Universidad de Buenos Aires
Facultades de Ciencias Económicas, Ciencias Exactas y Naturales e Ingeniería

Carrera de Especialización en Seguridad Informática

Trabajo Final

Criptografía post cuántica

Análisis de la criptografía asimétrica en la era post cuántica

Autor: Cevallos García Rick
Tutor de Trabajo Final: PhD. Pedro Hecht

2018
Cohorte 2017

Declaración Jurada de origen de los contenidos

“Por medio del presente, el autor manifiesta conocer y aceptar el Reglamento de Trabajos Finales vigente y que se hace responsable que la totalidad de los contenidos del presente documento son originales y de su creación exclusiva, o bien pertenecen a terceros u otras fuentes, que han sido adecuadamente referenciados y cuya inclusión no infringe la legislación Nacional e Internacional de Propiedad Intelectual”.

_____

Rick Marcel Cevallos García

C.I. 0.925.351.181 Guayaquil, Ecuador

DNI. 90.712.064 CABA., Argentina

Resumen

La presente investigación muestra una recopilación de los principales algoritmos criptográficos asimétricos existentes, sus falencias ante ataques utilizando computación cuántica, y las soluciones planteadas ante esta amenaza.

Se comienza con un recorrido por los algoritmos más importantes en la computación clásica, detallando tanto sus usos como seguridad matemática luego se analiza la computación cuántica y las vulnerabilidades que plantea en los mismos, continúa en el último capítulo con las soluciones planteadas hasta el momento, y finaliza con las conclusiones.

Se deja constancia que parte del presente análisis está fundamentado en base a los libros “Handbook of Applied Cryptography” de Alfred Menezes, considerado la biblia de la criptografía, “Post-Quantum Cryptography” de Daniel J. Bernstein & Johannes Buchmann y la tesis doctoral de Stefan Heyse con el nombre: “Post Quantum Cryptography: Implementing Alternative Public Key Schemes on Embedded Devices”, los cuales se detallan en la bibliografía general. Se recomienda antes de comenzar la lectura de este texto, familiarizarse con las bases matemáticas de la criptografía utilizada, para ello se aconseja leer el segundo y tercer capítulo del primer libro mencionado.

Palabras claves

Algoritmos criptográficos asimétricos, computación clásica, computación cuántica, vulnerabilidad.

Índice General

Agradecimientos

Agradezco a Dios y a mi familia por la ayuda que me han brindado, sin ellos no podría salir de mi País a estudiar la carrera de especialización.

A mis profesores y a mis amigos por el conocimiento y el apoyo que me han transmitido durante este proceso.

Índice de figuras

Índice de tablas

Prólogo

La idea de que la criptografía es aplicable sólo en el área militar o para mensajes entre personas de altos cargos políticos, quedó hace mucho en el pasado. Cada día sin darnos cuenta en el mundo se utiliza criptografía. Hoy en día, es difícil imaginar una transacción en línea o la navegación por sitios web “seguros”, sin el uso de certificados digitales emitidos por una autoridad que legitime su valor.

Esta autoridad y sus certificados son válidos, gracias a la creación de un sistema de clave pública o PKI por sus siglas en inglés, el cual funciona en base a algoritmos criptográficos. Muchos de los cuales se encuentran en riesgo de obsolescencia.

Ese desuso, es un peligro que a menudo mediante una nota en un blog o por el boletín de algún instituto de seguridad informática, se puede sentir más cercano. Inmediatamente salta una pregunta a la mesa: ¿Por qué ocurre esto?, ¿Se puede solucionar?, ¿Qué se está haciendo al respecto? Estas dudas se pretenden contestar en las siguientes 50 páginas siguientes.

Se debe recalcar que la información en la que se basa este “Trabajo Final de Especialización”, es aquella disponible hasta el momento de su conclusión. Por lo que para el instante en que el lector la lea, puede que algún protocolo criptográfico cambie, o el turno del desuso llegue a su puerta.

Con esto no quiero desalentar al lector de continuar leyendo esta obra, pero pienso que es necesario aclarar este punto, dado el rápido avance de la tecnología.

Antes de concluir este prólogo, quiero explicar cuál fue el motivo de elegir este tema. La razón de ello es el interés de conocer como los algoritmos criptográficos que usamos hoy en día y que, como ya se habló en el primer párrafo son necesarios en las transacciones y en la navegación en internet, entre otras cosas, serán reemplazados por algoritmos más complejos, o alguna modificación de los actuales. Pienso además que es un tópico, del que se hablará por los próximos años, y que combina dos materias que a lo largo de mi vida han cautivado mi atención: Matemática e Informática.

Estimado lector, espero que le sea de utilidad el trabajo que se presenta a continuación.

Nómina de abreviaturas

CA	Autoridad de Confianza
CBC	Change Block Cipher
CRL	Certificate Revocation List
CSP	Conjugator Search Problem
CVP	Closest Vector Problem
DCP	Double Coset Problem
DLC	Discrete Logarithm Cryptography
DLP	Discrete Logarithm Problem
DP	Decomposition Problem
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
DTLS	Datagram Transport Layer Security
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie Hellman
ECDLP	Elliptic Curve Discrete Logarithm Problem
ECDSA	Elliptic Curve Digital Signature Algorithm
FFC	Finite Field Cryptography
GNU	GNU’s Not Unix
GPG	GNU Privacy Guard
GSDP	Generalized Symmetric Decomposition Problem
HFE	Hidden Fields Equations
HSP	Hidden Subgroup Problem
IETF	Internet Engineering Task Force
LD-OTS	Lamport-Diffie one-time signature scheme
LWE	Learning With Errores problem
MPKC	Multivariate Public Key Cryptosystem
MSS	Merkle Signature Scheme
NAC	Non-Associative Cryptography
NCC	Non-Commutative Cryptography

NIST	National Institute of Standards and Technology
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
QFT	Quantum Fourier Transform
RA	Registration Authority
RFC	Request for Comments
RSA	Rivest, Shamir y Adleman
SD	Symmetric Decomposition
SHA	Secure Hash Algorithm
SIVP	Shortest Independent Vectors Problem
SVP	Shortest Vector Problem
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
ZKP	Zero Knowledge Protocol

CAPÍTULO 1: Criptografía asimétrica previa a la computación cuántica

1.1. Introducción y principales usos

La criptografía, es según el diccionario de la Real Academia de la Lengua Española el "Arte de escribir con clave secreta o de un modo enigmático". Para los estándares actuales, esta definición es muy general y no lo suficientemente precisa, esto debido a que pasó de ser un arte a ser una ciencia apoyada fuertemente en la matemática.

La criptografía era hasta hace algunos años, limitada a entornos militares o excepcionales casos civiles donde se requería de altos estándares de seguridad. A pesar de ello, se encuentra presente a lo largo de la historia, con ejemplos tan antiguos como el código de César, que se ilustra en la figura 1.1 o los jeroglíficos egipcios, descifrados mediante la piedra Rosetta. Fue con la llegada de la segunda guerra mundial que la criptografía aumentaría su complejidad, y la teoría de la información se convirtió en una parte indispensable de la misma. [1]

A partir del año 1976, se comenzaron a difundir de manera pública algoritmos criptográficos, que más tarde se clasificarían como asimétricos o de clave pública (en la sección 1.2 se analizará la diferencia entre algoritmos asimétricos y de clave pública).

Con la llegada del internet, se introdujo años después el comercio electrónico, el cual se vio obstaculizado en un inicio por la desconfianza de los consumidores en colocar información personal en un entorno donde no se creía posible asegurar los datos. La criptografía de clave pública fue entonces una herramienta para el desarrollo de esa incipiente seguridad cibernética.

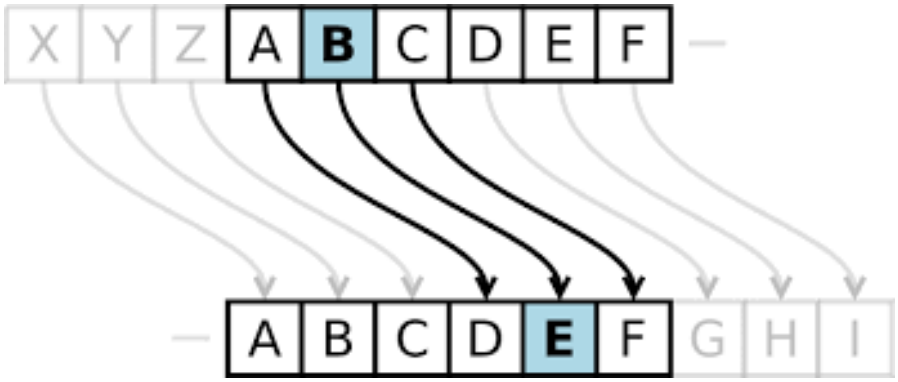

Figura 1.1: Ejemplo del código César. [2]

Actualmente se puede diferenciar tres tipos de criptografía:

- Aquella en que no se utiliza ningún parámetro secreto, ejemplo las funciones de hash, y generadores aleatorios de bits. Generalmente son utilizados como parte de criptosistemas. [1]

- Los algoritmos de clave secreta, los cuales utilizan información que se distribuyen entre las entidades participantes, y sólo es conocida por las mismas. [1]
- Los algoritmos de clave pública, los cuales poseen información compartida entre las partes involucradas y otra exclusiva de cada entidad participante. [1]

La criptografía de clave pública radica en la existencia de dos claves. Una que se encuentra segura a disposición de quién lo solicite, y otra de conocimiento y uso exclusivo de quien es representado por ella.

