para cada uno de lo conjuntos de vértices antes generados nvs_i, $nvs_j \in V_0$, con $1 \leq i, j \leq m$ y $j \neq i$. Hágase lo siguiente:
   a) Agregar aristas de manera aleatoria a G_0 entre los conjuntos de vértices nvs_i y nvs_j, donde exista una arista $[u_i, u_j] \in G_b$.

A continuación se mencionarán las consideraciones que se tomaron en cuenta para la implementación del algoritmo descrito anteriormente:

1. En el punto 3 del algoritmo, dado que las aristas son creadas aleatoriamente, no siempre se logran cubrir todas las posibles opciones para crear una nueva arista, por lo que el número de veces que se repite el punto 3 define un parámetro de control en el incremento de las aristas dejando al número de vértices constante.

2. Al finalizar el algoritmo se hace un ordenamiento de los identificadores de los vértices pertenecientes a la gráfica de salida, debido a que se eliminan aquellos vértices que no inciden sobre ninguna arista, es decir, son vértices aislados.

El número máximo de aristas que se puede construir mediante la técnica antes mencionada es la siguiente:

- Al expandir cada uno de los vértices seleccionados se obtienen:

$$nvg = m \times nvs$$

- donde:
  - m es la cardinalidad del conjunto de vértices seleccionados.
  - nvs es el número de vértices que se obtendrán al expandir un vértice de la gráfica G_b.
  - nvg es el número total de vértices generados.
  - $Nvs = \bigcup_{i=1}^{i=m} Nei(u_i)$ es el conjunto que contiene la unión de los vecinos de cada vértice $u_i \in K \subseteq V_b$.

- Se deben de incluir las aristas de la gráfica G_b a la gráfica G_0 en donde los vértices que inciden sobre una arista en A_0 son vértices en Nvs , es decir, si existe una arista $[x, y] \in A_b$ y los vértices $x, y \in Nvs$ pertenecen a la unión de los vecinos de cada vértice en K, entonces tal arista $[x, y] \in A_0$ debe añadirse en la gráfica G_0.

  Al número de aristas antes descrito lo llamaremos nas.

- Las aristas construidas aleatoriamente sobre los vértices $x, y \in Nvs$ son:

$$A_{vexp} = \frac{nvg \times (nvg-1)}{2}$$

Por lo tanto, el número total de aristas que se puede obtener como máximo es:

$$A_T = nvg + nas + A_{vexp}$$

El número total de vértices que se puede obtener como máximo es:

$$V_T = |V| + nvg$$

Donde $|V|$ es el orden o número de vértices de la gráfica G_b base que se tomo como entrada. Es importante mencionar que los valores máximos en el número de vértices y aristas se puede alcanzar si y sólo si el conjunto de vértices de la unión de los vecinos en cada vértice $u_i \in K$ es igual al conjunto de vértices de la gráfica base, es decir, $\text{card}(Nvs) = \text{card}(V_b)$.

3.3.2. Funcionamiento del algoritmo: Expansión de vértices

A continuación se mostrará un ejemplo del funcionamiento del algoritmo antes descrito:

Como primer paso se tiene la construcción de la gráfica base G_b, la cual fue obtenida mediante el modelo de Erdös-Rényi $(n,\ p)$ con un número de vértices $n = 10$, número de aristas $a = 25$ y ,por lo tanto, $p = \frac{2\times a}{n\times(n-1)}$ es la siguiente:

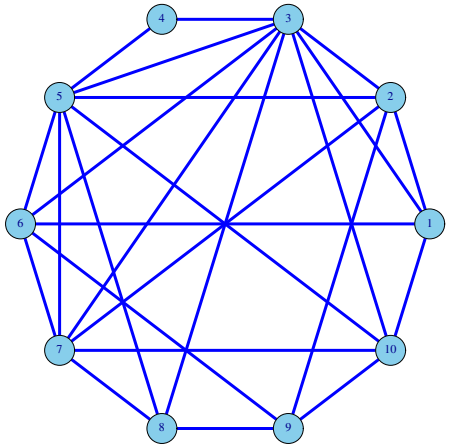


Figura 3.1: Gráfica G_b con 10 vértices y 25 aristas

Seleccionamos el subconjunto de vértices K de manera aleatoria. Para este ejemplo se tomó $K = \{1, 4, 10\}$ con $m = 3$. Procedemos a calcular los vecinos de cada uno de los vértices en K y enseguida generamos nvs vértices por cada uno de los vértices en K y los unimos de manera aleatoria.

Los vecinos del vértice 1 son $\{2, 3, 6, 10\}$ y se crean los vértices 11, 12 y 13.

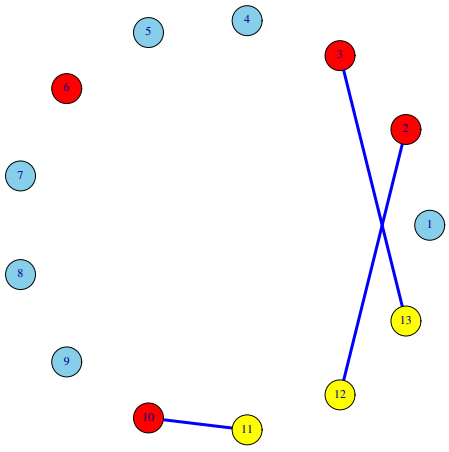


Figura 3.2: Expansión del vértice 1

Los vértices en color rojo son los vecinos del vértice 1 en la gráfica base G_b, mientras que los

vértices en color amarillo son los vértices creados por el parámetro *nvs* relacionados con el vértice 1.

Los vecinos del vértice 4 son $\{3, 5\}$ y se crean los vértices 14, 15 y 16.

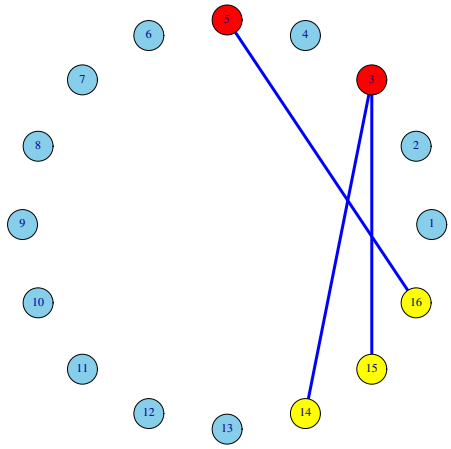


Figura 3.3: Expansión del vértice 4

Los vértices en color rojo son los vecinos del vértice 4 en la gráfica base G_b, mientras que los vértices en color amarillo son los vértices creados por el parámetro *nvs* relacionados con el vértice 4.

Los vecinos del vértice 10 son $\{1, 3, 5, 7, 9\}$ y se crean los vértices 17, 18 y 19.

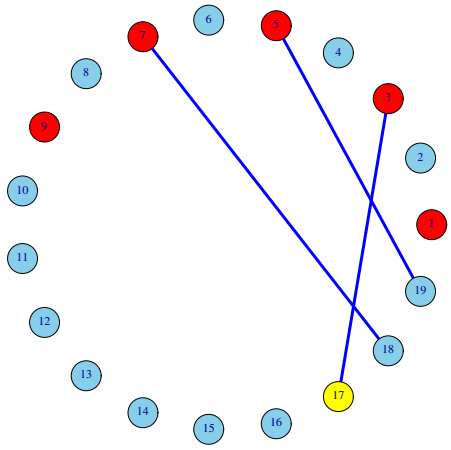


Figura 3.4: Expansión del vértice 10

Los vértices en color rojo son los vecinos del vértice 10 en la gráfica base G_b, mientras que los vértices en color amarillo son los vértices creados por el parámetro *nvs* relacionados con el vértice 10.

La subgráfica resultante Sg de G_b debe incluir a los vértices vecinos de cada uno de los vértices en K, es decir, $V(Sg) = Nvs = (Nei(1) = \{2, 3, 6, 10\}) \cup (Nei(4) = \{3, 5\}) \cup (Nei(10) =$

$\{1,3,5,7,9\}) = \{1,2,3,5,6,7,9,10\}$ y las aristas $[x,y] \in A(G_b)$ que se deben de incluir a la subgráfica Sg son tal que $x, y \in Nvs$.

La subgráfica que se obtiene como resultado es la siguiente:

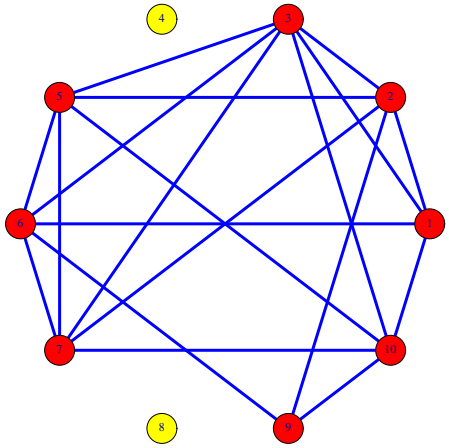


Figura 3.5: Subgráfica Sg de G_b después de la selección del subconjunto de vértices $K = \{1, 4, 10\}$

Los vértices en color rojo corresponden al conjunto Nvs y los vértices en color amarillo son los vértices aislados que se tienen como resultado de la selección del subconjunto K en G_b.

Ahora se procede a construir la gráfica expandida mediante la subgráfica mostrada en la Figura 3.5 y la unión de las expansiones de cada uno de los vértices mostrados en las Figuras 3.2- 3.4.

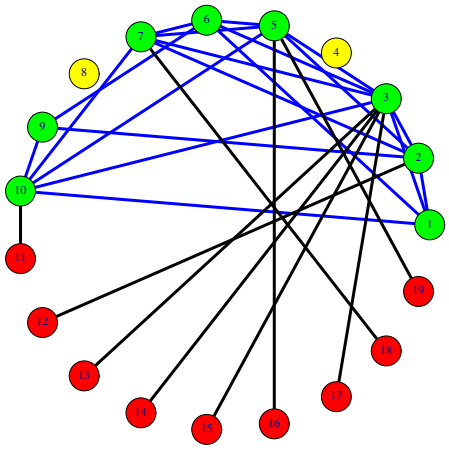


Figura 3.6: Gráfica expandida

Los vértices en color rojo son el resultado del conjunto de vértices seleccionado K y el parámetro nvs, por lo que la cantidad de vértices añadidos son $nvg = \text{card(K)} \times nvs = m \times nvs = 9$. Los vértices en color verde son el resultado de los vecinos de cada uno de los vértices en K, donde card(Nvs) = 8. Las aristas en color azul son las inducidas por la gráfica G_b correspondientes al conjunto de vértices $Nvs \subseteq V(G_b)$. Las aristas en color negro representan las aristas generadas

aleatoriamente entre cada conjunto de vértices vecinos y los nvs vértices creados por cada vértice en K. Los vértices en color amarillo son los vértices aislados, los cuales serán eliminados. Con la eliminación antes descrita se obtendrá una nueva ordenación de los identificadores de los vértices y una gráfica G_0 sin vértices aislados.

Después se crean aristas de manera aleatoria en cada una de las expansiones antes mencionadas en las Figuras 3.2- 3.4, con la condición de que exista una arista que una al correspondiente par de vértices en G_b:

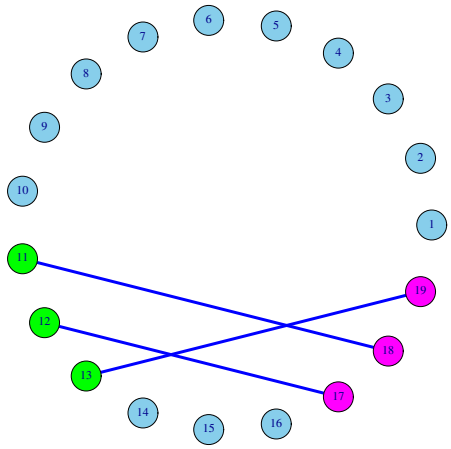


Figura 3.7: Aristas entre la expansión del vértice 1 y 10

Note que las posibles combinaciones de aristas para los vértices en K son: $\{(1,4)(1,10),(4,10\}$, donde sólo la arista $(1,10)$ existe en G_b, por lo que sólo se pueden añadir aristas de los vértices en color verde 11, 12 y 13 a los vértices en color rosa 17, 18 y 19. Las aristas añadidas aleatoriamente fueron: $(11,17)$, $(12,19)$ y $(13,18)$.

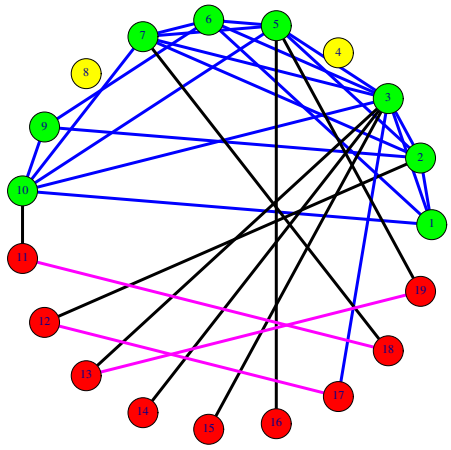


Figura 3.8: Gráfica G_0 sin eliminar a los vértices aislados

Las aristas en la Figura 3.7 son agregadas a la gráfica expandida mostrada en la Figura 3.6, como resultado se obtiene a la gráfica de la Figura 3.8.

Por último, se tiene la gráfica G_0 que corresponde a la gráfica sin vértices aislados de la Figura 3.8 y que es homomorfa a la gráfica G_b.