Hoy en día los principales usos de este tipo de criptografía son los siguientes:

- Encriptación de mensajes.
- Firma digital

El primero de ellos debe entenderse como un método para enviar un mensaje, en el cual existe una clave que está a disposición del público para cifrarlo, y otra diferente y a la vez secreta para descifrarlo. [3]

Con la llegada del primer algoritmo asimétrico, se establecieron las bases a cumplir por los algoritmos de este mismo tipo que se desarrollaron consecuentemente; en los cuales primaron las siguientes condiciones:

- Debe ser computacionalmente fácil generar las claves: pública y privada para cada entidad. [4]
- Debe ser computacionalmente fácil para un emisor, saber la clave pública del receptor y el mensaje a cifrar. [4]
- Deber ser computacionalmente fácil para el receptor descifrar el texto encriptado resultante utilizando su llave privada. [4]
- Debe ser computacionalmente inviable para cualquier persona que conoce la clave pública determinar la clave privada. [4]

Para entenderlo mejor, se plantea el siguiente ejemplo:

En la figura 1.2, se muestra un esquema básico del cifrado y descifrado de un mensaje. En él se puede observar que Rick (emisor) quiere enviarle un mensaje m a Elena (receptora), para ello Rick utiliza la clave pública de Elena f, y mediante alguno de los algoritmos de criptografía asimétrica existente, obtiene el mensaje encriptado $E_f(m) = C$, el cual envía a Elena. Una vez recibido el mensaje C, lo descifra utilizando su clave privada l, y obtiene el mensaje original $D_l(C) = m$.

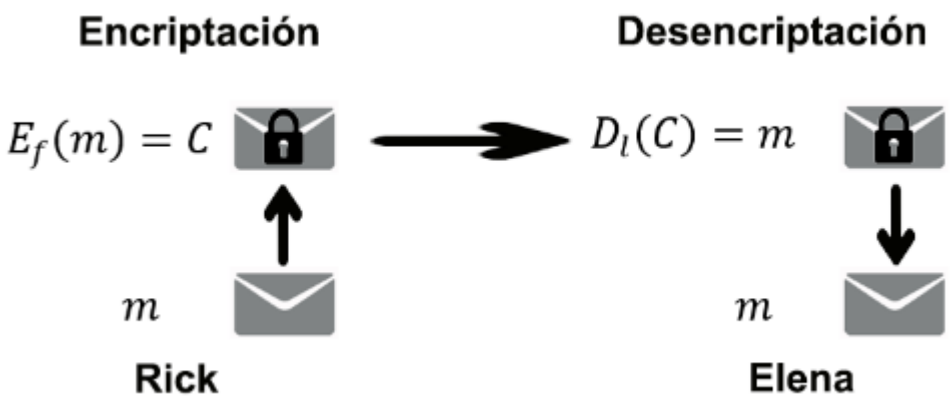

Figura 1.2: Esquema de un sistema criptográfico de clave pública para envío de mensajes.

La firma digital es un análogo informático a una huella de cera, un sello o una firma manuscrita, consiste en un código que se adjunta a un mensaje con la finalidad de que el receptor del mensaje pueda avalar quien le envió el mensaje. Debido a que en el mundo digital, es sencillo cortar y pegar un grupo de bits, se necesita que la firma digital dependa de por lo menos los siguientes elementos: [5] [6]

- El contenido del mensaje.
- La clave privada del autor.

Por ello, la firma es única para cada mensaje y autor, lo que permite establecer los siguientes parámetros para la misma:

- Autenticación: la firma digital facilita establecer el origen genuino de los mensajes, debido a que está firmado con información que sólo el remitente conoce, en este caso la clave privada. [6]
- Integridad: dada la dependencia de la firma al mensaje, cualquier alteración del mismo conllevaría forzosamente, a un cambio en la firma asociada al mensaje. Al momento, no existe un mecanismo para cambiar el mensaje y la firma sin conocer la clave privada. [6]
- No repudio: a razón de que la firma dependa de un secreto conocido únicamente por su remitente, el mismo no puede negar la firma de un mensaje. [6]

El siguiente ejemplo ilustra el funcionamiento de la firma digital.

En la figura 1.3, se puede observar que Rick quiere enviarle un mensaje m con su firma a Elena, para ello Rick realizará un hash del mensaje $H(m) = m'$. Ese resultado m' lo encriptará con su clave privada $E_r(m') = F$, lo cual resulta en la firma digital de Rick del mensaje F. Finalmente se juntan la firma, la clave pública de Rick y el mensaje m, para enviarlos a Elena.

Para verificar que el mensaje está firmado por Rick, Elena separa el mensaje original de la firma, y realiza un hash del mensaje $H(m) = m'$. Luego

desencripta la firma con la clave pública de Rick $D_c(F) = m'$ y obtiene el hash del mensaje. Compara ambos y verifica si Rick firmó o no el mensaje.

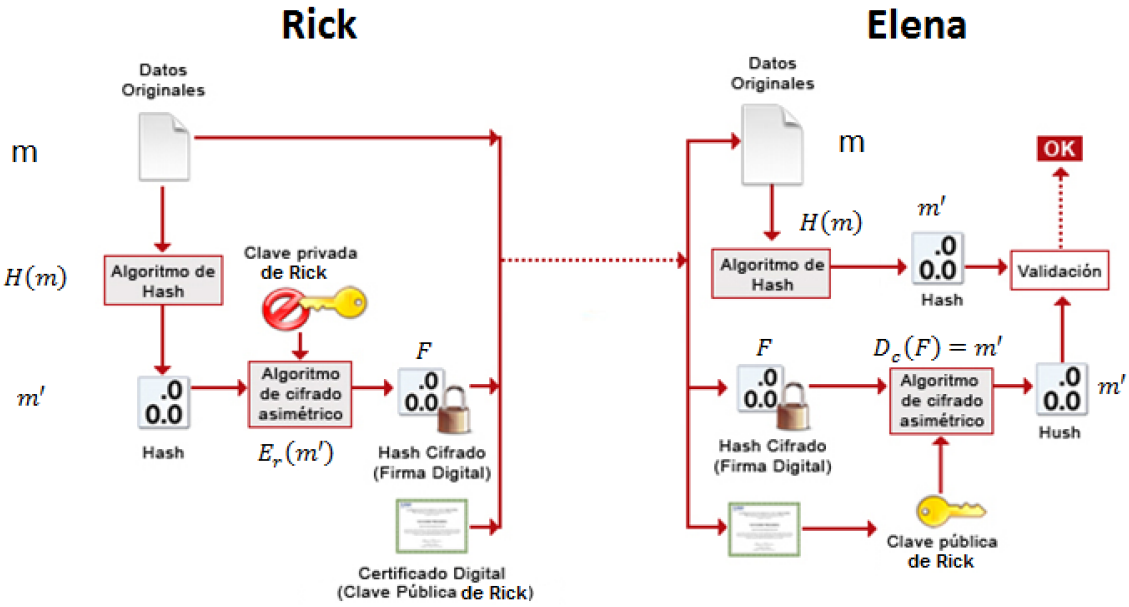

Figura 1.3: Esquema del envío de un mensaje firmado digitalmente y su comprobación. [7]

Para realizar correctamente el procedimiento descrito en la figura 1.3, es necesario que tanto Rick como Elena usen la misma función de hash.

1.2. Algoritmos de criptografía asimétrica

Hasta el momento, cuando se habla de criptografía asimétrica, se asocia directamente con la criptografía de clave pública. Y aunque en la actualidad los principales algoritmos asimétricos están catalogados como de clave pública, no todos los algoritmos existentes, ofrecen información conocida. [3]

A continuación, la figura 1.4, muestra en orden cronológico algunos de los más importantes algoritmos criptográficos asimétricos. Comenzando con el protocolo de intercambio seguro de claves Diffie-Hellman.

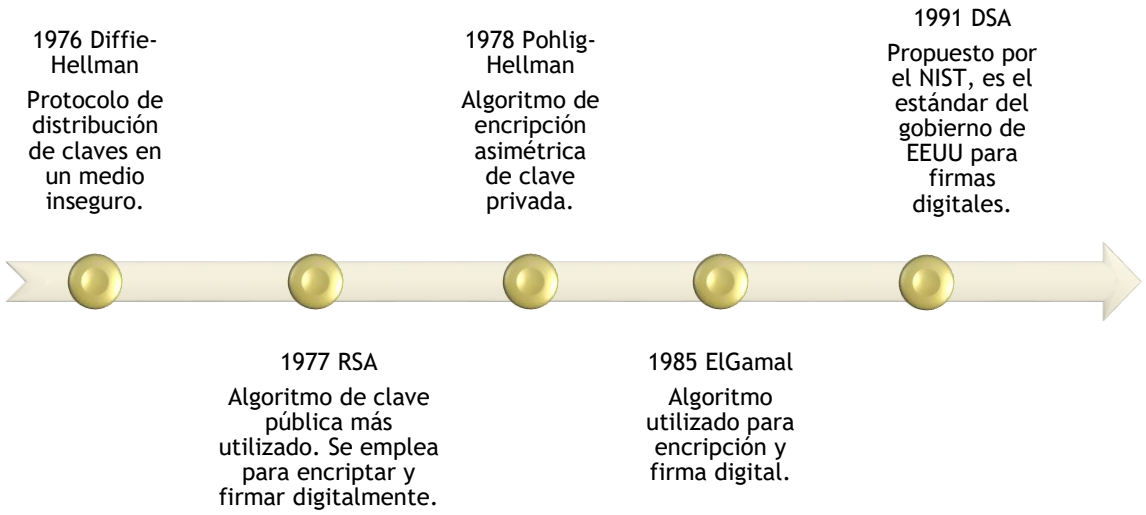

Figura 1.4: Algunos de los más importantes algoritmos criptográficos de clave pública.

1.2.1. Diffie-Hellman

Es el más antiguo algoritmo de clave pública, se usa principalmente para el intercambio seguro de claves en un medio no seguro. Además de ello, existen variantes usadas para la encriptación, como es el caso de ElGamal.

1.2.1.1. Distribución de claves Diffie-Hellman

En la figura 1.5, se explica el funcionamiento de este algoritmo. Donde Ariana y Bryan acuerdan compartir (información pública) un grupo finito *G,* a los que pertenecen una raíz generadora *g* y un número primo *p.* La información privada son los números que seleccionaron secretamente Ariana y Bryan, *a* y *b* respectivamente. [3]

Ariana calcula un número *A,* con la siguiente fórmula: $A = g^a mod(p)$, y se lo envía a Bryan. El cual calcula $B = g^b mod(p)$, y lo envía a Ariana. Entonces Ariana calcula la clave secreta compartida realizando: $k = B^a mod(p)$ y Bryan la encuentra realizando $k = A^b mod(p)$. [3]

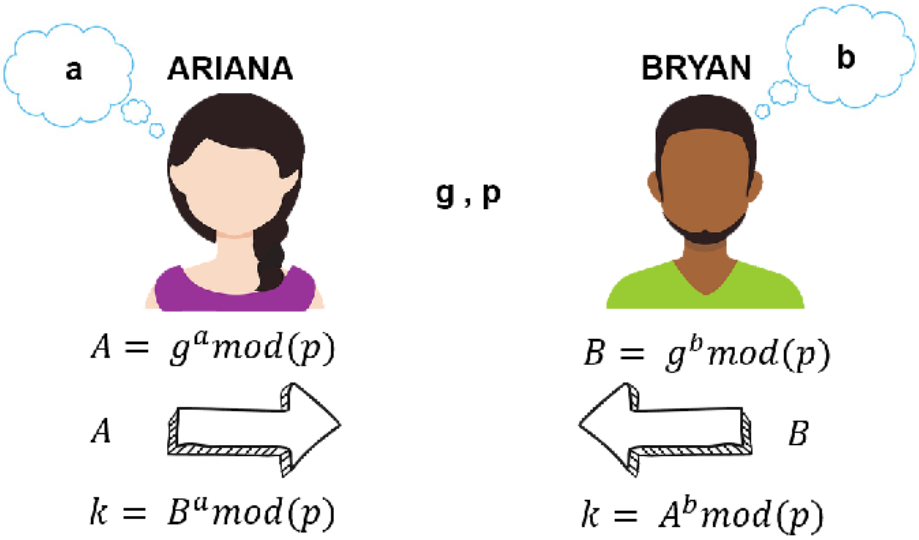

Figura 1.5: Ejemplo del funcionamiento del algoritmo Diffie-Hellman. [3]

1.2.1.2. Seguridad en Diffie-Hellman

Por el momento, es considerado un algoritmo seguro. Esta afirmación es consecuencia del "*Problema Diffie-Hellman",* el cual también asegura la fortaleza del protocolo de encriptación ElGamal, esta seguridad a su vez se basa en el problema del logaritmo discreto.

El problema Diffie-Hellman es el siguiente: dado un primo *p,* el generador *g* del conjunto $\mathbb{Z}_p^*$, los elementos $A = g^a mod(p)$ y $B = g^b mod(p)$, encontrar la clave secreta $k = g^{ab} mod(p)$. Para ello el orden del grupo finito *G* debe ser suficientemente grande para que un atacante no pueda encontrar por fuerza bruta los exponentes *a* ó *b,* actualmente 2048 bits. [8]

Si se lograra resolver el problema del logaritmo discreto (DLP) en *A* ó *B,* se podría encontrar alguno de los exponentes *a* ó *b* y encontrar la clave secreta mediante:

$$k = (g^a)^b mod(p)$$

1.2.2. RSA

Este algoritmo, toma su nombre de sus creadores Rivest, Shamir y Adleman, es el más utilizado y fue durante muchos años, el estándar internacional de facto tanto para firma digital como para encriptación segura de datos.

Como se indicó en la introducción, para que A y B intercambien mensajes o firmar mensajes, cada parte que debe crear una clave pública y otra privada, para ello se deben seguir los siguientes pasos:

1. Se generan dos números aleatorios diferentes, estos deben ser primos. En este caso *p* y *q*.
2. Se obtiene: $n = p * q$ y $\emptyset = (p-1)(q-1)$.

3. Se escoge un número aleatorio *e,* tal que:

$$1 \leq e \leq p - 2 \text{ y } MCD(e, \emptyset) = 1.$$

4. Se calcula un número *d,* que cumpla lo siguiente

$$1 \leq d \leq \emptyset \text{ y } e * d \equiv 1 \ (mod \ \emptyset),$$

Para ello se puede utilizar el algoritmo extendido de Euclides.

Resultando (n, e) como clave pública y (n, d) la privada o viceversa.

1.2.2.1. Encriptación RSA

Si A quiere encriptar un mensaje *M* para que B lo lea, primero debe obtener las claves públicas del destinatario (n, e).

El mensaje a cifrar *M,* se fracciona bloques de manera que:

$$1 \leq M \leq n - 1$$

Para calcular el mensaje encriptado *C* se realiza:

$$C = M^e mod(n)$$

Este mensaje *C* se envía al receptor, el cual lo desencripta a texto plano, con su clave secreta *d*:

$$M = C^d mod(n)$$

1.2.2.2. Firma digital RSA

Debido a que el algoritmo RSA fue considerado cerca de 15 años el estándar de facto en la criptografía asimétrica, existen algunas variantes de la firma digital RSA. La versión que se muestra a continuación es aquella presentada comúnmente en los libros de texto. [9]

Al igual que en la encriptación, se obtienen las claves públicas (n, e) y privadas (d) para las partes, o al menos las que pertenecerán al firmante del mensaje. Se comienza fraccionando el mensaje como se había visto antes, la resultante fracción *M,* será usada como entrada de una función de una sola vía, habitualmente un hash *H.*

$$\tilde{m} = H(M)$$

Ese resultado $\tilde{m}$ y la clave privada del firmante, serán utilizados en el siguiente cálculo:

$$s = \tilde{m}^d mod(n)$$

Finalmente, el firmante envía el mensaje en texto plano (en algunos casos se encripta este mensaje utilizando un algoritmo simétrico), junto con *s* que es la firma única para el mensaje.

Para verificar que el remitente del mensaje fue quien lo firmó, se realizan dos operaciones por separado.

Utilizando el mensaje original se aplica la misma función de una sola vía, para obtener el resultado $\tilde{m}$. Mientras que al resultado *s,* se le aplicará la siguiente operación:

$$\check{m} = s^e mod(n)$$

Si los resultados $\tilde{m} = \check{m}$ son iguales, podemos confirmar la autenticidad de la firma y con ella el mensaje recibido.

1.2.2.3. Seguridad en RSA

La fortaleza de este algoritmo, reside fundamentalmente en dos problemas:

- La dificultad de factorizar enteros.
- El problema RSA.