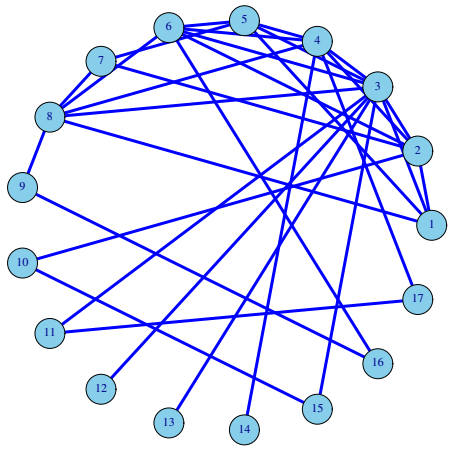


Figura 3.9: Gráfica G_0 sin vértices aislados y con los identificadores de vértices ordenados

El homomorfismo de gráficas $\alpha : V(G_0) \to V(G_b)$ del conjunto de vértices de la gráfica G_0 al conjunto de vértices de la gráfica G_b construido por el algoritmo Expansión de vértices es el siguiente:

$u \in V(G_0)$	$\alpha(u) \in V(G_b)$
1	1
2	2
3	3
4	5
5	6
6	7
7	9
8	10
9	1
10	1
11	1
12	4
13	4
14	4
15	10
16	10
17	10

Tabla 3.1: Homomorfismo de gráficas $\alpha : V(G_0) \to V(G_b)$

En la columna uno se tiene el conjunto de vértices de la gráfica G_0 y en la columna dos se tiene la

imagen $\alpha(u) \in V(G_b)$ correspondiente a cada uno de los vértices $u \in V(G_0)$ bajo el homomorfismo α. Cabe destacar que el vértice 8 de la gráfica G_b no forman parte del conjunto imagen de α.

3.3.3. Contracción de vértices

El funcionamiento del algoritmo que contrae vértices se basa principalmente en la agrupación de la vecindad cerrada del vértice u (incluyendo a u), donde u es seleccionado arbitrariamente de la gráfica G_b. A esta agrupación la representación por medio de un solo vértice llamado *supernodo*. La gráfica G_1 contiene al vértice supernodo agregando al resto de los vértices no adyacentes a u y sus correspondientes aristas de la gráfica base G_b, lo anterior permite establecer un homomorfismo de gráficas $f : V(G_b) \to V(G_1)$ tal que los vecinos de u (incluyendo a u) de G_b se relaciona únicamente con el supernodo en $G1$ y el resto de los vértices $V_r = V(G_b) - (Nei(u) \cup \{u\})$ se relación con la función identidad de V_r, es decir, dado un vértice $x \in V_r$: $id_{V_r}(x) = x \in V(G_1)$.

Algoritmo: *Contracción de vértices*:

- Entrada: Gráfica $G_b(V_b, A_b)$.
- Salida: Una gráfica $G_1(V_1, A_1)$ y un homomorfismo de gráficas $\phi : V_b \to V_1$.

1. Seleccionar un vértice $u \in V_b$.
2. Obtener la vecindad cerrada del vértice $u \in V_b$ y agregarlo a la gráfica G_1 como un solo vértice (supernodo).
3. Agregar el resto de los vértices que no son adyacentes a u y sus correspondientes aristas de la gráfica G_b a G_1.
4. Agregar las aristas de la gráfica G_b que incidan sobre un vértice en la vecindad de u, cambiando el identificador del vértice en la vecindad por el identificador del vértice (supernodo).

A continuación se mencionarán las consideraciones que se tomaron en cuenta para la implementación del algoritmo descrito anteriormente:

1. Siempre se selecciona un vértice con un grado que se encuentre en el promedio de todos los vértices en la gráfica G_b.
2. Al finalizar el algoritmo se hace un ordenamiento de los identificadores de los vértices pertenecientes a la gráfica de salida.

3.3.4. Funcionamiento del algoritmo: Contracción de vértices

A continuación se mostrará un ejemplo del funcionamiento del algoritmo antes descrito:

La gráfica base G_b que se tomó como entrada es la mostrada en la Figura 3.1. El vértice seleccionado aleatoriamente es $u = 4$, los vecinos del vértice u son $Nei(u) = \{3, 5\}$ y los vértices de la vecindad cerrada de u es $N_{G_b}(u) = Nei(u) \cup \{u\} = \{3, 4, 5\}$, la cual representamos por

un supernodo Sn, donde el identificador del vértice Sn es igual al identificador del vértice u, es decir $Sn = 4$. El resto de los vértices que no son adyacentes a u son $N_r = V(G_b) - N_{G_b}(u) = \{1,2,3,4,5,6,7,8,9,10\} - \{3,4,5\} = \{1,2,6,7,8,9,10\}$. Los vecinos del vértice 3 en la gráfica base son $\{1,2,4,5,6,7,8,10\}$ y los vecinos del vértice 5 son $\{2,3,4,6,7,8,10\}$, por lo que todos los vértices, a excepción del 9, tendrán una arista con el vértice $Sn = 4$.

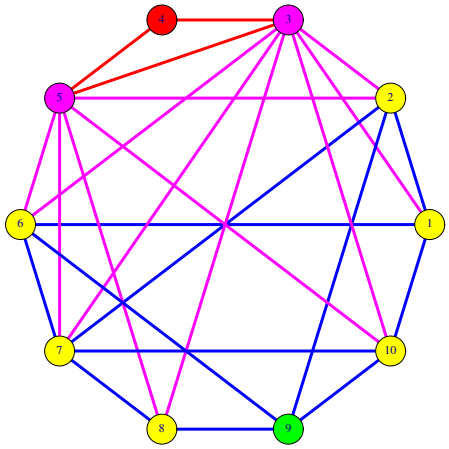


Figura 3.10: Gráfica G_b y la selección del vértice 4 para la contracción

De la Figura3.10 el vértice en color rojo es u, sus dos vecinos 3 y 5 se encuentran coloreados de rosa, los vértices que pertenecen al conjunto N_r están de color amarillo a excepción del vértice 9 que se encuentra de color verde, ya que este no tendrá una arista con el supernodo en la gráfica G_1. Las aristas en color azul corresponden a las aristas de los vértices que no son adyacentes con el vértice u en la gráfica G_b, estas aristas se incluirán en la gráfica G_1. Las aristas en color rosa son de la forma $[x,y] \in A(G_b)$ tal que $x \in Nei(u)$ y $y \in N_r$, estas aristas se añadirán a la gráfica G_1 con el cambio de los vértices $x \in Nei(u)$ por el supernodo $Sn = 4$.

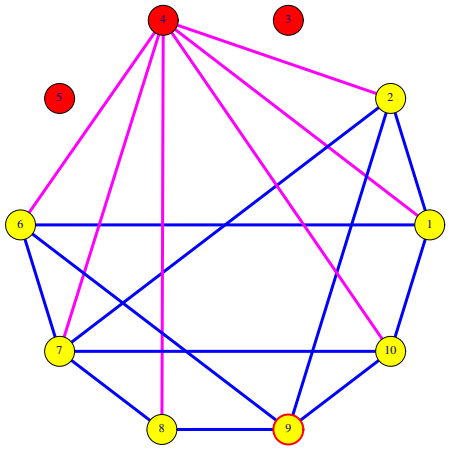


Figura 3.11: Contracción de la gráfica base G_b en el vértice 4

Los vértices aislados que se tienen como resultado de la contracción son los vecinos del vértice

$u = 4$, las aristas de la gráfica inducida por la vecindad del vértice u se transformarán en un bucle en la gráfica G_1 (aristas de color rojo en la Figura 3.10) bajo el homomorfismo $\beta : V(G_b) \to V(G_1)$.

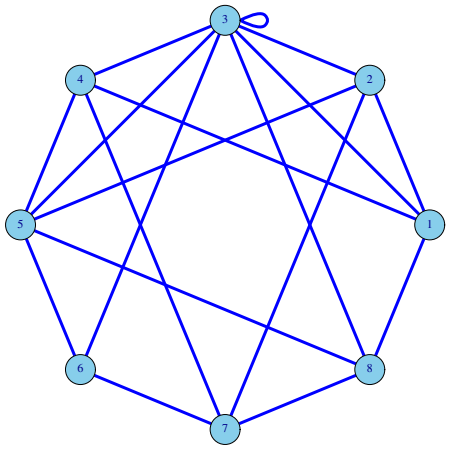


Figura 3.12: Gráfica G_1 con bucle en el supernodo

Por último, se tiene la gráfica G_1 que corresponde a la gráfica sin vértices aislados de la Figura 3.11 y que es homomorfa a la gráfica G_b.

El homomorfismo de gráficas $\beta : V(G_b) \to V(G_1)$ del conjunto de vértices de la gráfica G_b al conjunto de vértices de la gráfica G_1 construido por el algoritmo Contracción de vértices es el siguiente:

$u \in V(G_b)$	$\beta(u) \in V(G_1)$
1	1
2	2
3	3
4	3
5	3
6	4
7	5
8	6
9	7
10	8

Tabla 3.2: Homomorfismo de gráficas $\beta : (G_b) \to V(G_1)$

Dadas dos gráficas cualesquiera G y H, si la gráfica H contiene un bucle en el vértice x, entonces la función $\rho : V(G) \to V(H)$ que asocia todos los vértices de G con el vértice $x \in H$ es un homomorfismo de gráficas. En consecuencia, el problema de homomorfismo de gráficas es resuelto trivialmente si H contiene bucles.

El protocolo propuesto está basado en la dificultad de resolver el problema de homomorfismo de gráficas, por lo anterior consideramos en utilizar gráficas simples, es decir, gráficas que no incluyen

bucles. En el siguiente capítulo se mencionarán los cambios realizados al algoritmo Contracción de vértices para evitar que la gráfica G_1 contenga bucles.

De esta manera, se obtendría la clave pública (G_0, G_1) de la Figura 3.9 y Figura 3.12 respectivamente. Mientras que la clave privada correspondería a la composición de homomorfismo de α y β mostrados en la Tabla 3.1 y Tabla 3.2 respectivamente.

$u \in V(G_0)$	$\beta(\alpha(u)) \in V(G_1)$
1	1
2	2
3	3
4	3
5	4
6	5
7	7
8	8
9	1
10	1
11	1
12	3
13	3
14	3
15	8
16	8
17	8

Tabla 3.3: Homomorfismo de gráficas $\phi = \beta \circ \alpha$

La clave privada es 12334578111333888 para este ejemplo.

3.4. Tipos de representación para gráficas

Existen diversas maneras de representar la estructura de una gráfica, tales como: lista de aristas, lista de adyacencia y matriz de adyacencia. Los algoritmos que se aplican sobre gráficas adoptan tiempos de ejecución diferentes dependiendo de la forma de representación elegida. En particular, los tiempos de ejecución variarán en función del número de vértices y de aristas que contenga la gráfica, por lo que la utilización de una representación u otra dependerá en gran medida si la gráfica es densa (el número de aristas es considerable) o disperso (el número de aristas es mínimo). Sea $G = (V, E)$ una gráfica con $n = |V|$, $a = |E|$. Definiremos la densidad de la gráfica como:

$$d = \frac{2a}{n(n-1)}$$

Note que la expresión analítica de la probabilidad en el modelo *Erdos-Rényi (n,p)* es exactamente igual a la densidad de la gráfica:

$$p = \frac{2a}{n(n-1)} = d$$

Por lo tanto $0 \leq d \leq 1$, tal que si $d = 0$, entonces no existe ninguna arista entre los vértices de la gráfica, en otro caso, si $d = 1$, entonces la gráfica es completa K_n. Si d es cercano a 0, se dice que la gráfica es dispersa, y si d es cercano a 1, se dice que la gráfica es densa.

A continuación se muestra una gráfica de prueba G_p para calcular su densidad:

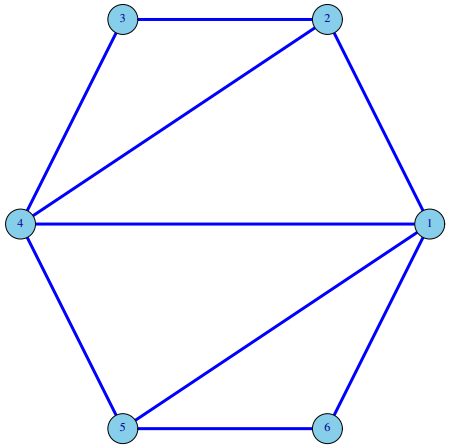


Figura 3.13: Gráfica de prueba G_p con 6 vértices y 9 aristas

$$d(G_p) = \frac{2a}{n(n-1)} = \frac{2\times 9}{6\times(6-1)} = \frac{18}{30} = 0.6$$

Tendrían que añadirse 6 aristas más para que la gráfica mostrada en la Figura 3.13 fuese una gráfica completa K_6. De esta manera, si calculamos la densidad de una gráfica completa K_6 obtendríamos su valor máximo:

$$d(K_6) = \frac{2a}{n(n-1)} = \frac{2\times 15}{6\times(6-1)} = \frac{30}{30} = 1$$

3.4.1. Lista de aristas

Esta representación corresponde al modelo de datos de la biblioteca `igraph`, donde los vértices están etiquetados por un número entre 0 y el número de vértices menos uno. Las aristas son pares de vértices ordenados y la gráfica es un conjunto de aristas, vértices y algunos meta datos que sirven para identificar algunas de las características de la gráfica.

A continuación, se muestra un ejemplo de cómo se hace la representación de la gráfica mostrada en la Figura 3.13 con el modelo de datos de la biblioteca `igraph`:

(Vértices: 6, Gráfica dirigida: no, {(1,2), (1,4), (1,5), (1,6), (2,3), (2,4), (3,4), (4,5), (5,6)}).

En este caso, el espacio de almacenamiento de la gráfica es $2 \times a$.

3.4.2. Lista de adyacencia

Una lista de adyacencia consiste de una lista de los vértices de la gráfica y para cada vértices una lista de sus vértices vecinos. Es decir, un vértice i tendrá una lista asociada en la que aparecerá un elemento con una referencia al vértice j si existe una arista (i, j) que los une. Para evitar repeticiones de vértices en las listas, se puede ordenar la lista de los vértices vecinos y aceptar los vértices de la lista que sean mayores al vértice seleccionado.

A continuación, se muestra un ejemplo de cómo se hace la representación de la gráfica mostrada en la Figura 3.13 con lista de adyacencia:

(Vértices: 6, Gráfica dirigida: no, {(1,{2,4,5,6}), (2,{1,3,4}), (3,{2,4}), (4,{1,2,3,5}), (5,{1,4,6}), (6,{1,5})}).

Aplicando el criterio para no repetir vértices:

(Vértices: 6, Gráfica dirigida: no, {(1,{2,4,5,6}), (2,{3,4}), (3,{4}), (4,{5}), (5,{6}), (6,{ })}).

En este caso, el espacio de almacenamiento de la gráfica es $n + a$.

3.4.3. Matriz de adyacencia

La matriz de adyacencia es la forma más común de representación por ser la más directa. Consiste de una matriz $n \times n$, donde el elemento a_{ij} tendrá como valor 1 si existe una arista (i, j) que los una. En caso contrario, el valor será 0.

A continuación, se muestra un ejemplo de cómo se hace la representación de la gráfica mostrada en la Figura 3.13 con la matriz de adyacencia:

(Vértices: 6, Gráfica dirigida: no).

i/j	1	2	3	4	5	6
1	0	1	0	1	1	1
2	1	0	1	1	0	0
3	0	1	0	1	0	0
4	1	1	1	0	1	0
5	1	0	0	1	0	1
6	1	0	0	0	1	0

Tabla 3.4: Matriz de adyacencia de la gráfica de prueba G_p

En este caso, el espacio de almacenamiento de la gráfica es $n \times n$.

3.5. Codificación del par de claves

Para la representación de la clave pública se puede emplear la matriz de adyacencia asociada a cada una de las gráficas generadas en el protocolo propuesto. Dado que estamos usando gráficas no

dirigidas, sin bucles y sin múltiples aristas, la matriz de adyacencia asociada a nuestras gráficas es simétrica y solo contiene $0's$ y $1's$ como posibles valores. Por lo anterior, sólo basta con extraer la información que está por encima de la diagonal para poder reconstruir a la matriz por completo.

La información que está por encima de la diagonal de la matriz simétrica se puede almacenar en un vector que recorra a la matriz de manera horizontal, vertical o en diagonal según convenga el caso. El vector resultante se puede codificar mediante un código de repetición para compactar las claves generadas por el protocolo, donde una secuencia de $0's$ y $1's$ de longitud mayor o igual a tres es cambiada por la longitud de la secuencia seguida del símbolo $0's$ o $1's$ según sea el caso.

Por ejemplo, de la matriz de adyacencia mostrada en la Tabla 3.4 recorriendo la matriz en diagonal se tiene el siguiente vector:

$$111110100100101$$

La codificación con la técnica antes descrita es la siguiente:

$$510100100101$$

El código anterior se puede interpretar de la siguiente manera: 5 (unos) seguido de 1 (cero) ... seguido de 1 (uno).

Un método mejor para la codificación de la clave pública se puede conseguir mediante el uso de un arreglo A de 64 caracteres ASCII imprimibles, después se toman bloques de 6 elementos del vector resultante y por último se asigna a cada bloque el carácter de A correspondiente. A continuación se muestra un ejemplo de la matriz de adyacencia mostrada en la Figura 3.4.

Donde:

$A = \{0-9\} \cup \{a-z\} \cup \{A-Z\} \cup \{@, \$\}.$

$Bloque_1 = 111110_2 = 62_{10}$

$Bloque_2 = 100100_2 = 36_{10}$

$Bloque_3 = 101000_2 = 40_{10}$

Dado que el número de elementos del vector resultante no es divisible por 6, se añaden $0's$ a la derecha del último bloque hasta que contenga 6 elementos. La nueva codificación es la siguiente:

$$A[62] - A[36] - A[40] = @AE$$

En las pruebas realizadas, la codificación del segundo método siempre fue mejor que la codificación basada en repetición. Por otro lado, el número en bytes de la codificación del segundo método se puede calcular conociendo el número de vértices n de la siguiente manera:

$$Longitud = \lceil \frac{n(n-1)}{2 \times b} \rceil$$

(donde b es la longitud del bloque en bits)

Para nuestro ejemplo, dado que la gráfica tiene $n = 6$ y la longitud de los bloques son de 6 bits, la longitud es de $\lceil \frac{6\times(6-1)}{2\times6} \rceil = \lceil \frac{5}{2} \rceil = 3$ bytes.

El resultado de la longitud de la gráfica codificada mediante la segunda técnica depende únicamente del valor del número de vértices de la gráfica, mientras que la codificación por repetición depende de la distribución de las aristas en la gráfica.

La codificación del homomorfismo $\phi : V(G_0) \rightarrow V(G_1)$ se hace de manera similar al de la clave pública, ya que solo es una secuencia de números. En la implementación actual, el número de vértices de G_1 se encuentra restringido a 64 debido a que solo se emplean hasta 6 bits por bloque. Una posible solución a este problema es comentada en el capítulo Conclusiones y perspectivas. Cabe destacar que esto sólo afecta a la codificación de la clave privada.

3.6. Biblioteca `igraph`

`Igraph` es un conjunto de herramientas de creación y manipulación de gráficas, con el énfasis en la eficiencia, portabilidad y facilidad de uso. `Igraph` es de código abierto y libre que se puede programar en `GNU R`, `Python` y `C/C++`.

En primer lugar `Igraph` contiene la implementación de algoritmos clásicos de la teoría de gráficas como: isomorfismo de gráficas, conectividad, transitividad y recorridos en gráficas, detección de comunidades, etc.

En segundo lugar, `igraph` proporciona una plataforma para los algoritmos de gráficas en desarrollo y/o aplicación. Tiene una estructura de datos muy eficiente para la creación, representación y manipulación de gráficas. También cuenta con otras estructuras de datos muy flexibles y fáciles de usar, tales como: vectores, matrices, pilas, colas, listas, etc. Para mayor información puede visitar la página principal de la librería `Igraph` [15].

En la siguiente página, de la Tabla 3.5:

- En la función `igraph_neighbors`, d es el número de vértices vecinos al vértice dado.
- En la función `igraph_get_eid`, de la pareja de vértices dada $(v1, v2)$, d es el valor mínimo del grado de salida del vértice $v1$ y del grado de entrada del vértice $v2$, $d1$ es el valor mínimo del grado de salida del vértice $v2$ y del grado de entrada del vértice $v1$.
- En la función `igraph_add_edges`, $|V|$ y $|E|$ corresponden a la nueva gráfica.
- En la función `igraph_delete_vertices`, $|V|$ y $|E|$ corresponden a la gráfica original.
- En la función `igraph_vector_init`, n es el número de elementos del vector.
- En la función `igraph_vector_binsearch`, n es el número de elementos del vector. El vector debe de estar ordenado previamente a la aplicación de esta función.
- En la función `igraph_vector_contains`, n es el número de elementos del vector. La búsqueda del elemento en el vector es lineal.

A continuación, se mencionan las funciones de la librería `Igraph` que se utilizaron para desarrollar el protocolo propuesto:

Nombre de la función	Operación	Complejidad
`igraph_neighbors`	Proporciona los vecinos de un vértice dado	$O(d)$
`igraph_get_eid`	Proporciona el identificador de una arista	$O(\log(d) + \log(d1))$
`igraph_get_edgelist`	Calcula la lista de aristas de una gráfica	$O(\|E\|)$
`igraph_add_edges`	Agrega aristas a una gráfica	$O(\|V\| + \|E\|)$
`igraph_delete_vertices`	Borra vértices de una gráfica (con todas sus aristas)	$O(\|V\| + \|E\|)$
`igraph_vcount`	Calcula el numero de vértices en una gráfica	$O(1)$
`igraph_ecount`	Calcula el número de aristas de una gráfica	$O(1)$
`igraph_write_graph_edgelist`	Escribe la lista de aristas de una gráfica en un archivo	$O(\|E\|)$
`igraph_erdos_renyi_game`	Genera una gráfica aleatoria *Erdös-Rényi(n,p)*	$O(\|V\| + \|E\|)$
`igraph_empty`	Genera una gráfica vacía	$O(\|V\|)$
`igraph_create`	Genera una gráfica con las aristas especificadas	$O(\|V\| + \|E\|)$
`igraph_vector_init`	Construye un vector	$O(n)$
`igraph_vector_size`	Devuelve la longitud del vector	$O(1)$
`igraph_vector_binsearch`	Encuentra un elemento por búsqueda binaria	$O(\log(n))$
`igraph_vector_contains`	Verifica si un elemento esta incluido en el vector	$O(n)$
`igraph_vector_destroy`	Libera la memoria asignada a un vector	$O(n)$
`igraph_destroy`	Libera la memoria asignada a una gráfica	$O(\|V\| + \|E\|)$
`igraph_copy`	Crea la copia de una gráfica	$O(\|V\| + \|E\|)$
`igraph_simplify`	Elimina ciclos y múltiples aristas de una gráfica	$O(\|V\| + \|E\|)$
`igraph_contract_vertices`	Reemplaza múltiples vértices por uno solo	$O(\|V\| + \|E\|)$
`igraph_isomorphic`	Decide si dos gráficas son isomorfas	exponencial
`igraph_isomorphic_vf2`	Decide si dos gráficas son isomorfas y permite conocer el homomorfismo en ambos sentidos	exponencial
`igraph_isomorphic_bliss`	Decide si dos gráficas son isomorfas y ademas de conocer el homomorfismo en ambos sentidos, permite manejar coloración de vértices y aristas	exponencial
`igraph_subisomorphic_vf2`	Decide el problema de isomorfismo de subgráficas	exponencial

Tabla 3.5: Complejidad temporal de las funciones utilizadas de la librería `Igraph`

Capítulo 4

Protocolo de autenticación

En este capítulo se describen los parámetros de control del protocolo de autenticación propuesto, la manera de estimar la cantidad de vértices de cada una de las gráficas involucradas en el protocolo a partir de la construcción de la gráfica base G_b, esto con el fin de conocer la longitud en bytes de la clave generada. Se muestra un ejemplo ilustrativo de todas las fases de la generación de claves del protocolo propuesto, donde se explica con detalle la obtención de los homomorfismos correspondientes.

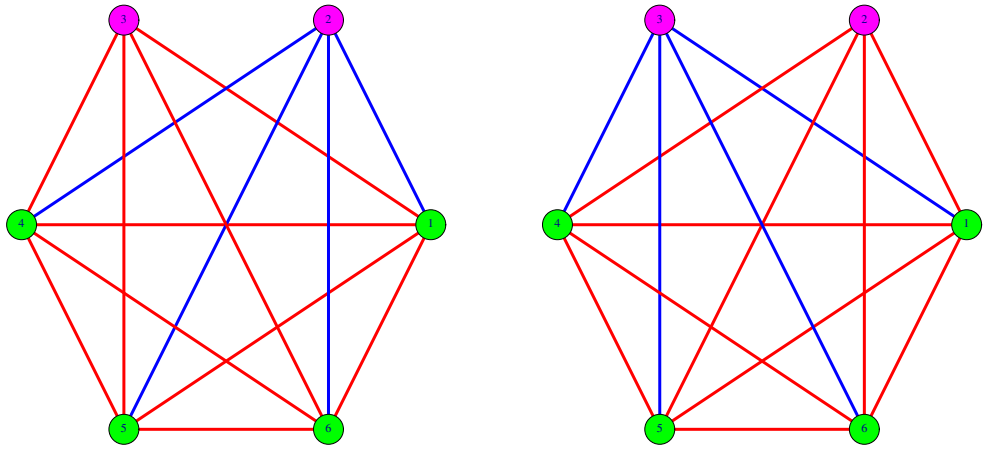
4.1. Elección de la gráfica base

Sea una gráfica completa de n vértices $K_n = (V_k, A_k)$, de la cual son seleccionados dos vértices al azar, digamos $u, v \in V_k$. Llamemos G_b a la gráfica que resulta de K_n eliminando la arista $[u, v] \in A_k$, entonces el conjunto de vértices de G_b está definido por V_k y su conjunto de aristas por $A_k - \{[u, v]\}$.

Se puede observar que la subgráfica inducida de G_b obtenida por la vecindad cerrada del vértice u es una gráfica completa K_{n-1}, lo mismo sucede si tomamos la vecindad cerrada del vértice v, más aún, es la subgráfica completa más grande contenida en la gráfica G_b, es decir, es un clan máximo. El orden de un clan máximo de una gráfica G es denotado por $\omega(G)$. Por lo anterior, se tiene que la gráfica G_b es homomorfa a la gráfica completa de $n-1$ vértices K_{n-1}, que a su vez es la gráfica más pequeña (hablando de gráficas simples) que permite un homomorfismo de gráficas $f : V_k \to V(K_{n-1})$ de la gráfica G_b a K_{n-1}.

Consideremos cualquier gráfica base G_b a la cual sólo le hace falta una arista para ser una gráfica completa de n vértices, se puede observar que su número cromático y el orden de su clan máximo coinciden, es decir, $\chi(G_b) = \omega(G_b) = n - 1$.

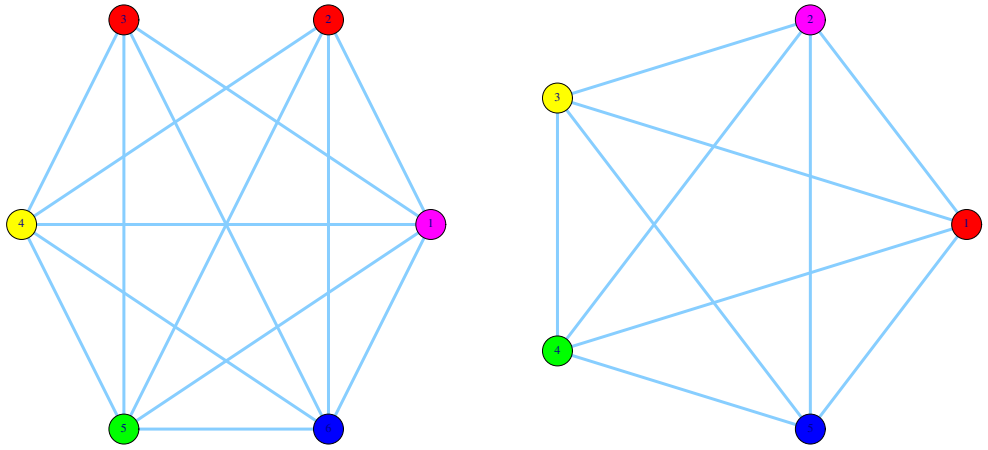
A continuación se muestra un ejemplo de una gráfica base G_b con un número de vértices $n = 6$, mostrando los dos clanes máximos que se obtienen de elegir la vecindad cerrada de los únicos dos vértices que no son adyacentes en G_b.