Ambos ítems dan seguridad tanto a la encriptación como a la firma digital. El primero de ellos trata sobre la obtención de los primos *p* y *q*, a partir de la factorización de *n.* En esta dificultad reside la mayor parte de la fortaleza de RSA contra posibles atacantes. Suponiendo que un adversario logre encontrar los primos *p* y *q*, sería tan sólo cuestión de resolver la siguiente ecuación para encontrar la clave privada de la víctima.

$$d \equiv e^{-1} \ (mod \ ((p-1)(q-1)))$$

Es importante entonces, que la longitud de los valores *p* y *q,* sea por lo menos 1024 bits. [10]

Mientras que el problema RSA, corresponde a la posibilidad de obtener el mensaje en texto plano *M*, sin la necesidad de conocer la clave privada *d,* a partir de la clave pública *e* y el mensaje cifrado *C* como información conocida. [11]

Para ello el atacante debe plantear la siguiente ecuación:

$$C = M^e mod(n)$$

Dadas las condiciones del algoritmo RSA, se conoce que para todo *C* que se encuentra en el rango: $1 \leq C \leq n-1$, corresponde un único mensaje *M* dentro del siguiente intervalo: $1 \leq M \leq n-1$. [11]

A pesar de que resolver este problema es más sencillo que factorizar *n,* se considera que si *n* es lo suficientemente grande (mínimo 2048 bits), calculado con un generador que garantice su aleatoriedad, y M se encuentra entre *1* y *n-1.* La dificultad de resolución no es polinomial, el requisito de que el intervalo de *M* sea amplio*,* evita que un posible atacante encuentre el mensaje probando todos los valores posibles para *M*. [11]

1.2.3. Pohlig-Hellman

El algoritmo Pohlig-Hellman ó Silver-Pohlig-Hellman, es conocido como el primer asimétrico que no ofrece información pública, el cual fue desarrollado y presentado por los dos primeros, e independientemente por Silver. En la

actualidad su uso como algoritmo de encriptación o en el caso de firma difgital, es nulo.

1.2.3.1. Encriptación Pohlig-Hellman

A continuación se explica la utilización del algoritmo en la criptografía: [12]

Se escoge un *p* que es primo grande, y *k* la clave para la encriptación, tal que cumpla lo siguiente:

$$1 \leq k \leq p-2$$

$$MCD(k, p-1) = 1$$

Ahora, el mensaje a cifrar *M,* lo fraccionaremos en bloques de manera que:

$$1 \leq M \leq p-1$$

Entonces el mensaje encriptado *C* es igual a:

$$C = M^k mod(p)$$

Y el receptor del mensaje deberá desencriptar a texto plano, con una clave secreta *D*:

$$D = k^{-1} mod(p-1)$$

$$M = C^D mod(p)$$

La seguridad de este algoritmo radica en mantener en secreto las claves *k* y *D*, además de resolver la siguiente ecuación:

$$k = log_M C$$

1.2.4. ElGamal

Este algoritmo desarrollado por el criptógrafo egipcio Taher Elgamal, está basado en el protocolo de establecimiento de claves Diffie-Hellman, y con él su seguridad.

Es de uso libre, es decir, no está patentado y es utilizado tanto para la encriptación/desencriptación como para firma digital. Entre las aplicaciones prácticas actuales están: el protocolo GNU Privacy Guard, PGP, entre otros. [13]

Para crear las claves pública y privada, cada parte debe realizar los siguientes pasos: [14]

1. Generar un número primo aleatorio *p* y un generador *g* que pertenece al grupo multiplicativo $\mathbb{Z}_p^*$.
2. Elegir un entero aleatorio *a,* el cual debe estar en el rango: $1 \leq a \leq p-2$.

3. Calcular $c = g^a mod(p)$.

La clave pública es (*p, g, c*), y la privada *a.* Ahora que las partes tienen sus claves, pueden encriptar y desencriptar mensajes.

1.2.4.1. Encriptación ElGamal

Ahora que las partes tienen sus claves, pueden encriptar y desencriptar mensajes. En el caso de la encriptación, el algoritmo a seguir es el siguiente:

1. Conseguir la clave pública del destinatario (*p, g, c*)
2. Fraccionar el mensaje *m* dentro del intervalo, $0 \leq a \leq p - 1$.
3. Seleccionar un entero aleatorio *k,* $1 \leq k \leq p - 2$.
4. Determinar $\gamma = g^k mod(p)$ y $\delta = m.(c^k) mod(p)$.
5. Enviar el texto cifrado $C = (\gamma, \delta)$ al destinatario.

Para que el destinatario desencripte el mensaje debe calcular $(\gamma^{-a}).\delta\ mod(p)$. La razón de esto se explica de la siguiente forma: [14]

$$(\gamma^{-a}).\delta\ mod(p) = (g^k)^{-a}.m.(c^k) mod(p) =$$

$$(g^k)^{-a}.m.(g^a)^k\ mod(p) = (g^a)^{-k}.(g^a)^k.m\ mod(p).$$

1.2.4.2. Firma Digital ElGamal

Para realizar la firma digital se necesita que cada parte posea su clave pública (*p, g, c*) y privada *a*, estas se pueden obtener con el mismo procedimiento que se explicó en la encriptación.

La entidad firmante deberá seguir el siguiente procedimiento:

1. Elegir un entero aleatorio *k*, $1 \leq k \leq p - 2$. Tal que el MCD$(k, p - 1) = 1$.
2. Calcular $\gamma = g^k mod(p)$.
3. Utilizando el resultado anterior, se procede a encontrar $s = (k^{-1}(H(m) - a\gamma)) mod(p - 1)$.

Finalmente, la firma del mensaje *(γ, s)* será enviada junto con el mensaje *m* al destinatario, el cual la verificará siguiendo estos pasos:

1. Verificar que γ se encuentra en el intervalo $1 \leq \gamma \leq p - 1$, caso contrario es falsa la firma.
2. Determinar $\grave{t} = c^{\gamma}\gamma^{s} mod(p)$.
3. Encontrar $H(m)$ y $\ddot{t} = g^{H(m)} mod(p)$.

Se valida que la firma se corresponda al mensaje y al firmante, sí y sólo sí, se cumple que $\grave{t} = \ddot{t}$.

1.2.4.3. Seguridad en ElGamal

Una vez vistos los procedimientos matemáticos involucrados en la encriptación y la firma ElGamal, corresponde la pregunta: ¿Qué seguridad existe de que no se altere el mensaje mediante métodos algebraicos, o que se logre falsificar una firma?

En el caso de la encriptación se considera las siguientes características para limitar los posibles atacantes:

- Problema de Diffie-Hellman.
- La no reutilización del entero *k.*

Recuperar el mensaje *m,* a partir de la información pública $(p, g, c, \gamma, \delta)$ es equivalente a resolver el problema Diffie-Hellman aplicado a determinar g^{ak}, por ello como se mencionó en el inciso 1.2.4, la seguridad de ElGamal está basada en la seguridad de Diffie-Hellman.

El uso del entero *k* en la encriptación de más de un mensaje permite que mediante métodos algebraicos un posible atacante, pueda conocer el contenido de un mensaje, si ya conoce otro encriptado con el mismo valor *k.* Un ejemplo de esto es el siguiente:

Sean m_1 y m_2 mensajes encriptados con el mismo valor de entero *k*, donde se conoce los mensajes encriptados (γ_1, δ_1) y (γ_2, δ_2), es posible concluir la siguiente relación $\frac{\delta_1}{\delta_2} = \frac{m_1}{m_2}$, por lo que conociendo uno de los dos mensajes, se puede encontrar el otro.

Debido a que la firma digital comprende mayor complejidad en su algoritmo, posee mayor cantidad de requisitos de seguridad.

Para ello existen por lo menos 3 seguridades que posee la firma ElGamal a considerar:

- El problema del logaritmo discreto.
- El número aleatorio *k* para cada mensaje.
- La función de una sola vía, habitualmente un hash.

Un atacante puede pretender falsificar una firma en un mensaje *m,* seleccionando un entero aleatorio *k* y calculando $\gamma = g^k mod(p)$, además debe encontrar $s = (k^{-1}(H(m) - a\gamma))mod(p-1)$. Debido a la imposibilidad de resolver el problema del logaritmo discreto, lo único que el atacante puede hacer es elegir un número al azar *s,* la posibilidad de que ese número sea el correcto es $\frac{1}{p}$, y debido a que p es número de un orden de 1024 bits o mayor, la probabilidad de éxito es prácticamente nula.

Al igual que en la encriptación, el caso de seleccionar una misma *k* para la firma de más de un mensaje genera una vulnerabilidad grave, la cual puede ser aprovechada por un posible atacante de la siguiente manera:

Para el mensaje m_1, se sabe que $s_1 = (k^{-1}(H(m_1) - a\gamma))mod(p-1)$ y para el m_2, $s_2 = (k^{-1}(H(m_2) - a\gamma))mod(p-1)$ se restan $s_1 - s_2$,

$$s_1 - s_2 = \left(k^{-1}(H(m_1) - a\gamma)\right)mod(p-1) - (k^{-1}(H(m_2) - a\gamma))mod(p-1)$$

$$s_1 - s_2 = k^{-1}((H(m_1) - a\gamma)mod(p-1) - (H(m_2) - a\gamma))mod(p-1))$$

$$k(s_1 - s_2) = (H(m_1) - H(m_2))mod(p-1)$$

Si $k(s_1 - s_2) \neq (0)mod(p-1)$, entonces:

$$k = (s_1 - s_2)^{-1}(H(m_1) - H(m_2))mod(p-1)$$

Una vez encontrado *k,* es posible encontrar *a.* Por ello es imperativo usar un valor *k* diferente para cada mensaje a cifrar.