(a) Clan máximo con $V = \{1, 3, 4, 5, 6\}$ (b) Clan máximo con $V = \{1, 2, 4, 5, 6\}$

Figura 4.1: Gráfica base G_b con 6 vértices

De la Fgura 4.1 los vértices en color rosa son los únicos dos vértices que no son adyacentes en la gráfica G_b. Los vértices en color verde son aquellos que tienen grado $n-1$, las aristas en color rojo pertenecen al clan máximo de G_b para ambos casos. A continuación se muestra una coloración óptima de la gráfica G_b y la gráfica completa K_{n-1}.



(a) Gráfica G_b (b) Gráfica completa K_{n-1}

Figura 4.2: Homomorfismo de la gráfica G_b a K_{n-1}

De la Figura 4.2 se puede observar que existe un homomorfismo de gráficas que relaciona a los vértices de la gráfica G_b con los vértices del mismo color en la gráfica completa K_5. Las gráficas base de n vértices construidas de esta manera, tienen la característica de que $\chi(G) = \omega(G) = n-1$.

Una *gráfica perfecta* G es tal que el número cromático de toda subgráfica inducida H de G es igual al tamaño del clan máximo de esa subgráfica, es decir, $\chi(H) = \omega(H)$. Si G es una gráfica perfecta, entonces el problema de coloración de gráficas puede ser resuelto por un algoritmo de

complejidad temporal polinomial. Por lo anterior, se debe de evitar que la gráfica G_0 construida a partir de la gráfica base G_b no sea una gráfica perfecta.

Un ciclo inducido C_{2n+1} con un número de vértices $n \geq 2$ es llamado *agujero impar* y su complemento $\overline{C_{2n+1}}$ es llamado un *anti-agujero*. Una gráfica que no contiene agujeros impares ni sus complementos es llamada una gráfica de Berge. Por el teorema fuerte de las gráficas perfectas se tiene que una gráfica G es perfecta si y sólo si es una gráfica de Berge. Llamemos *gráfica imperfecta* a una gráfica G que no es perfecta. Por último, una gráfica G necesariamente debe de tener un agujero impar inducido para ser una gráfica imperfecta.

4.2. Descripción de los parámetros de control

Los parámetros de control nos permiten construir gráficas con un número de vértices y aristas equiparables. La expansión de vértices permite crear gráficas con un número de vértices y aristas variable sin importar el orden y tamaño de la gráfica base. Por otro lado, el número de vértices de G_1 generados por la contracción, se encuentran limitadas por el orden de la gráfica base G_b, es decir, $V(G_1) \leq V(G_b)$.

A continuación se muestran los parámetros de control para la construcción de las gráficas involucradas en el protocolo, así como sus correspondientes homomorfismos:

- Generación de G_b: $(v_b, k) \mapsto G_bGen(v_b, k) = (G_b(\text{card}(V(G_b), \text{card}(A(G_b)))$
- Generación de claves: $G_b \mapsto KeyGen(G_b) = ((G_0, G_1), \pi)$
  - $T_i : G_b \mapsto T_i(G_b) = (G_i, \pi_i)$, con $i = [0, 1]$
  - Clave pública: (G_0, G_1)
  - $G_i(v_i, a_i) = (\text{card}(V(G_i)), \text{card}(A(G_i)))$
  - Clave privada: homomorfismo $\pi : V(G_0) \to V(G_1)$
- Compromiso: $T_0 : G_0 \mapsto T_0(G_0) = (H_p, \psi)$
  - Gráfica compromiso: H_p
  - $H_p(v, a) = (\text{card}(V(H_p)), \text{card}(A(H_p)))$

Donde:

- La transformación T_0 es el algoritmo expansión de vértices.
- La transformación T_1 es el algoritmo contracción de vértices.
- El homomorfismo $\pi_0 : V(G_0) \to V(G_b)$.
- El homomorfismo $\pi_1 : V(G_b) \to V(G_1)$.
- El homomorfismo $\pi = \pi_1 \circ \pi_0$
- El homomorfismo $\psi : V(H_p) \to V(G_0)$.

Para la construcción de la gráfica base $G_b(V, A)$:

- v_b: Número inicial de vértices de la gráfica base G_b.

El modelo Erdös-Rényi (n, p) presenta secuencias de grados de vértices muy variadas en la gráfica base G_b, aún cuando los parámetros de su construcción permanecen constantes. Esto conlleva a que las gráficas G_0 y G_1 construidas a partir de la gráfica G_b, no se sitúen en un rango específico de valores en relación a su número de vértices y aristas. Lo anterior no es deseable ya que es imposible tener un control sobre el orden y tamaño de nuestras gráficas. En cambio, una gráfica base tomada de una gráfica completa eliminando una arista de ésta, nos permite crear gráficas G_0 y G_1 de tal manera que se puede construir un homomorfismos de gráficas entre éstas en base a la coloración óptima de gráficas de manera muy sencilla. También es fácil determinar el crecimiento del número de vértices y aristas de las gráficas G_0 y G_1 construidas a partir de la gráfica base G_b

Para la construcción de la gráfica G_0:

- G: Gráfica de entrada, en este caso se usa la gráfica base G_b.
- nvs: Número de selección inicial, número de vértices diferentes seleccionados aleatoriamente de la gráfica de entrada.
- exp: Coeficiente de expansión, número de vértices a generar por cada vértice seleccionado.
- $intentos$: Coeficiente de aristas.

La construcción de la gráfica compromiso H_p es similar a la construcción de la gráfica G_0, con la diferencia de que se toma G_0 como la gráfica de entrada.

Para la construcción de la gráfica G_1:

- G: Gráfica de entrada, en este caso se usa la gráfica base G_b.

Donde:

- $nvs \in [3, v_b]$
- $exp \in \mathbb{Z}^+$
- $intentos \in \mathbb{Z}^+$
- $\text{card}(V(G_b)) = v_b$
- $\text{card}(A(G_b)) = \frac{v_b \times (v_b - 1)}{2} - 1$

4.3. Generación del homomorfismo en la expansión de vértices

Sea G_b una gráfica que resulta de una gráfica completa con n vértices $K_n = (V_k, A_k)$ eliminando la arista $[u, v] \in A_K$ con $u, v \in V_K$. Si seleccionamos dos vértices diferentes al azar en G_b, digamos $x, y \in V(G_b)$, existe una probabilidad de $p = 1 - \frac{2}{n\times(n-1)}$ de que la unión de los vecinos N de x y y en G_b sea el conjunto de vértices de G_b, entonces la subgráfica inducida por N de como resultado G_b. Por otro lado si se escogen tres vértices diferentes al azar en G_b, es un hecho que la unión de los vecinos de esos tres vértices es el conjunto $V(G_b)$. Por lo anterior, con $nvs \geq 3$ se garantiza que el número cromático de G_0 es $v_b - 1$, donde v_b es el número de vértices de la gráfica G_b.

El homomorfismo $\pi_0 : V(G_0) \to V(G_b)$ se obtiene de la siguiente manera:

Recordemos que se hace una selección de un subconjunto de vértices diferentes $K \subseteq V(G_b)$, con $\text{card}(K) = nvs$. Después se obtienen los vecinos de cada vértice $u \in K$ y se genera el conjunto $N = \bigcup_{i=1}^{i=nvs} Nei(u_i)$. La subgráfica inducida por N obtenida de la gráfica G_b la llamamos Sb_0, donde $V(Sb_0) = N \subseteq V(G_0)$, existe un homomorfismo $hom_0 : V(Sb_0) \to V(G_b)$, con $hom_0(v) = v$ para cualquier vértice $v \in V(Sb_0)$.

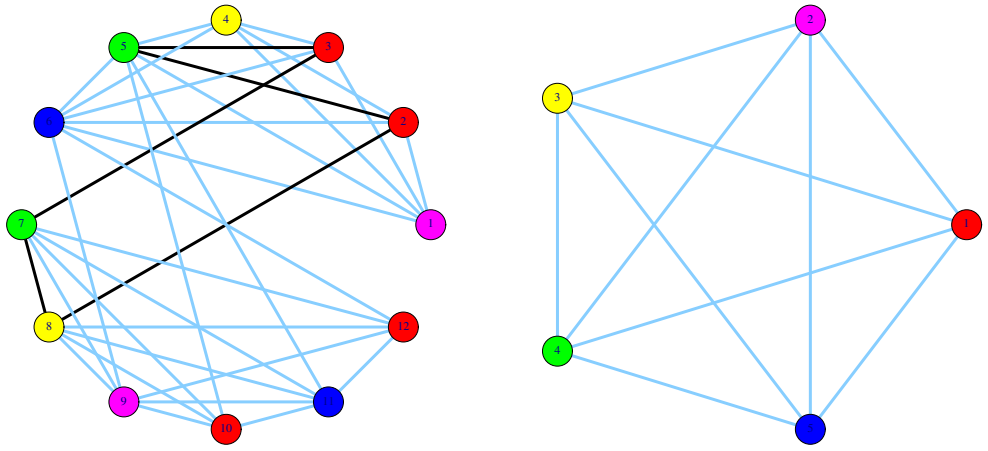
La gráfica generada Sb_1 por el coeficiente de expansión, tiene exp vértices por cada vértice $u \in nvs$. Sea X_i cada uno de los conjuntos de vértices generados con $\text{card}(X_i) = exp$, donde $V(Sb_i) = \bigcup_{i=1}^{i=nvs} X_i$. Entonces se puede construir un homomorfismo $hom_1 : V(Sb_1) \to V(G_b)$, con $hom_1(v) = u_i$ para cada uno de los vértices $v \in X_i$ y $u \in K$.

De esta manera la gráfica $G_0 = (V(Sb_0) \cup V(Sb_1), A(Sb_0) \cup A(Sb_1))$ y el homomorfismo correspondiente entre G_0 y G_b es el siguiente:

$$\pi_0(v) = \begin{cases} hom_0(v), & \text{si } v \in V(Sb_0) \\ hom_1(v), & \text{si } v \in V(Sb_1) \end{cases}$$

Para asegurar que la gráfica G_0 sea una gráfica imperfecta, basta con que G_0 contenga un agujero impar. Un agujero impar se puede construir durante el paso $2b$ del algoritmo Expansión de vértices, si y sólo si existen tres aristas $\{[u, x], [v, y], [w, z]\} \in G_0$ tal que $\{u, v, w\} \in V(Sb_0)$, $\{x, y, z\} \in V(Sb_1)$ y $[u, v] \notin A(G_b)$.

Lo anterior es muy sencillo de implementar, ya que los vértices u y v de G_b siempre se incluyen en K y para las aristas $\{[u, x], [v, y], [w, z]\}$ sólo basta con calcular una permutación de tres vértices valida y unirlos con u, v y w. Recordar que w tiene un grado de $v_b - 1$ y el grado para u y v es de $v_b - 2$. A continuación se muestra un ejemplo ilustrativo que explica lo comentado anteriormente.



(a) Gráfica G_0 (b) Gráfica G_1

Figura 4.3: Homomorfismo de la gráfica G_0 a G_1

Los parámetros usados para construir a la gráfica G_0 a partir de la gráfica base G_b de la Figura 4.1 son los siguientes: $nvs = 6$ y $exp = 1$. Los vértices $\{2, 3, 5, 7, 8\}$ inducen un agujero impar de cinco vértices en la gráfica G_0, por lo que la gráfica G_0 es una gráfica imperfecta.

La subgráfica inducida Sb_0 de G_b para este ejemplo tiene como conjunto de vértices a $\{1, 2, 3, 4, 5, 6\}$ y la gráfica Sb_1 tiene como conjunto de vértices a $\{7, 8, 9, 10, 11, 12\}$. Se puede observar de la Figura 4.3 que las aristas que van del conjunto de vértices de Sb_0 al conjunto de vértices de Sb_1 son $\{[2, 8], [3, 7], [5, 10], [5, 11], [6, 9], [6, 12]$, llamemos a este conjunto de aristas A_{Sb}. Los vértices que no son adyacentes en G_b son $u = 2$ y $v = 3$, por lo que basta generar en A_{Sb} dos aristas incidentes sobre estos vértices y dos vértices diferentes $x = 8$ y $y = 7$ del conjunto $V(Sb_1)$; y una arista más $[c = 5, z = 10] \in G_0$ con $c \neq a$ y $c \neq b$, y $z \neq x$ y $z \neq y$. La existencia de estas aristas permiten generar el agujero impar inducido C_5 por el conjunto de vértices $\{a, b, c, x, y\}$ en G_0.

4.4. Generación del homomorfismo en la contracción de vértices

El homomorfismo $\pi_1 : V(G_b) \to V(G_1)$ se obtiene de la siguiente manera:

La contracción se realiza mediante la selección de un vértice $u \in G_b$, los vértices que conforman su vecindad cerrada $S = \{v \in V(G_b) : [u, v] \in A(G_b)\}$ y el vértice u son contraídos para formar un super vértice llamado Sn, hacemos $Sun = S \cup u$. De esta manera, la gráfica G_1 contiene al vértice Sn y al resto de los vértices $Sv = G_b - Sun$ que no son adyacentes a $u \in V(G_b)$.

Para dos vértices cualesquiera $u, v \in Sv$ si $[u, v] \in A(G_b)$ es una arista en G_b, entonces $[u, v] \in A(G_1)$ es una arista en G_1. Si para dos vértices $u \in Sv$ y $v \in Sun$ si $[u, v] \in A(G_b)$ es una arista en G_b, entonces $[u, Sn] \in A(G_1)$ es una arista en G_1.

Por lo anterior, se puede construir un homomorfismo $hom_2 : V(G_b) \to V(G_1)$ de la siguiente manera:

$$\hom_2(v) = \begin{cases} v, & \text{si } v \in Sv \\ Sn, & \text{si } v \in Sun \end{cases}$$

Para dos vértices cualesquiera $x, y \in Sun$ si $[x, y] \in G_b$, entonces $[x, y] \in Sa \subseteq A(G_b)$. Toda arista $[x, y] \in Sa$, bajo el homomorfismo hom_2 genera colisiones de la forma $hom_2(x) = hom_2(y)$. Como resultado se tiene la construcción de una gráfica G_1 con bucles en el super vértice Sn, es decir, existe una arista donde sus dos extremos es el vértice Sn.

Bajo esta construcción, un homomorfismo trivial $T : G \to G_1$ puede asignar cualquier conjunto de vértices $Vc \subseteq V(G)$ hacia un vértice que contenga un lazo [12, 13, 21]. Por lo anterior, es necesario hacer una serie de ajustes en la construcción de la gráfica G_1 así como en el homomorfismo π_1, los cuales son descritos a continuación:

En particular, si el vértice u seleccionado tiene un grado de $n - 1$, donde n es el número de vértices de G_b, entonces toda la gráfica G_b se representaría mediante un super vértice Sn, en consecuencia todas las aristas de G_b bajo hom_2 crearían bucles en G_1.

El primer cambio en el algoritmo Contracción de vértices es seleccionar un vértice u con grado $n - 1$. Después se procede a crear un vértice z en G_1 con la arista $[z, Sn] \in A(G_1)$. El valor de $hom_2(u) = Sn$ es cambiado por $hom_2(u) = z$. Este cambio en el homomorfismo garantiza que si una arista $[x, y] \in A(G_b)$ con $(x = u) \mid (y = u)$, entonces $h(x)h(y) \in A(G_1)$ es una arista en G_1 con $h(x) \neq h(y)$.

El segundo cambio es crear una gráfica auxiliar $Gaux$ de la gráfica G_b eliminando al vértice u junto con todas las aristas que incidan sobre este.

Por último, mientras $Gaux$ no sea una gráfica vacía, hágase lo siguiente:

1. Creamos un vértice a en G_1 y lo unimos con cada uno de los vértices $v \in V(G_1)$.

2. Buscamos el vértice x con el grado más bajo en $Gaux$, asignamos el valor de $hom_2(x) = a$.

3. Generamos una lista l con todos los vértices que no son adyacentes a x.

4. Generamos una lista m con todos los vértices que son adyacentes a x y creamos una cantidad de vértices b igual a la longitud de la lista m en G_1

5. Para cada vértice b en G_1 se crea una arista con cada uno de los vértices $v \in V(G_1)$.

6. Para todos los vértices $y \in l$, asignamos el valor del homomorfismo $hom_2(y) = a$.

7. Eliminamos al vértice x de la gráfica auxiliar $Gaux$ junto con todas las aristas que incidan sobre este.

8. Para cada uno de los vértices $y \in l$, eliminamos al vértice y de la gráfica auxiliar $Gaux$ junto con todas las aristas que incidan sobre este.

4.5. Estimación del orden de las gráficas involucradas en el protocolo

El número de vértices máximo que se obtiene por la expansión de vértices es $Vmax = v_b + nvs \times exp$. En consecuencia, los parámetros de control que determinan la cantidad de vértices son: v_b, nvs y exp.

Ahora bien, si el valor de nvs está cercano a su valor máximo, es decir $nvs = v_b$, la probabilidad de obtener $Vmax$ es muy alta y el número de vértices tiende a $v_b + (nvs \times exp)$. Como se vio anteriormente, es necesario que el parámetro $nvs \geq 3$ para que G_0 sea una gráfica imperfecta, por lo que el número de vértices esperado de G_0 es igual a:

$$E(|V(G_0)|) = Vmax.$$

Mientras que el número de aristas esperado de la gráfica G_0 es igual a:

$$E(|A(G_0)|) = \text{card}(A(G_b)) + (nvs \times exp) + \frac{(nvs \times exp) \times ((nvs \times exp) - 1)}{2}$$

El número de vértices esperado de la gráfica G_1 es igual a:

$$E(|V(G_1)|) = v_b - 1$$

como ya se había visto anteriormente. La gráfica G_1 siempre es una gráfica completa de $v_b - 1$ vértices, por lo que su número esperado de aristas es de:

$$E(|A(G_1)|) = \frac{(v_b - 1) \times (v_b - 2)}{2}$$

La longitud de la clave(pública, privada) depende únicamente de la cantidad de vértices de la gráfica G_0 y G_1. El número esperado de la longitud de la clave en bytes es:

$$E((G_0, G_1), \pi) = \frac{(E(v_0) \times (E(v_0) - 1)) + (E(v_1) \times (E(v_1) - 1))}{2} + E(v_0).$$

4.6. Ejemplo práctico

A continuación se muestra un ejemplo detallado de cada una de las fases de la generación de claves del protocolo:

El parámetro para la gráfica base G_b es el siguiente:

- $v_b = 5$

El paquete de `igraph` para `R` tiene un error con los identificadores de los vértices mostrados en sus gráficas, ya que los identificadores de los vértices en las estructuras de datos para la manipulación de las gráficas inician en 0, cuando se usa la función `tkplot` se generan gráficas donde el identificador del vértice inicia en 1. Observe que la gráfica mostrada en la Figura 4.4, el primer identificador es el 1.

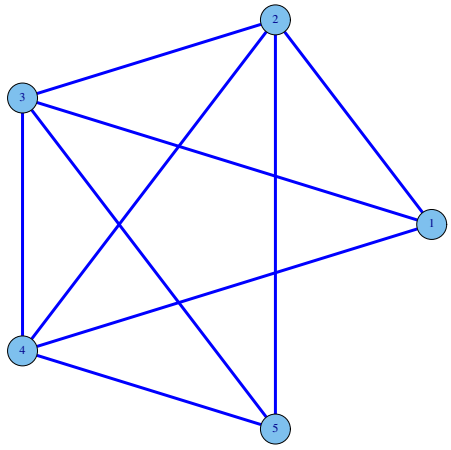


Figura 4.4: Gráfica G_b con $v_b = 5$

Los parámetros para la gráfica G_0 son los siguientes:

- $G = G_b$
- $nvs = 3$
- $exp = 1$
- $intentos = 1$

Los vértices seleccionados $nvs = \{2, 5, 4\} \in V(G_b)$.

$v \in nvs$	Vecinos	Arista creada
2	{1, 3, 4, 5}	[5, 6]
5	{2, 3, 4}	[2, 7]
4	{1, 2, 3, 5}	[1, 8]

Tabla 4.1: Vértices expandidos

La subgráfica inducida Sb_0 esta conformada por la unión de las vecindades de cada vértice $v \in nvs$, por lo tanto:

- $V(Sb_0) = \{1, 3, 4, 5\} \cup \{2, 3, 4\} \cup \{1, 2, 3, 5\} = \{1, 2, 3, 4, 5\}$.
- $A(Sb_0) = \{[1, 2], [1, 3], [1, 4], [2, 3], [2, 4], [2, 5], [3, 4], [3, 5], [4, 5]$.

i	X	Arista creada
1	$\{6\}$	$[6,7]$
2	$\{7\}$	$[6,8]$
3	$\{8\}$	$[7,8]$

Tabla 4.2: Aristas generadas en los vértices expandidos

La gráfica generada Sb_1 por el coeficiente de expansión esta conformada por la unión de los conjuntos de vértices X_i, por lo tanto:

- $V(Sb_1) = X_1 \cup X_2 \cup X_3 = \{6\} \cup \{7\} \cup \{8\} = \{6, 7, 8\}$.
- $A(Sb_1) = \{[6, 7], [6, 8], [7, 8]\}$.

Las aristas creadas aleatoriamente entre los vértices de Sb_0 y Sb_1 son $A = \{[1, 6], [2, 7], [1, 8]\}$ (ver Tabla 4.1). La gráfica $G_0 = Sb_0 \cup Sb_1$, por lo tanto:

- $V(G_0) = V(Sb_0) \cup V(Sb_1) = \{1, 2, 3, 4, 5, 6, 7, 8\}$.
- $A(G_0) = A(Sb_0) \cup A(Sb_1) \cup A = \{[1, 2], [1, 3], [1, 4], [2, 3], [2, 4], [2, 5], [3, 4], [3, 5], [4, 5], [6, 7], [6, 8], [7, 8], [5, 6], [2, 7], [1, 8]\}$.

El homomorfismo $\pi_0 : V(G_0) \to V(G_b)$ se construye de la siguiente manera:

$$\pi_0(v) = \begin{cases} hom_0(v), & \text{si } v \in V(Sb_0) \\ hom_1(v), & \text{si } v \in V(Sb_1) \end{cases}$$

Donde:

- $hom_0(v) = v$ para cualquier vértice $v \in V(Sb_0)$.
- $hom_1(v) = u$ para cada uno de los vértices $v \in X_i$ correspondientes a $u_i \in nvs$.

$V(G_0)$	$V(G_b)$
$1 \in V(Sb_0)$	1
$2 \in V(Sb_0)$	2
$3 \in V(Sb_0)$	3
$4 \in V(Sb_0)$	4
$5 \in V(Sb_0)$	5
$6 \in V(X_1)$	$nvs_1 = 2$
$7 \in V(X_2)$	$nvs_2 = 5$
$8 \in V(X_3)$	$nvs_3 = 4$

Tabla 4.3: Homomorfismo de gráficas $\pi_0 : V(G_0) \to V(G_b)$

A continuación se muestra la gráfica resultante G_0:

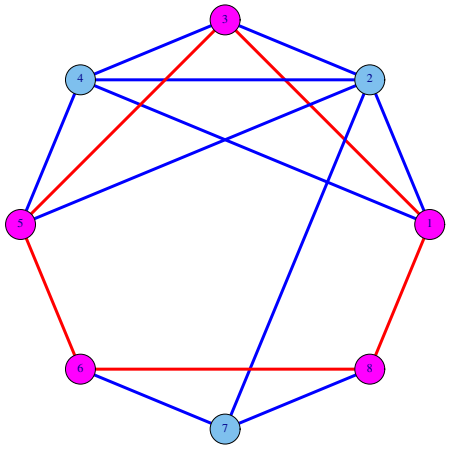


Figura 4.5: Gráfica G_0

De la Figura 4.5 los vértices en color rosa inducen un ciclo impar C_5 en la gráfica G_0 , donde sus sus aristas se encuentran coloreadas de rojo. Debido a esto, podemos estar seguros que la gráfica G_0 es imperfecta. Dado que la gráfica G_b es una gráfica inducida por el conjunto de vértices $\{1, 2, 3, 4, 5\}$ en la gráfica G_0 y existe un homomorfismo de gráficas $\pi_0 : V(G_0) \to V(G_b)$, entonces el número cromático de G_0 es igual al número cromático de G_b, es decir $\chi(G_0) = \chi(G_b) = 4$.

El algoritmo Expansión de gráficas nos ayuda a construir una gráfica G_0 imperfecta a través de una gráfica base G_b muy particular, con $\chi(G_0) = \chi(G_b)$ y un homomorfismo de gráficas $\pi_0 : V(G_0) \to V(G_b)$ que es suprayectivo para el conjunto de vértices y aristas de G_0 al conjunto de vértices y aristas de G_b respectivamente.

El parámetro para la gráfica G_1 es el siguiente:

- $G = G_b$

El vértice seleccionado para la contracción es $u = 2$. El primer vértice generado en G_1 es el super vértice $Sn = 1$, por lo que inicialmente para cada uno de los vértices $v \in V(G_b)$, $hom(v) = 1$. Como el vértice 5 no es adyacente a $Sn = 1$ en la gráfica G_b, entonces ambos vértices son excluidos de la gráfica auxiliar $Gaux$.

Después se crea el vértice $z = 2$ con la arista $[Sn = 1, z = 2]$ en la gráfica G_1. Entonces el valor de $hom_2 = 2$. La gráfica auxiliar tiene como conjunto de vértices $V(Gaux) = \{1, 2, 3, 4, 5\} - \{1, 2, 5\} = \{3, 4\}$ y su conjunto de aristas $A(Gaux) = \{[3, 4]\}$.

A continuación se crea el vértice $a = 3$ junto con las aristas $\{[a, Sn], [a, z]\}$ en la gráfica G_1. El vértice con el grado más bajo en $Gaux$ es $x = 3$. No existen vértices que no sean adyacentes a $x = 3$ en $Gaux$, entonces $hom_2(3) = 3$. El vértice adyacente a $x = 3$ es el vértice 4 en la gráfica auxiliar $Gaux$, por lo que se crea el vértice $b = 4$ y las aristas $\{[b = 4, Sn = 1], [b = 4, z = 2], [b = 4, a = 3]\}$ en la gráfica G_1.

La gráfica $Gaux$ es vacía después de eliminar el vértice 3 junto con la arista $[3, 4]$.

El homomorfismo $\pi_1 : V(G_b) \to V(G_1)$ se construye de la siguiente manera:

$V(G_b)$	$V(G_1)$
1	1
2	2
3	3
4	4
5	1

Tabla 4.4: El homomorfismo $\pi_1 : V(G_b) \to V(G_1)$

A continuación se muestra la gráfica resultante G_1:

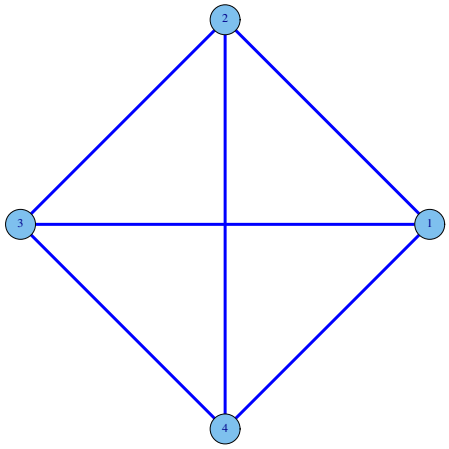


Figura 4.6: Gráfica G_1

El algoritmo Contracción de gráficas nos ayuda a construir una gráfica G_1 que es una gráfica completa de $v_b - 1$ vértices a través de una gráfica base G_b muy particular y un homomorfismo de gráficas $\pi_1 : V(G_b) \to V(G_1)$ que es suprayectivo para el conjunto de vértices y aristas de G_b al conjunto de vértices y aristas de G_1 respectivamente.

Los parámetros para la gráfica compromiso H_p son los siguientes:

- $G = G_0$
- $nvs = 8$
- $exp = 1$
- $intentos = 1$

Los vértices seleccionados $nvs = \{2, 4, 1, 8, 5, 6, 3, 7\} \in V(G_0)$.