El uso de la función hash es otra herramienta importante en la seguridad de la firma, sin ella el valor *s,* se calcularía $s = (k^{-1}(m - a\gamma))mod(p-1)$. Por lo que un posible atacante puede elige un par de valores (u, v), siempre y cuando $MCD(v, p-1) = 1$.

Luego calcular $\gamma = (g^u c^v)mod(p) = g^{u+av}mod(p)$ y $s = -\gamma v^{-1}mod(p-1)$. El par (γ, s) es una firma válida para el mensaje $m = (su)mod(p-1)$, debido a que $(g^m g^{-a\gamma})^{s^{-1}} = g^m c^{-a\gamma} = \gamma$.

1.2.5. DSA

En el año 1991 fue concebido por el Instituto Nacional de Estándares y Tecnología (NIST), ante la necesidad del gobierno de Estados Unidos de poseer un algoritmo que sirva como estándar de firma digital. Es una variante de ElGamal (visto en el punto 1.2.4), además de ser una de las tres técnicas aprobadas dentro del Estándar Federal de Procesamiento de Información (FIPS 186). Donde se especifica los procedimientos para la generación de parámetros de dominio, pares de claves públicas y privadas, así como la generación y verificación de firmas digitales. [10]

Al igual que en el resto de los algoritmos asimétricos, antes de firmar un mensaje se deben generar las claves públicas y privadas de las entidades participantes.

Para ello cada firmante debe completar el siguiente procedimiento:

1. Generar un número primo aleatorio *p,* cuyo valor debe estar en el rango $2^{L-1} \leq p \leq 2^L$. Donde *L* es el número de bits de longitud de *p,* se aconseja un mínimo de 1024 bits.
2. Seleccionar un primo *q,* el cual estará limitado por el rango $2^{N-1} \leq q \leq 2^N$. El valor *N* representa el número de bits de longitud de *q.*

3. Seleccionar un número *g,* que sea generador de al grupo multiplicativo $\mathbb{Z}_p^*$. Por lo tanto, para cualquier $\beta \in \mathbb{Z}_p^*$, se cumple que $1 \neq (\beta^{(p-1)/q}) mod(p)$.
4. De manera aleatoria se escoge una clave aleatoria *x,* cuya magnitud se encontrará entre $1 \leq x \leq q-1$.
5. Finalmente se encontrará *y,* a partir de resolver $y = (g^x) mod(p)$.

Una vez que se complete este proceso, la clave pública para el firmante será (p, q, g, y)*;* mientras que la privada *x.*

Según el estándar de firma digital (FIPS 186-4) los valores de *L* y *N* pueden ser respectivamente: [10]

- L = 1024, N=160.
- L= 2048, N=224.
- L= 2048, N=256.
- L= 3072, N=256.

1.2.5.1. Firma Digital DSA

El proceso criptográfico involucrado en DSA, posee como se ha mencionado similitud con el algoritmo de firma ElGamal, a continuación se muestra los pasos a seguir para la generación de la misma: [10]

1. Elegir un entero aleatorio *k*, donde k^{-1} es la inversa multiplicativa de *k* con respecto al módulo *q,* $0 < k^{-1} < q$.
2. Encontrar $\gamma = (g^k mod(p)) mod(q)$.
3. Utilizando el resultado γ, se procede a calcular $s = (k^{-1}(H(m) + a\gamma)) mod(q)$.

Completados los pasos anteriores, se reconoce que la firma generada es *(γ, s),* la cual debe ser enviada junto con el mensaje *m* al destinatario. Para confirmar la autenticidad e integridad, el receptor debe verificarla siguiendo estos pasos: [10]

1. Obtener la clave pública del firmante (p, q, g, y).
2. Corroborar que (γ, s) se encuentra en el intervalo $0 < \gamma < q$ y $0 < s < q$, caso contrario es falsa la firma.
3. Determinar $\grave{t} = s^{-1} mod(p)$.
4. Encontrar $H(m)$.
5. Computar $u_1 = (\grave{t} H(m)) mod(q)$ y $u_2 = (\grave{t}\gamma) mod(q)$.
6. Calcular $\psi = ((g^{u_1} y^{u_2}) mod(p)) mod(q)$.

Se valida que la firma se corresponda al mensaje y al firmante, sí y sólo sí, se cumple que $\psi = \gamma$. [10]

1.2.5.2. Seguridad en DSA

Debido a que DSA, es un algoritmo estándar del gobierno de los Estados Unidos, su seguridad debe permitir que se lo identifique como "inquebrantable". Esta confianza se basa en dos problemas de logaritmos discretos distintos pero relacionados.

Uno es el problema de la imposibilidad de resolver el logaritmo en el conjunto $\mathbb{Z}^*$. El otro es el problema de logaritmo en el subgrupo cíclico de orden q.

Como se ha mencionada DSA, se basa en el algoritmo ElGamal, esto explica que algunos puntos de su seguridad se pueda definir también de aquellos que afectan a ElGamal (1.2.4.3), entre ellos la necesidad de un valor aleatorio k diferente para cada mensaje firmado, o los ítems asociados a la ecuación s.

1.2.6. ECDSA

Hasta el momento los algoritmos enumerados a excepción de Diffie-Hellman comparten una característica común, son criptografía de campo finito (FFC), pero dentro de la criptografía logarítmica discreta (DLC), existe otra rama llamada criptografía de curva elíptica (ECC). [15]

ECC, como su nombre lo indica se basa en curvas elípticas, las cuales están formadas por un conjunto de puntos que satisfacen una ecuación cúbica de la forma:

$$y^2 = x^3 + ax + b$$

Para el caso de ECDSA, que significa algoritmo de firma digital de curva elíptica, las partes que intervienen acuerdan lo siguiente: [10]

1. Definir *E* una curva elíptica sobre el campo F_q.
2. Seleccionar un punto *P* de orden *n* en la curva *E.*

Cabe destacar que el campo F_q, es un campo cuyo orden es finito. Una vez acordado lo anterior entre las partes involucradas, la selección de las claves pública y privada, se realizará mediante el siguiente par de pasos: [10] [15]

1. Estocásticamente se selecciona un entero *d,* el cual debe ser un primo del orden de 2^n bits.
2. Se calcula $Q = dP$.

Concluido lo anterior, la clave privada es *d* y la pública es *Q.*

1.2.6.1. Firma Digital ECDSA

Este proceso guarda cierta similitud con la firma digital DSA vista en el punto 1.2.5.1, a pesar de ello el hecho de que se calcula sobre la curva elíptica *E,* fundamenta que no sea el mismo proceso, como se explica a continuación:

1. Estocásticamente se escoge un entero *k* en el intervalo $[1, n-1]$.
2. Se encuentran las coordenadas (x_1, y_1) calculando $kP = (x_1, y_1)$.
3. Se computa $r = (x_1) mod(n)$.
4. Se encuentra $e = H(M)$, *H* es una función tipo Hash.
5. Con el resultado anterior se calcula $s = (k^{-1}(e + dr)) mod(n)$.

Finalizados estos pasos el remitente puede enviar el mensaje *M,* junto con la firma (r, s). El receptor deberá verificar entonces que la firma se corresponda con el mensaje, con ello validará la autenticidad de lo recibido. [15] [10]

1. Se calcula $e = H(M)$.
2. Se obtiene los valores u_1 y u_2, mediante $u_1 = (es^{-1}) mod(n)$ y $u_2 = (rs^{-1}) mod(n)$.
3. Se encuentra el punto $(x_1, y_1) = u_1 P + u_2 Q$.
4. Se computa $v = (x_1) mod(n)$.

Si el receptor puede validar que $v = r$, entonces podrá determinar que es auténtica la firma.

1.2.6.2. Seguridad ECDSA

Debido a que ECDSA forma parte de otra rama de la criptografía de campo finito (FFC), las propiedades que aseguran su confianza no son las mismas que en los otros algoritmos, aunque comparten cierta similitud, las principales seguridades son las siguientes: [10] [15]

- El problema del logaritmo discreto en la curva elíptica (ECDLP).
- Orden de la clave *d.*

En el primer problema, se debe tomar las siguientes consideraciones:

1. Se define *E* una curva elíptica sobre el campo F_p.
2. Se escoge un punto *P* de orden *n* en la curva *E.*
3. El orden *n* es un entero de por lo menos de 160.

El problema consiste en determinar un entero *u,* tal que $0 \leq u \leq n-1$ y $Q = uP$. En el segundo caso, la longitud de *n* debe ser mínimamente mayor a 2^{160}.