$v \in nvs$	Vecinos	Arista creada
2	{1, 3, 4, 5, 7}	[5, 9]
4	{1, 2, 3, 5}	[1, 10]
1	{2, 3, 4, 8}	[4, 11]
8	{1, 6, 7}	[1, 12]
5	{2, 3, 4, 6}	[3, 13]
6	{5, 7, 8}	[7, 14]
3	{1, 2, 4, 5}	[5, 15]
7	{2, 6, 8}	[6, 16]

Tabla 4.5: Vértices expandidos

La subgráfica inducida Sb_2 esta conformada por la unión de los vecinos de cada vértice $v \in nvs$, por lo tanto:

- $V(Sb_2) = \{1,3,4,5,7\} \cup \{1,2,3,5\} \cup \{2,3,4,8\} \cup \{1,6,7\} \cup \{2,3,4,6\} \cup \{5,7,8\} \cup \{1,2,4,5\} \cup \{2,6,8\} = \{1,2,3,4,5,6,7,8\}$.
- $A(Sb_2) = \{[1,2],[1,3],[1,4],[1,8],[2,3],[2,4],[2,5],[2,7],[3,4],[3,5],[4,5],[5,6],[6,7],[6,8],[7,8]\}$

i	X	Arista creada
1	{9}	[9,10], [9,11]
2	{10}	[10,11], [10,13]
3	{11}	[11,12], [11,15]
4	{12}	[12,14], [12,16]
5	{13}	[9,13], [13,15]
6	{14}	[14,16], [13,14]
7	{15}	[9,15], [10,15]
8	{16}	[9,16]

Tabla 4.6: Aristas generadas en los vértices expandidos

La gráfica generada Sb_1 por el coeficiente de expansión esta conformada por la unión de los conjuntos de vértices X_i, por lo tanto:

- $V(Sb_3) = X_1 \cup X_2 \cup X_3 \cup X_4 \cup X_5 \cup X_6 \cup X_7 \cup X_8 = \{9\} \cup \{10\} \cup \{11\} \cup \{12\} \cup \{13\} \cup \{14\} \cup \{15\} \cup \{16\} = \{9,10,11,12,13,14,15,16\}$.
- $A(Sb_3) = \{[9,10],[9,11],[9,13],[9,15],[9,16],[10,11],[10,13],[10,15],[11,12],[11,15],[12,14],[12,16],[13,14],[13,15],[14,16]\}$.

Las aristas creadas aleatoriamente entre los vértices de la gráfica Sb_0 y Sb_1 son $A = \{[5,9],[1,10],[4,11],[1,12],[3,13],[7,14],[5,15],[6,16]\}$ (ver Tabla 4.5). La gráfica $H_p = Sb_2 \cup Sb_3$, por lo tanto:

- $V(H_p) = V(Sb_2) \cup V(Sb_3) = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 111, 12, 13, 14, 15, 16\}$.
- $A(H_p) = A(Sb_0) \cup A(Sb_1) \cup A = \{[1, 2], [1, 3], [1, 4], [1, 8], [2, 3], [2, 4], [2, 5], [2, 7], [3, 4], [3, 5], [4, 5], [5, 6], [6, 7], [6, 8], [7, 8], [9, 10], [9, 11], [9, 13], [9, 15], [9, 16], [10, 11], [10, 13], [10, 15], [11, 12], [11, 15], [12, 14], [12, 16], [13, 14], [13, 15], [14, 16], [5, 9], [1, 10], [4, 11], [1, 12], [3, 13], [7, 14], [5, 15], [6, 16]\}$

El homomorfismo $\psi : V(H_p) \to V(G_0)$ se construye de la siguiente manera:

$$\psi_3(v) = \begin{cases} hom_0(v), & \text{si } v \in V(Sb_2) \\ hom_1(v), & \text{si } v \in V(Sb_3) \end{cases}$$

Donde:

- $hom_0(v) = v$ para cualquier vértice $v \in V(Sb_2)$.
- $hom_1(v) = u$ para cada uno de los vértices $v \in X_i$ correspondientes a $u_i \in nvs$.

$V(H_p)$	$V(G_0)$
$1 \in V(Sb_2)$	1
$2 \in V(Sb_2)$	2
$3 \in V(Sb_2)$	3
$4 \in V(Sb_2)$	4
$5 \in V(Sb_2)$	5
$6 \in V(Sb_2)$	6
$7 \in V(Sb_2)$	7
$8 \in V(Sb_2)$	8
$9 \in X_1$	nvs_1=2
$10 \in X_2$	nvs_2=4
$11 \in X_3$	nvs_3=1
$12 \in X_4$	nvs_4=8
$13 \in X_5$	nvs_5=5
$14 \in X_6$	nvs_6=6
$15 \in X_7$	nvs_7=3
$16 \in X_8$	nvs_8=7

Tabla 4.7: Homomorfismo de gráficas $\psi : V(H_p) \to V(G_0)$

A continuación se muestra la gráfica resultante H_p:

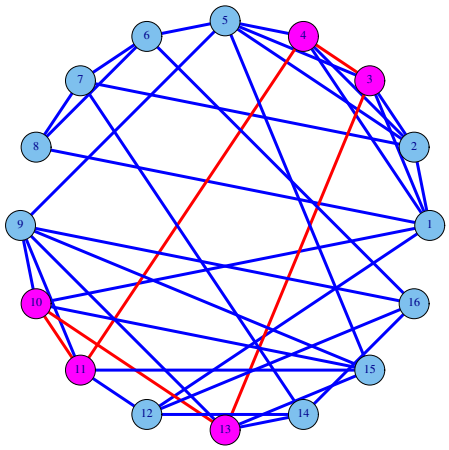
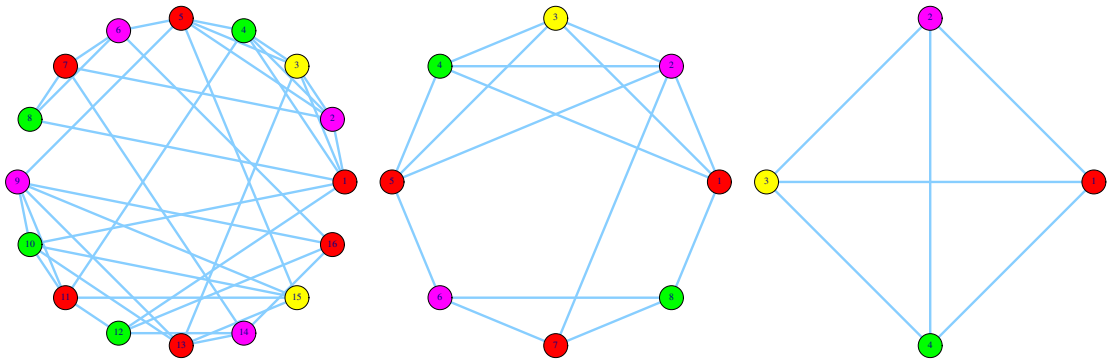


Figura 4.7: Gráfica H_p

De la Figura 4.7 los vértices en color rosa inducen un ciclo impar C_5 en la gráfica H_p , donde sus sus aristas se encuentran coloreadas de rojo. Debido a esto, podemos estar seguros que la gráfica H_p es imperfecta. Dado que la gráfica G_0 es una gráfica inducida por el conjunto de vértices $\{1, 2, 3, 4, 5, 6, 7, 8\}$ en la gráfica H_p y existe un homomorfismo de gráficas $\psi : V(H_p) \to V(G_0)$, entonces el número cromático de H_p es igual al número cromático de G_0, es decir $\chi(H_p) = \chi(G_0) = \chi(G_b) = 4$.



(a) H_p (b) G_0 (c) G_1

Figura 4.8: Gráficas construidas en la generación de claves

De la Figura 4.8 se puede observar que un intruso que quisiera quebrantar el protocolo tendría que resolver el problema de coloración propia de las gráficas H_p y G_0. Es difícil decir con exactitud para que clases de gráficas no existe un algoritmo que resuelva el problema de coloración propia en complejidad temporal polinomial. En cambio, en la literatura especializada se dice que el problema de coloración propia de una gráfica G es resuelto en complejidad temporal polinomial si la gráfica G es perfecta. En seguida se mencionan las clases de gráficas que son perfectas: gráficas bipartitas y sus complementos, gráficas cordales, gráficas de línea de gráficas bipartitas y sus complementos, gráficas de barbell, gráficas alfil, gráficas cavernícolas, gráficas completas, gráficas ventilador, gráficas de hanoi, gráficas rueda, gráficas timón, gráficas enrejadas, gráficas

rey, gráficas de sol, gráficas reinas, gráficas de Turán, permutaciones de gráficas estrellas, etc. Por lo anterior, en esta tesis garantizamos que la gráfica G_0 y H_p sean imperfectas. Además, el homomorfismo entre nuestras gráficas debe ser suprayectivo en relación al conjunto de vértices y aristas. Se utilizo el algoritmo glotón de Welsh y Powell sobre las gráficas G_0 y H_p con los parámetros: $v_b = [10, 20], nvs = 3, exp = 1, intentos = 1$ y un número de 1000 ejecuciones por cada variación del parámetro v_b, el algoritmo glotón no fue capaz de encontrar la coloración propia de nuestras gráficas, en `http://computacion.cs.cinvestav.mx/~jbastida/` se pueden consultar los resultados y datos estadísticos sobre las pruebas realizadas.

La composición de homomorfismos $\pi = \pi_1 \circ \pi_0$ y $\Psi = \pi \circ \psi$, donde $\pi : V(G_0) \to V(G_1)$ es la clave privada y $\Psi : V(H_p) \to V(G_1)$ es uno de los desafíos :

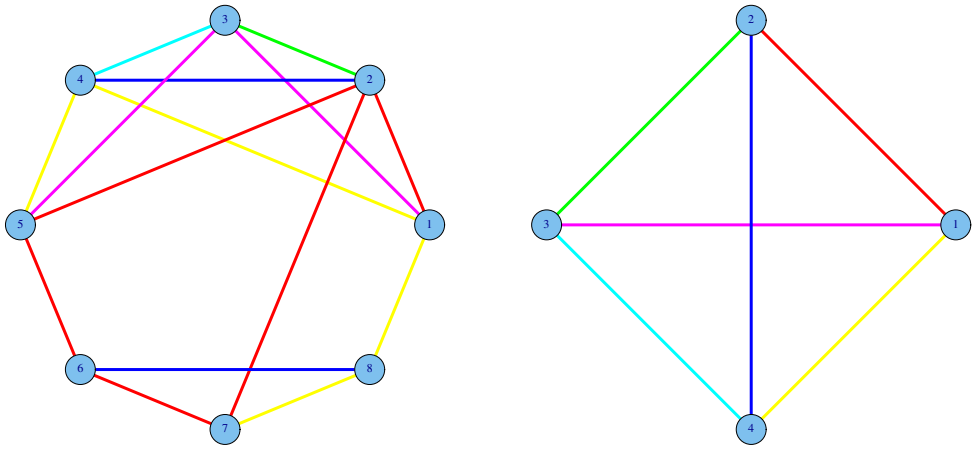
$V(G_0) \to$	$V(G_b) \to$	$V(G_1)$
1	1	1
2	2	2
3	3	3
4	4	4
5	5	1
6	2	2
7	5	1
8	4	4

Tabla 4.8: Homomorfismo de gráficas $\pi : V(G_0) \to V(G_1)$

$V(H_p) \to$	$V(G_0) \to$	$V(G_b) \to$	$V(G_1)$
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	1
6	6	2	2
7	7	5	1
8	8	4	4
9	2	2	2
10	4	4	4
11	1	1	1
12	8	4	4
13	5	5	1
14	6	2	2
15	3	3	3
16	7	5	1

Tabla 4.9: Homomorfismo de gráficas $\Psi : V(H_p) \to V(G_1)$

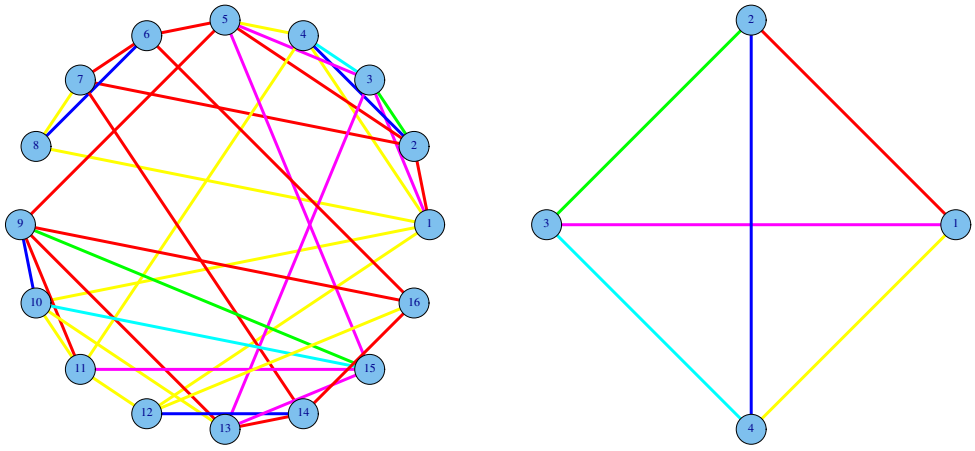
Como $\pi : V(G_0) \to V(G_1)$, entonces se produce una relación $\pi_A : A(G_0) \to A(G_1)$, con $\pi_A([u,v]) = [\pi(u), \pi(v)]$, y podemos definir la gráfica $\pi(G_0) = (\pi(V(G_0)), \pi_A(A(G_0)))$ como la imagen homomorfa de G_0 a G_1:



(a) Gráfica G_0 (b) Gráfica G_1

Figura 4.9: Relación entre las aristas de G_0 y G_1

Como $\psi : V(H_p) \to V(G_1)$, entonces se produce una relación $\psi_A : A(H_p) \to A(G_1)$, con $\pi_A([u,v]) = [\pi(u), \pi(v)]$, y podemos definir la gráfica $\psi(H_p) = (\psi(V(H_p)), \psi_A(A(G_0)))$ como la imagen homomorfa de H_p a G_1:



(a) Gráfica H_p (b) Gráfica G_1

Figura 4.10: Relación entre las aristas de H_p y G_1

De las Figuras 4.8, 4.9 y 4.10 se puede observar que los homomorfismos de gráficas en nuestras gráficas es suprayectivo en relación al conjunto de vértices y aristas.