1.3. Principales aplicaciones de la criptografía de clave pública.

Todo el trabajo matemático de crear y mejorar los algoritmos de clave pública, se ve aplicado en distintos protocolos de uso diario; a tal punto que algunos gobiernos han delineado las características específicas que deben tener y sus variables para ser reconocidos como seguros.

Un ejemplo de esto es el Estándar de Firma Digital (DSS), el cual es definido por el Instituto Nacional de Estándares y Tecnología (NIST) que pertenece al gobierno de los Estados Unidos, este menciona varios algoritmos vistos antes: DSA, RSA y ECDSA.

Actualmente en su cuarta versión, para cada uno de los antes mencionados especifica: cómo se deben generar los pares de claves, la firma y su verificación. Este estándar es de gran relevancia para el mundo criptográfico, no sólo por el nivel de detalle y el autor, también es utilizado en otros protocolos como: Privacidad Bastante Buena (PGP) y Seguridad de la capa de transporte (TLS).

PGP es un protocolo desarrollado hace más de 25 años, por Phil Zimmermann para asegurar comunicaciones seguras entre partes, su uso principal se encuentra en el correo electrónico. Su importancia radica en su antigüedad y la variedad de plataformas en que se ha implementado.

Fue desarrollado como un programa gratuito, capaz de proporcionar cifrado y autenticación en el correo electrónico y almacenamiento seguro de archivos. Debido al momento en que se publicó tuvo obstáculos legales con RSA, debido a que el algoritmo se utiliza para el transporte de claves. Otras versiones de PGP usan Diffie-Hellman para la gestión de claves y DSS para las firmas digitales.

Actualmente se encuentra disponible como complemento para aplicaciones como Apple Mail, Outlook ó Thunderbird. Así como su versión de software libre GNU Privacy Guard (GPG) la cuál es compatible con OpenPGP, que se encuentra descrito en RFC 4880. [16]

TLS es un protocolo de seguridad en la capa de transporte, cuya primera versión fue definida en 1999 por el Grupo de Trabajo de Ingeniería de Internet (IETF). En su primera versión (TLS v1.0) descrita en el RFC 2246, utiliza 3-DES para la seguridad de clave secreta, Diffie-Hellman para el intercambio de clave y DSS en la firma digital.

En el año 2002 se describió teóricamente una vulnerabilidad para TLS v1.0 en el encadenamiento de bloques de cifrado (CBC), nueve años más tarde se desarrolló un exploit para aprovecharla. Antes de ello en el año 2006 en el RFC 4346 se especificó la siguiente versión de TLS (TLS v1.1), la cual suma protección contra ataques de CBC.

En el año 2008 se lanzó TLS v1.2 en el RFC 5246, en el cual aumenta las seguridades criptográficas con respecto a TLS v1.1, actualmente se considera “seguro” el protocolo TLSv1.2 con AES 256 bits. Debido a que TLS se encuentra definido para el protocolo de transporte TCP, el IETF lanzó una variante para UDP llamada Datagram Transport Layer Security (DTLS), cuya última versión (DTLS v1.2) se encuentra descrita en el RFC 6347. [16]

Para finalizar este punto y el capítulo, se debe mencionar a la infraestructura de clave pública (PKI). Este sistema permite el intercambio seguro de datos a

través de redes públicas o privadas, comprobando la identidad de los participantes.

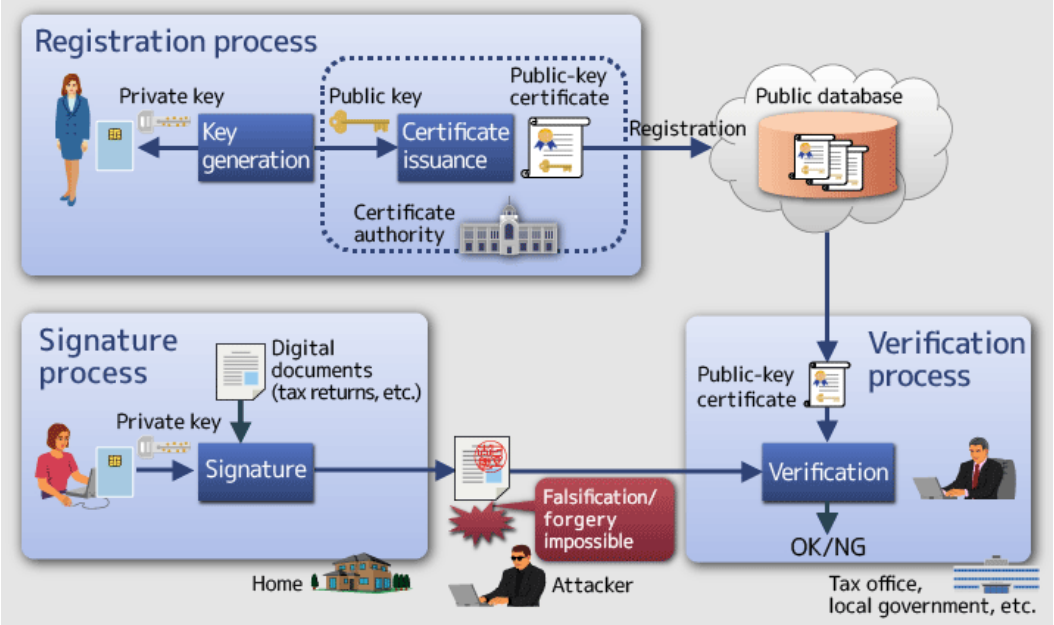

Figura 1.6: Estructura de una PKI. [17]

En la figura 1.5 se muestra la estructura de una PKI, para que algún miembro pueda enviar un mensaje firmado dentro la infraestructura, primero debe pasar por un proceso de registro, en el cual la Autoridad de Registro (RA) o también conocida como subCA procesa la solicitud de registro ante la Autoridad Certificante (CA), en caso de ser aprobada se crea un certificado para el individuo y se almacenan en una base datos certificada, si el usuario hace un mal uso, o la seguridad del certificado es comprometida se envía a la Lista de Revocación de Certificados (CRL).

Si un miembro del PKI recibe un mensaje firmado con el certificado de otro, y quiere verificar que es auténtico, debe enviar el certificado a la autoridad de validación de la infraestructura, para que esta la compare con los certificados de la base de datos certificada y determine su autenticidad. Debido a lo extenso de este tema, se recomienda consultar cualquier duda de PKI en algún libro de Criptografía.

CAPÍTULO 2: Computación cuántica en la criptografía asimétrica actual.

2.1. Computación cuántica

La mejora de la capacidad de procesamiento de los microprocesadores, llevó a Gordon Moore cofundador de Intel, a expresar en 1975: "*aproximadamente cada dos años se duplica el número de transistores en un microprocesador".* Esto se conoce como la ley de Moore, la cual se ha cumplido con el paso del tiempo. [18]

Para lograr este cometido, se ha disminuido el tamaño de los transistores a escalas cada vez más pequeñas, lamentablemente existe un límite a partir del cual ocurre el efecto túnel y no se puede continuar reduciendo. El efecto túnel viola los principios de la mecánica clásica y permite que electrones puedan penetrar una barrera de potencial mayor que la energía cinética que poseen. [19]

Buscando evitar este problema y continuar aumentando la capacidad de procesamiento en la computación, Paul Benioff en 1981 sugirió el aprovechamiento de las leyes cuánticas en la computación.

En la computación clásica una partícula tiene dos valores posibles: 0 y 1, mientras que en la mecánica cuántica puede ser: 0, 1 o la superposición cuántica de 0 y 1, este detalle permite aumentar la cantidad de procedimientos que se pueden llevar a cabo en simultáneo. Esta partícula se llama cúbit, y es la base de la computación cuántica.

Las memorias clásicas (actuales) con x bits poseen una dimensión de estado de x, pero un sistema x número de cúbit tiene una dimensión de estado de 2^x. A medida que más procedimientos se puedan llevar a cabo de manera paralela, mayor es la capacidad de cómputo, pero el tiempo para la ejecución es más o menos el mismo. Si los algoritmos usan las propiedades específicas de la mecánica cuántica, los sistemas cuánticos pueden superar a las computadoras clásicas, y lograr que problemas con dificultad actual de intratables se transformen en resolubles.

2.1.1. Transformada Cuántica de Fourier

Para entender los algoritmos que se mostrarán a continuación, es necesario comprender primero la transformada cuántica de Fourier, la cual es una parte importante en los algoritmos de Shor.

Esta transformada es una operación unitaria en un registro de cúbits con $n + 1$ cúbits, esto permite que $N = 2^n$ cantidad de estados sean posibles. Es aplicada en un estado base y descrito como:

$$F^{\otimes n}|x\rangle_n := \frac{1}{\sqrt{N}} \sum_{j=0}^{2^n-1} \exp\left(2\pi i \frac{jx}{N}\right)|j\rangle_n, x \in \{0, \dots, N.1\}$$

En la ecuación descrita, $F^{\otimes n}$ es una matriz unitaria de dimensiones 2^n por 2^n, con una parte imaginaria i. Según lo descrito por Shor, la aplicación de la transformada requiere una cantidad polinomial de pasos de tiempo en tamaño de bit n. La transformada es en esencia una matriz compleja unitaria y la matriz $F^{-1} = F^*$, en la cual F^* es la conjugada transpuesta de *F.* En esta operación la inversa es igual a la matriz original con un menos en el exponente de la función “exp”.

2.2. Algoritmos de computación cuántica que amenazan la criptografía asimétrica actual

2.2.1. Algoritmo de Shor para la factorización de enteros

En el año 1994 Peter Shor describió algoritmos cuánticos, para encontrar la factorización de cualquier número entero positivo N en sus factores primos y resolver el problema del logaritmo discreto.

Dado el efecto revolucionario que poseen los algoritmos de Shor en la criptografía asimétrica, varias investigaciones se han llevado a cabo para analizar y perfeccionar los costos exactos del algoritmo de Shor para la factorización de números enteros positivos; específicamente, en torno al número de cúbits necesarios y la cantidad de operaciones de cúbit requeridas.

La razón por la cual algoritmos criptográficos que se muestran en la tabla 2.1., como RSA y ECDSA se mantienen seguros, es porque, para vulnerar el nivel de seguridad que tienen hoy en día es necesario billones de operaciones en miles de cúbits lógicos. Al manejar una mayor cantidad de operaciones la tasa de error tiende a aumentar, por lo que ataques capaces de tolerar fallas eventualmente necesitaran billones de operaciones en millones de cúbits físicos. Actualmente no se ha identificado algún impedimento en que la computación cuántica pueda lograr esos niveles de operaciones, por lo que resulta prudente buscar protegerse ante la posibilidad que ataques de esa clase tengan éxito.

Algoritmo Criptográfico	Tipo	Uso	Impacto
AES	Clave Simétrica	Encriptación	Necesario claves más largas
SHA-2, SHA-3	----------	Funciones Hash	Necesario salidas más largas
ECDSA, ECDH	Clave Pública	Firma, Establecimiento de claves	No será seguro
DSA	Clave Pública	Firma, Establecimiento de claves	No será seguro

RSA	Clave Pública	Firma, Establecimiento de claves	No será seguro

Tabla 2.1: Impacto en algoritmos criptográficos vulnerables a ataques de computación cuántica. [20]

2.2.1.1. Factorización de enteros con el algoritmo de Shor

Todo número entero positivo, puede ser descompuesto en los valores primos que lo componen, es decir, un valor *N* puede ser descompuesto en *k* números primos $N = \prod_{i=0}^{k} p_i^{a_i}$. Para valores pequeños como 35 que puede ser descompuesto en: 5 y 7 no existe mayor dificultad, pero cuanto más grande es el valor, más difícil se torna el problema, de esto se aprovechan algunos algoritmos criptográficos como RSA (ver sección 1.2.2), en el cual se multiplican dos primos de orden de al menos 1024 bits, y a partir de allí se escoge un número y luego se calcula las claves públicas y privadas.

En la figura 2.1 se muestra el algoritmo de Shor, para comenzar se escoge un número aleatorio *x* menor que el número *N* a descomponer.

Se calcula el máximo común divisor entre ambos, $\gcd(N, x)$.

Si ese valor es mayor que 1, entonces *x* es un factor de *N*, debido a que ambos tienen términos en común y resta dividir *N* entre *x* para conocer el otro factor.

Si es igual a 1, entonces se calcula el menor valor de *r*, de la siguiente ecuación:

$$x^r \equiv (1) mod(N)$$

Si *r* es un número par, se puede reescribir la ecuación como:

$$x^{r}/_{2} \equiv (y) mod(N)\ , donde\ y^2 \equiv (1) mod(N)$$

Entonces:

$$y^2 - 1 \equiv (0) mod(N) \equiv (y-1)(y+1) \equiv (0) mod(N)$$

Si $1 < y < N - 1$ y $1 < y - 1 < y + 1 < N$, *N* no puede dividir $y - 1$ y $y + 1$ por separado, entonces se puede encontrar los factores que forman *N,* de la siguiente manera:

$p = \gcd(N, y - 1)$, $q = \gcd(N, y + 1)$

Si no se cumplen las condiciones:

- *r* es un número par
- $1 < y - 1 < y + 1 < N$

Se debe volver a tomar otro número aleatorio *x* menor que *N,* hasta conseguir los factores que forman *N,* como se muestra en la figura 2.1.

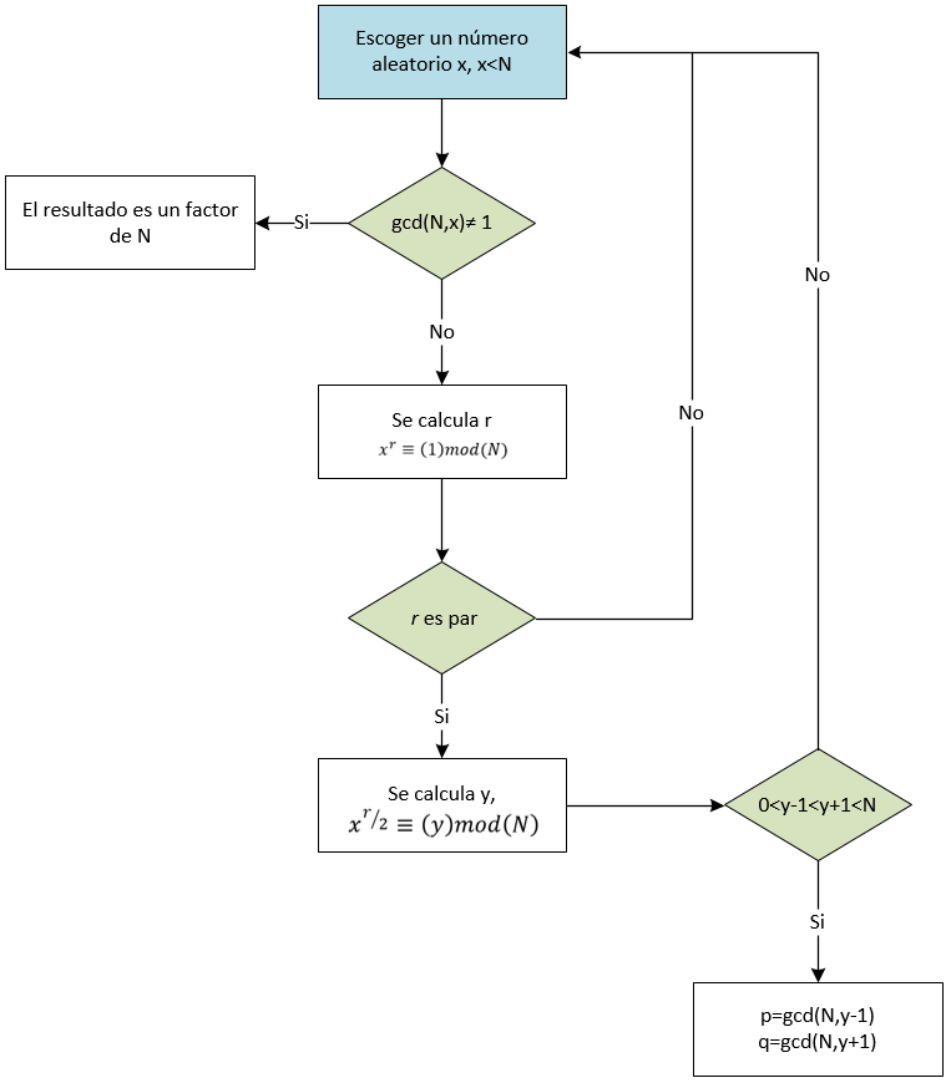

Figura 2.1: Algoritmo de Shor para factorizar un número *N.*

Para ejemplificar la aplicación de este algoritmo, se puede descomponer un valor $N = 21$ en sus factores, y se tomará al azar los siguientes valores:

1. $x_1 = 4$
2. $x_2 = 13$
3. $x_3 = 15$

En el primer caso se calcula el máximo común divisor entre ambos $\gcd(21,4) = 1$, debido a que el resultado es igual a 1, se busca el orden o periodo de x_1 en la siguiente ecuación:

$$4^r \equiv (1) mod(21)$$

Cuyo valor es 3, 6, 9,… debido a que el menor valor es impar se debe buscar otro número aleatorio.

Para comprobar que no es posible encontrar los factores de *N* a partir de $x_1 = 4$, se seleccionará $r = 6$ y se calculará y. Para lo cual se aplica la ecuación:

$$4^3 \equiv (y) mod(21)$$

Obteniendo una respuesta trivial $y = 1$, la cual no cumple la condición de que $1 < 0 < 2 < 21$, por tanto se debe buscar otro número.