Para finalizar, se muestra la forma compacta de la clave pública y privada de este ejemplo:

```
----BEGIN PUBLIC KEY----

7L6he$

-----END PUBLIC KEY-----

----BEGIN PRIVATE KEY----

01230103

-----END PRIVATE KEY-----
```

Figura 4.11: Nombramiento de la clave pública y privada

La aplicación genera un archivo de texto diferente para cada una de las gráficas y sus homomorfismo, así como un `script` en R que permite visualizar la correspondiente coloración de aristas y vértices entre los homomorfismos generados. Por último se genera un archivo de texto que contiene la clave pública con el formato mostrado en la Figura 4.6.

La implementación del protocolo propuesto, los algoritmos que expanden y contraen vértices, así como el desempeño de la generación de claves y datos estadísticos sobre las pruebas realizadas pueden ser descargados/consultados de la siguiente página `http://computacion.cs.cinvestav.mx/~jbastida/`.

Capítulo 5

Robustez del protocolo

Uno de los principales propósitos de la autenticación de usuarios es facilitar el control de acceso a recursos, cuando los privilegios están ligados a una identidad en particular. Un esquema basado en contraseñas es usado para permitir el acceso mediante cuentas de usuarios (a computadoras por ejemplo), lo cual puede ser visto como la manera más simple de un control de acceso, donde cada recurso tiene una lista de identidades asociadas con él y la comprobación exitosa de identidad permite el acceso autorizado a esos recursos listados para esa entidad. En muchas aplicaciones (en celulares por ejemplo) la motivación por la autenticación de usuarios permite el uso de recursos para realizar un seguimiento de las identidades identificadas. La identificación es también típicamente un requisito inherente a protocolos de establecimiento de claves autenticadas.

Pruebas interactivas usadas para la autenticación de usuarios pueden ser formuladas como pruebas de conocimiento. Una parte, digamos A, posee algún secreto s que le sirve como una partícula de identidad, e intenta convencer a una segunda parte, llamémosla B, de que posee dicho secreto, mediante una serie de desafíos (que implica conocer públicamente una serie de entradas y funciones acordadas previamente) los cuales pueden ser resueltos únicamente mediante el conocimiento de s. Note que la prueba de conocimiento de s difiere de demostrar que tal s existe, por ejemplo, probar el conocimiento de la factorización en primos de n difiere de probar que n es un número compuesto. La cantidad de desafíos t que deben ser elaborados para que B quede convencido de que A posee el secreto s y, en consecuencia, de la autenticidad de A puede variar según la especificación del protocolo.

5.1. Posibles ataques

El protocolo propuesto está basado en la dificultad de resolver el problema de homomorfismo de gráficas. Consideremos dos gráficas simples $S_0 = (V(S_0), A(S_0))$ y $S_1 = (V(S_1), A(S_1))$. Un homomorfismo de gráficas restrictivo es una función $g : V(S_0) \to V(S_1)$ del conjunto de vértices de S_0 al conjunto de vértices de S_1, tal que aristas (par de vértices distintos) se transforman efectivamente en aristas (par de vértices distintos) bajo la función g, es decir:

- Si $[u, v] \in A(S_0)$ es una arista en S_0, entonces $[g(u), g(v)] \in S_1$ es una arista en S_1, con $g(u) \neq g(v)$.

- Para dos vértices cualesquiera $u, v \in V(S_0)$, si las imágenes de u y v son iguales $g(u) = g(v)$, entonces $[u, v] \notin A(S_0)$.

Sean dos gráficas simples $S_0 = (V(S_0), A(S_0)$ y $S_1 = (V(S_1), A(S_1)$. Una función $\pi : V(S_0) \to V(S_1)$ es cocolapsable si cumple con las siguientes propiedades:

1. La función π es un homomorfismo de gráficas restrictivo, es decir:
   - $[u, v] \in A(S_0) \Rightarrow \pi(u) \neq \pi(v)$ & $[\pi(u), \pi(v)] \in A(S_1)$.
   - $u, v \in V(S_0)$ & $\pi(u) = \pi(v) \Rightarrow [u, v] \notin A(S_0)$.
2. La función π es una función suprayectiva, es decir:
   - $\forall u \in V(S_1)$, $\exists v \in V(S_0)$: $\pi(v) = u$.
   - $\forall [u, v] \in A(S_1)$, $\exists [x, y] \in A(S_0)$: $[\pi(x), \pi(y)] = [u, v]$

Decimos cocolapsable debido a dos razones:

1. Porque el homomorfismo que se construye entre las gráficas $H_p \to G_0 \to G_1$ se puede ver como una **co**loración propia de cada una de las gráficas (identificando vértices del mismo color).
2. Porque los vértices que no forman aristas en G, es decir que determinan aristas en la gráfica complementaria $\overline{G}$, son **colapsables** en un mismo punto.

Sean dos gráficas simples $G = (V_G, A_G)$ y $H = (V_H, A_H)$, la gráfica G es colapsable si y sólo si existe una partición P del conjunto de vértices V_G tal que:

1. Cada subconjunto $Sub \in P$ es un conjunto independiente.
2. $\mathrm{card(P)} = \mathrm{card(V_H)}$.
3. La gráfica cociente $G' = G/P$ es isomorfa a H.

Sean dos gráficas simples G y H. Una función $\pi : V(G) \to V(H)$ es un homomorfismo de gráficas si y sólo si la pre-imagen $\phi^{-1}(I)$ de cada subconjunto independiente I de $V(H)$ es un conjunto independiente.

Sea G una gráfica y $P = V1, \ldots V_k$ una partición del conjunto de vértices de G, entonces π_P es un homomorfismo de gráficas si y sólo si V_i es un conjunto independiente, donde $1 \leq i \leq k$.

Para cada homomorfismo $\phi : V(G) \to V(H)$ existe una partición P del conjunto de vértices $V(G)$ en conjuntos independientes y un monomorfismo $\psi : V(G/H) \to V(H)$ tal que $\phi = \psi \circ \pi_P$.

Si la cardinalidad de la partición P es igual a la cardinalidad de $V(H)$, entonces ψ es un isomorfismo.

El homomorfismo de gráficas de G_0 a G_1 construido por el protocolo propuesto es una función cocolapsable, por lo que un ataque a nuestro protocolo se reduciría a decidir si la gráfica G_0 es colapsable. Lo mismo sucede en relación al reto y la gráfica compromiso H_p.

Para las clases de gráficas $\mathcal{G}$ y $\mathcal{H}$ denotamos por HOM($\mathcal{G}$, $\mathcal{H}$) al problema de decidir para dos gráficas dadas $G \in \mathcal{G}$ y $H \in \mathcal{H}$ si acaso G es homomorfa a H (noción de homomorfismo restrictivo). Si $\mathcal{G} = \{G\}$ o $\mathcal{H} = \{H\}$, entonces escribimos G y H en vez de $\mathcal{G}$ y $\mathcal{H}$ respectivamente. Si $\mathcal{G}$ o $\mathcal{H}$ es la clase de todas las gráficas, entonces las denotamos por "_". La complejidad computacional del problema de homorfismo de gráficas ha sido estudiada por diferentes "lados".

Homomorfismos de lado izquierdo Para cualquier gráfica fija G, HOM(G, _) se resuleve trivialmente en complejidad temporal polinomial. Muchos autores han mostrado independientemente que HOM($\mathcal{G}$, _) se resuelve en complejidad temporal polinomial si todas las gráficas en $\mathcal{G}$ poseen anchuras de árbol (*treewidth*) acotadas [14].

Homomorfismos de lado derecho En [12, 13] se muestra que para cualquier gráfica fija H, el problema HOM(_, H) se resuelve en complejidad temporal polinomial si H es bipartita. En tanto que el problema de decisión es completo-NP si H no fuera bipartita. Por lo tanto, el estudio de homomorfismos de lado derecho se ha dejado a un lado, mientras que las investigaciones se han concentrado en encontrar algoritmos de complejidad temporal que resuelvan homomorfismos de lado izquierdo para clases de gráficas especiales.

Si H está en la clase de gráficas completas, entonces HOM(_, H) es equivalente al problema de k-coloración de gráficas [12, 13]. Recientemente se ha realizado un trabajo muy extenso sobre el problema de la k-coloración de gráficas, lo que equivale a un caso especial de homomorfismo de lado derecho de gráficas, dando como resultado algoritmos de complejidad temporal exponencial cada vez más rápidos [31]. Los mejores resultados que se han conseguido, presentados de manera muy sintetizada, son los siguientes:

k	Complejidad
3	$O(1.3289^n)$[28]
4	$O(1.7504^n)$ [30]
5	$O(2.1020^n)$
6	$O(2.1809^n)$
≥ 7	$O(2.4023^n)$ [30]

(donde n es el número de vértices de la gráfica)

Tabla 5.1: Complejidad temporal de los mejores algoritmos que resuelven el problema de la k-coloración propia de gráficas

Por otra parte, la existencia de un homomorfismo entre dos gráficas dadas, G y H, puede ser resuelta en complejidad temporal $O(bw(\overline{H}) + 2)^2)$ [26], donde n es el numero de vértices de G y $bw(\overline{H})$ es el ancho de banda (*bandwidth*) de la gráfica complementaria de H.

Obsérvese que al resolver HOM($\mathcal{G}$, $\mathcal{H}$) se podría obtener la clave privada utilizada en el protocolo propuesto y con ello se lograría usurpar la identidad del poseedor de la correspondiente clave pública, es decir, se rompería el esquema de seguridad del protocolo.

Un homomorfismo f de un gráfica $G \in \mathcal{G}$ a $H \in \mathcal{H}$ es una relación entre vértices tal que para cada $v \in V(H)$:

1. existe exactamente un vértice $u \in V(G)$ tal que $f(u) = v$ si f es biyectiva,

2. existe como mucho un vértice $u \in V(G)$ tal que $f(u) = v$ si f es inyectiva,

3. existe al menos un vértice $u \in V(G)$ tal que $f(u) = v$ si f es suprayectiva.

El problema de decisión correspondiente a la primera variante es conocida como el *problema de isomorfismo de subgráficas* (*Spanning subgraph isomorphism problem*), el problema de decisión correspondiente a la segunda variante es conocida como el *problema de isomorfismo de subgráficas* (*Subgraph isomorphism problem*). La tercer variante denotada como SURHOM$(\mathcal{G}, \mathcal{H})$ se estudia en [32], en donde se esquematiza la tabla siguiente:

$\mathcal{G}$	$\mathcal{H}$	Complejidad
Gráficas completas	Todas las gráficas	Polinomial
Todas las gráficas	Gráfica lineal	Polinomial
Gráfica lineal	Todas las gráficas	completo-NP
Bosque lineal	Bosque lineal	completo-NP
Unión de gráficas completas	Unión de gráficas completas	completo-NP
Cográficas conectadas	Cográficas conectadas	completo-NP
Gráficas de intervalos adecuados conectados	Gráficas de intervalos adecuados conectados	completo-NP
Separación de gráficas (*Split graphs*)	Separación de gráficas	completo-NP

En la primera columna aparece el tipo de gráficas que constituye a $\mathcal{G}$, en la segunda el que constituye a $\mathcal{H}$ y en la tercera, la complejidad temporal del problema SURHOM$(\mathcal{G}, \mathcal{H})$.

Tabla 5.2: Complejidad temporal de SURHOM$(\mathcal{G}, \mathcal{H})$

La construcción de homomorfismo suprayectivos depende de como es construida la gráfica G_0, es decir, si el conjunto de vértices $V(Sb_0) = V(G_b)$, entonces se tiene un homomorfismo de gráficas suprayectivo $\phi : V(G_0) \to V(G_1)$.

El parámetro que nos permite generar un homomorfismo suprayectivo $\phi : V(G_0) \to V(G_1)$ es con $nvs \leq 3$.

Note que una solución a SURHOM$(\mathcal{G}, \mathcal{H})$ comprometería la robustez de nuestro protocolo si las gráficas G_0 y G_1 son construidas en la clase de gráficas completas para G_0 y para G_1 en la clase de gráficas lineales.

De los resultados experimentales donde la construcción del homomorfismo fue suprayectivo, en ninguna ejecución se construyo una gráfica G_0 que fuese completa, por otro lado tampoco sucedió que la gráfica G_1 fuese bipartita. Una gráfica lineal es un tipo especial de árbol, todo árbol es una gráfica bipartita, por lo que si G_1 no es bipartita, entonces no hay manera de que G_1 fuese una gráfica lineal.

Por lo anterior y en base a los resultados mostrados en la figura 5.2, la solución a SURHOM$(\mathcal{G}, \mathcal{H})$ en nuestro protocolo se mantiene en la clase completo$-NP$.

5.2. Análisis de seguridad

Los algoritmos criptográficos pueden ofrecer diferentes "fortalezas"de seguridad, dependiendo del algoritmo y el tamaño de la clave usada. Dos algoritmos son considerados comparables en seguridad para un tamaño de claves (X y Y), si la cantidad de trabajo necesario para quebrantar los protocolos o determinar las claves (con el tamaño de las claves X y Y) es aproximadamente el mismo usando los mismos recursos computacionales.

La seguridad de un algoritmo para un tamaño de clave es tradicionalmente descrito en términos de la cantidad de trabajo que toma en probar todas las posibles claves para un algoritmo de criptografía asimétrico con un tamaño de clave X que no tiene ningún tipo de ataque por atajos (*short cut attacks*), es decir, el ataque más eficiente consiste de probar todo el espacio de claves del algoritmo. En este caso, el mejor ataque es conocido como ataque por agotamiento (*exhaustion attacks*).

Un algoritmo que tiene un clave de tamaño Y bits, pero que es comparable con la seguridad de una clave de tamaño X bits para un algoritmo de criptografía asimétrica, se dice que tiene una seguridad de X bits o que puede ofrecer una seguridad de X bits. Un algoritmo que ofrece una seguridad de X bits, en promedio puede tomar un tiempo $2^{X-1}T$ por ataque, donde T es la cantidad de tiempo requerido para realizar las operaciones realizadas en el protocolo. Determinar la seguridad de un algoritmo puede ser difícil.

Debido a la construcción de nuestras gráficas, la k-coloración propia es el mejor ataque que se puede plantear al protocolo propuesto.

A continuación presentamos los parámetros necesarios en nuestro protocolo para obtener un nivel de seguridad de 128, 192 y 256 bits, utilizando en cada caso la generación de un homomorfismo suprayectivo y gráficas G_1 en la clase de gráficas completas. Cabe mencionar que la gráfica $G0$ se obtiene a partir de la expansión de una gráfica auxiliar, donde la gráfica auxiliar es una expansión de la gráfica base Gb con los siguientes parámetros: $nvs = v_b$, $exp = 3$ e $intentos = 5$.

Los parámetros necesarios para obtener algunos niveles de seguridad:

Seguridad en bits	v_b	nvs	exp	$intentos$	$n = V(G_0)$	Ataque de k-coloración
$2^{128} \approx 3.402 \times 10^{38}$	9	33	2	1	102	$2.4023^n \approx 6.667 \times 10^{38}$
$2^{192} \approx 6.277 \times 10^{57}$	8	29	4	1	152	$2.4023^n \approx 7.167 \times 10^{57}$
$2^{256} \approx 1.157 \times 10^{77}$	9	34	5	1	206	$2.4023^n \approx 2.565 \times 10^{78}$
$2^{512} \approx 1.34 \times 10^{154}$	8	32	12	1	416	$2.4023^n \approx 2.192 \times 10^{158}$

Tabla 5.3: Parámetros sugeridos para obtener diversos niveles de seguridad

Para obtener una seguridad de 128 bits, se desea encontrar una n tal que $2^{128} \leq 2^n$, lo que resulto en $n = 102$. Por lo que el número de vértices de $V(G_0) = n = v_b + (nvs \times exp)$, por lo que basta fijar dos de los parámetros para encontrar el tercero. La manera más sencilla es fijar v_b y $nvs \leq 3$ para construir el homomorfismo suprayectivo y por último obtener exp. El resto de los niveles de seguridad se pueden obtener de la misma manera.

La longitud de la clave en bits en relación al nivel de seguridad que puede ofrecer nuestro protocolo se muestra en la siguiente tabla, además se mencionan cuatro protocolos

criptográficos más: dos sobre criptografía de campos finitos (*FFC, Finite field cryptography*) *(DSA, Digital Signature Algorithm)* y Diffie-Hellman (DH), uno basado en criptografía de factorización entera (*IFC, Integer factorization criptography*) Rivest-Shamir-Adleman (RSA), uno basado en criptografía de curvas elípticas (*ECC, Elliptic curv criptography*) *(ECDSA, Elliptic Curve Digital Signature Algorithm)* y el protocolo desarrollado en esta tesis basado en el homomorfismo de gráficas.

Bits de seguridad	FFC DSA y DH	IFC RSA	ECC ECDSA	Nuestro protocolo
128	$L = 3072$ $N = 256$	$k = 3072$	$f = 256 - 383$	$C_{pub} = 7372$ $C_{pri} = 102$
192	$L = 7680$ $N = 384$	$k = 7680$	$f = 384 - 511$	$C_{pub} = 16240$ $C_{pri} = 152$
256	$L = 15360$ $N = 512$	$k = 15360$	$f = 512+$	$C_{pub} = 28132$ $C_{pri} = 203$

Tabla 5.4: Comparación de longitudes de clave entre otros protocolos y el nuestro

La Tabla 5.4 proporciona niveles de seguridad comparables para los algoritmos aprobados por el Instituto Nacional de Estándares y Tecnologías (*NIST, National institute of standards and technology*) y el protocolo que proponemos.

- La primer columna indica el numero de bits de seguridad que pueden ofrecer los algoritmos y el tamaño de la llave en particular. Note que los bits de seguridad no son necesariamente la misma que el tamaño de los llaves que los algoritmos emplean en las otras columnas, debido a que existen ataques que proporcionan ventajas computacionales.
- La segunda columna nos indica el tamaño mínimo de los parámetros asociados a los estándares usados en la criptografía de campos finitos. Ejemplos de tales algoritmos son DSA Y DH, donde L es el tamaño de la clave pública y N es el tamaño de la clave privada.
- La tercer columna indica el valor para k (el tamaño del modulo n) para algoritmos basados en factorización entera. El algoritmo predominante de este tipo de algoritmos es el RSA. El valor de k es considerado comúnmente como el tamaño de la clave.
- La cuarta columna indica el rango de f (el tamaño de n, donde n es el orden de la base del punto G) para algoritmos basados en criptografía de curvas elípticas. El valor de f es considerado comúnmente como el tamaño de la clave.
- Por último, en la quinta columna se indica el tamaño de la clave pública y privada del protocolo que proponemos. Donde C_{pub} es la clave pública, la cual consiste de un par de gráfica G_0 y G_1; y C_{pri} es la clave privada, la cual consiste de un homomorfismo de gráficas $\phi : G_0 \to G_1$.

De la tabla anterior se puede observar que la longitud en bits de nuestras claves son aproximadamente dos veces mas grandes en comparación a los otros protocolos. Por lo tanto, la consideración del espacio de almacenamiento requerido por nuestras claves es importante.

Los resultados de los ataques realizados al protocolo pueden consultarse en la página `http://computacion.cs.cinvestav.mx/~jbastida/`.

Capítulo 6

Conclusiones y perspectivas

En este trabajo de tesis se implementó y desarrolló un protocolo de autenticación, el cual fue desarrollado sobre el lenguaje `C` y el uso de la biblioteca `Igraph`. En este capítulo mencionaremos las conclusiones de nuestro trabajo y algunas perspectivas para el trabajo futuro.

6.1. Conclusiones

Este trabajo presenta la implementación y desarrollo de un protocolo de autenticación basada en el problema de homomorfismo de gráficas mediante esquemas del tipo desafío-respuesta. Hasta el momento solo se conocían las bases teóricas de lo que podría ser un protocolo de autenticación con las características antes mencionadas. Nunca se abordo el tema de como debían construirse las gráficas, cual debía ser el tamaño y orden de las gráficas involucradas, ni mucho menos las familias de las gráficas que pudieran haberse tomado como base de construcción.

En este trabajo se da a conocer la manera en que las gráficas se construyen a partir de una gráfica aleatoria tomada como base. Los tipos de homomorfismos "intermedios"que se logran construir hacia cada una de las gráficas involucradas en el protocolo.

En base a las pruebas experimentales se pudo establecer la clase de gráficas que se debían de considerar para garantizar diferentes niveles de seguridad. También se propone una codificación de gráficas que nos permite minimizar la complejidad del almacenamiento y poder tener una representación de las claves lo más pequeñas posibles.

Se definieron parámetros de control para el crecimiento de vértices y aristas en las gráficas G_0 y H_p, con el fin de que se pueda identificar fácilmente un posible ataque mediante el producto tensorial.

Otro resultado importante que se pudo observar gracias a las pruebas experimentales, es que se pueden construir homomorfismos suprayectivos, el cual es considerado en la clase completo$-NP$ para la mayoría de las clases de gráficas. Por otro lado, los otros dos tipos de homomorfismo (inyectivo y biyectivo) resultan ser mas "sencillos" en general.

Se da a conocer la manera en que se puede estimar el orden de las gráficas, así como la longitud de la clave (pública y privada) en relación a los parámetros empleados.

Por último, la aplicación también muestra la imagen homomorfa de G_0 en G_1 mediante la coloración de aristas. Por lo anterior, es más sencillo observar la existencia de un homomorfismo suprayectivo entre ambas gráficas. Cabe mencionarse que la aplicación genera un `scrip` en `R` para visualizar lo antes mencionado.

6.2. Perspectivas

A continuación se describen algunas observaciones para trabajo futuro en relación a la tesis:

1. Consideración de otras técnicas para la construcción de la gráfica base.
2. Estudiar posibles codificaciones para obtener longitudes de claves más cortas. Hasta el momento el número de vértices de la gráfica G_1 se encuentra restringido a 64, se podría emplear bloques de 12 bits en vez de 6 para ampliar el número de vértices, pero esto resultaría en claves privadas del doble de longitud. Por lo anterior, sería bueno pensar en una codificación alternativa para la clave privada.
3. Realizar un criptoanálisis más profundo, es decir tomar en consideración todos los algoritmos que resuelven el problema de coloración apropiada de gráficas y sus combinaciones, con el fin de quebrantar el protocolo propuesto.

El software descrito en esta tesis, los algoritmos que expanden y contraen vértices, así como el desempeño de la generación de claves, datos estadísticos sobre las pruebas realizadas, resultados de los ataques mencionados al protocolo propuesto pueden ser descargados/consultados de la siguiente página `http://computacion.cs.cinvestav.mx/~jbastida/`.

Bibliografía

[1] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1996.

[2] A. Myasnikov, V. Shpilrain, and A. Ushakov. *Non-commutative Cryptography and Complexity of Group-theoretic Problems*, 2011.

[3] Hannu A. Aronsson. *Zero Knowledge Protocols and Small Systems*. Department of Computer Science. Helsinki University of Technology.

[4] K. Arthi, N. Nandhitha, and S. Emalda Roslin, *A Study and Evaluation of Different Authentication Methods and Protocols*. International Journal of Computer Science and Management Research, Vol 2, Issue 1, January 2013, p. 1298-1302

[5] G. Hahn and C. Tardif. *Graph homomorphisms: structure and symmetry*. Graph symmetry, ASI ser C, Kluwer, 1997, pp. 107-166.

[6] Herish O. Abdullah and Mohammad Eftekhari. *Cryptanalysis and improvements on some graph-based authentication schemes*. Cryptography and Security, 27 Sep. 2012.

[7] Whitfield Difie and Martin E. Hellman. *New directions in cryptography*. IEEE Transactions on Information Theory, 22:644-65, June 3, 1976.

[8] R. Merkle. *Secure communication over an insecure channel*, Communications of the ACM, April 1978, Vol. 21, No. 4.

[9] D. Grigoriev and V. Shpilrain. *Authentication schemes from actions on graphs, groups, or rings*, Annals of Pure and Applied Logic 162, January 14, 2010, p. 194-200.

[10] S. Goldwasser, S. Micali, and C. Rackoff. *The knowledge complexity of interactive proof systems*. Proceedings of the seventeenth anual ACM symposium on theory of computing, September 1985, pp. 291-304.

[11] O. Goldreich, S. Micali, and A. Wigderson. *Proofs that Yield Nothing But Their Validity All Languages in NP Have Zero-Knowledge Proof Systems*. Journal of the ACM, Vol. 38, No. 1, pages 691-729, July 1991.

[12] P. Hell and J. Nešetřil. *On the complexity of H-coloring*. Journal of Combinatorial Theory Series B 48 (1990), 92-110.

[13] P. Hell and J. Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.

[14] Josep Díaz, Maria Serna, and Dimitrios M. Thilikos. *Counting H-colorings of partial k-trees.* Theoretical Computer Science 281 (2002) 291-309.

[15] Igraph page. URL `http://igraph.org/redirect.html`.

[16] Igraph Reference Manual for using the igraph C library. URL `http://igraph.org/c/doc/`.

[17] R Igraph manual pages. URL `http://igraph.org/r/doc/`.

[18] Dániel Marx. *Algorithmic graph structure theory.* Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SZTAKI) Budapest, Hungary. STACS 2013 Tutorial. February 27, 2013. Kiel, Germany.

[19] Ch. Borgs, J. Chayes, L.Lovász, V. T. Sós, and K. Vesztergombi. *Counting graph homomorphisms*, pp.315-371 in Topics in Discrete Mathematics (Algorithms and Complexity 26), ed. M. Klazar et al., Springer, 2006.

[20] Richard Hammack, Wilfried Imrich, and Sandi Klavžar. *Handbook of Product Graphs*, CRC Press, 2011.

[21] Godsil C. and Royle G. *Algebraic graph theory.* Springer, 2001.

[22] László Babai, Paul Erdös and Stanley M. Selkow. *Random Graph Isomorphism*, SIAM Journal on Computing, 9(3), 1980, p. 628-635.

[23] Saucy3. URL `http://vlsicad.eecs.umich.edu/BK/SAUCY/`

[24] B. D. Mckay. *Practical graph isomorphism*, Congr. Numer. 30, 1981, p 45-87.

[25] Tommi Junttila and Petteri Kaski. *Engineering an efficient canonical labeling tool for large sparse graphs*, proceeedings of the ninth workshop on Algorithm Engineering and Experiments, SIAM, 2007, p. 135-149.

[26] Pawel Rzazewski. *Exact Algorithm for Graph Homomorphism and Locally Injective Graph Homomorphism*, Information Processing Letters, Volume 114, Issue 7, July 2014, Pages 387-391.

[27] Fedor V. Fomin, Pinar Geggerner, and Dieter Kratsh. *Exact Algorithms for Graph Homomorphisms*, Theory of Computing Systems 41 (2007), p. 381-393.

[28] R. Beigel and D. Eppstein. *3-coloring in time* $O(1.3289^n)$. J. Algorithms, 54(2): 168-204, 2005.

[29] Guillermo Benito Morales Luna. Criptografía II: Criptografía no-conmutativa, 2011. URL `http://delta.cs.cinvestav.mx/~gmorales/catalogo/CriptoII/`

[30] J. M. Byskov. *Enumerating maximal independent sets with applications to graph colouring.* Operations Research Letters, 32:547-556, 2004.

[31] Andread Björklund, Thore Husfeldt, and Mikko Koivisto, *Set partitioning via inclusion-exclusion*, SIAM J. Comput. 39 (2009), pp. 546-563.

[32] Petr A. Golovach, Bernard Lidický, Barnaby Martin and Daniël Paulusma. *Finding vetex-surjective graph homomorphism* Discrete Mathematics, 2012.

[33] Elaine Barker, William Barker, William Burr, William Polk and Miles Smid. *Recommendation for Key Management-Part 1: General* NIST Special Publication 800-57, March, 2007.

[34] M. Garey, J. Johnson, *Computers and Intractability, A Guide to NP-Completeness*, W. H. Freeman, 1979.