Si se decide ignorar las dos condiciones necesarias y se intenta continuar con el algoritmo, aplicando el máximo común divisor para obtener los factores

$p = \gcd(21{,}0)$ y $q = \gcd(21{,}2)$

Se consigue el resultado trivial $p = 21$ y $q = 1$, por tanto se debe buscar otro valor aleatorio menor a *N* como se ha indicado.

Utilizando el valor de $x_2 = 13$, se calcula el máximo común divisor entre x_2 y *N,* $\gcd(21{,}13) = 1$. Luego se busca el orden de x_2:

$$13^r \equiv (1) mod(21)$$

Resultando $r = 2, 4, \ldots$ entonces se utiliza $r = 2$ para encontrar *y,*

$$13^1 \equiv (y) mod(21)$$

El valor que se obtiene es 13. Debido a que se cumple con la condición: $1 < 12 < 14 < 21$, se puede encontrar los factores de *N,* $p = 3$ y $q = 7$ mediante las ecuaciones:

$p = \gcd(21{,}12)$ y $q = \gcd(21{,}14)$

Finalmente con $x_3 = 15$, se puede encontrar un factor tan sólo realizando el máximo común divisor entre este valor y *N,* $\gcd(21{,}15) = 3$ y dividiendo *N* sobre el resultado se obtiene el otro factor $21/3 = 7$.

2.2.1.2. Shor Cuántico

A medida que el valor de *N* aumenta, poder encontrar el orden del número *x* resulta una operación más complicada, al punto de ser irrealizable en la capacidad computacional actual. Por ello Shor aprovechó las ventajas que permite la computación cuántica mediante su algoritmo, el cual evalúa una función periódica en una superposición de todas las entradas dentro de un rango amplio; y obtiene una superposición aproximada de los períodos de la función aplicando la transformada cuántica de Fourier, vista en el punto anterior (2.3.1.1.), con ese resultado se obtiene un período aleatorio *r* y se continua con el algoritmo para el cálculo de los factores de N, según la figura 2.1.

Antes de iniciar se asume que se ha escogido un número aleatorio $x < N$, tal que $\gcd(N, x) = 1$.

Se comienza utilizando un registro de cúbit, el cual se separa en dos conjuntos de memoria cuántica. En el primero existen *t* cúbits, donde *t* cumple la condición: $N^2 \leq 2^t \leq 2N^2$ y en el segundo registro *n* cúbits. Para poder representar los estados cuánticos se utiliza la notación de Dirac o bra-ket, $\langle\phi|\psi\rangle$, donde se conoce a la parte izquierda como bra $\langle\phi|$ y la derecha ket $|\psi\rangle$. El reporte original de Paul Dirac sobre este tema se llama: *"A new notation for quantum mechanics"*. [21]

$$|\psi_0\rangle = |r_1\rangle_t |r_2\rangle_n$$

Se puede representar el estado de todo el registro, donde la partición r_1 son todos los enteros hasta *t,* y la segunda partición todos los ceros*.*

$$|\psi_1\rangle = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle_t |0\rangle_n$$

Se realiza la operación $V(|j\rangle|k\rangle = |j\rangle|k\rangle + (x^j) mod(N)$ para cada entero de r_1, el resultado es una potencia de *x,* la cual se guarda en el segundo registro r_2. El estado de esta operación se representa como $|\psi_2\rangle$, la cual opera en todos los estados en la superposición simultánea.

$$|\psi_2\rangle = V|\psi_1\rangle = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} V(|j\rangle_t |0\rangle_n) = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle_t |x^j (mod\ N)\rangle_n$$

El paralelismo cuántico permite calcular todas las potencias de *x* simultáneamente. Debido a que se realiza la operación módulo en el segundo registro, es posible reescribir el estado $|\psi_2\rangle$ de tal forma:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^t}} [(|0\rangle + |r\rangle + |2r\rangle + \cdots)|1\rangle$$
$$+(|1\rangle + |r+1\rangle + |2r+1\rangle + \cdots)|x^1\rangle$$
$$+(|2\rangle + |r+2\rangle + |2r+2\rangle + \cdots)|x^2\rangle$$
$$\vdots$$
$$+(|r-1 \equiv -1\rangle + |2r-1 \equiv r-1\rangle + \cdots)|x^{r-1}\rangle \equiv (x^{-1}) mod(N)\rangle$$

Para cada ronda como máximo $\frac{2^t}{r}$ términos de la suma tienen período *r.* Se aplica la transformada cuántica de Fourier o su inversa en todos los estados base del primer registro para mapear cada estado en una superposición. [22]

$$|\psi_3\rangle == \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} (\frac{1}{\sqrt{2^t}} \sum_{j'=0}^{2^t-1} exp\left(2\pi i \frac{j' j}{2^t}\right) |j'\rangle_t)\ |x^j (mod(N))\rangle$$

Las características de la transformada cuántica de Fourier, incrementan la posibilidad de estimar múltiplos de $\frac{2^t}{r}$.

En el primer registro se realizó una estimación de un múltiplo aleatorio de $\frac{2^t}{r}$ y se aplicó la continuidad de fracciones para encontrar *r.*

La continuidad algorítmica de fracciones tiene la forma:

$$\frac{y}{2^t} = \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\ldots + \frac{1}{a_p}}}}$$

Se selecciona la convergencia con un denominador más pequeño que *N.* El denominador es *r* o un factor de r' de *r.* En el último caso se necesita calcular $x' \equiv (x^{r'}) mod(N)$ y aplicar la recursividad en x' para encontrar los factores restantes de *r.* Si se obtiene $y = 0$ se debe volver a realizar todo el algoritmo de nuevo hasta encontrar el orden. Si se cumple las condiciones necesarias, se calcula $y \equiv x^{r/2}$ para obtener los factores primos $p = \gcd(N, y-1)$ y $q = \gcd(N, y+1)$.

2.2.2. Algoritmo de Shor para el logaritmo discreto

Shor definió otro algoritmo para encontrar de manera rápida un valor desconocido *k* en la ecuación $h = (g^k) mod(p)$, si se reemplaza $mod(p)$ con una suma de puntos de una curva elíptica en $mod(p)$, se podría vulnerar algoritmos de curva elíptica como ECDSA, además de RSA.

El algoritmo funciona en tres registros cuánticos, utiliza dos exponenciales modulares y dos transformadas cuánticas de Fourier. Además de ello existe una optimización del mismo para la criptografía de curva elíptica (ECC).

El algoritmo de Shor reduce considerablemente la dificultad de resolver el problema del logaritmo discreto en F_q^* hasta:

$$O((logq)^2 loglog(q) logloglog(q))$$

En la computación clásica el algoritmo probabilístico riguroso más conocido posee una complejidad de: [23]

$$e^{O(\sqrt{logqloglogq})}$$

Mientras que el algoritmo probabilístico heurístico más conocido tiene una complejidad de: [23]

$$e^{O((logN)^{1/3}(loglogN)^{2/3})}$$

El reporte original de Peter Shor sobre este tema se llama: *"Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer"*.

2.2.3. Algoritmo de Grover

Uno de los algoritmos cuánticos más importantes actualmente es el algoritmo de búsqueda de Grover, el cual fue descrito en 1996. Este algoritmo es utilizado para realizar búsquedas de un factor que cumpla una condición específica en una base de datos desorganizada, y se considera que posee una complejidad de $O(\sqrt{n})$. En la figura 2.2 se puede observar el esquema del Algoritmo.

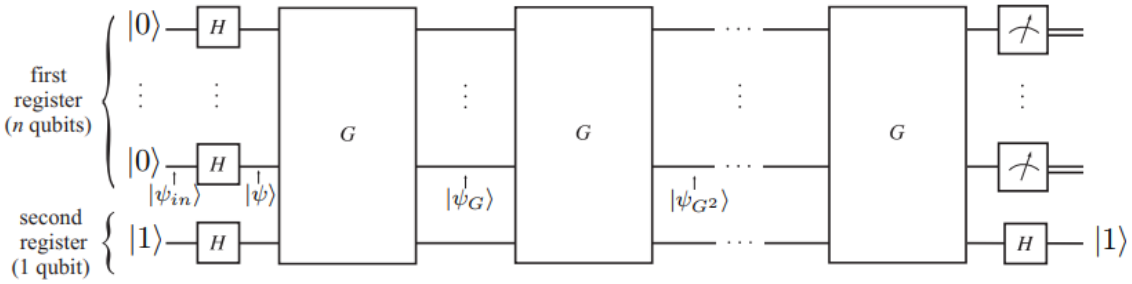

Figura 2.2: Esquema del Algoritmo de Grover. [24]

La estructura que muestra la figura 2.2, puede ser interpretada en la siguiente manera. Se comienza con un estado $v_1 \coloneqq |\psi\rangle|-\rangle$, para aumentar la amplitud del estado deseado en los primeros *n* cúbits se debe realizar un ciclo desde $r = 1$ hasta $r = k_0$. Este ciclo consiste en:

1. $v'_{r+1} \coloneqq U_f(v_r)$.
2. $v_{r+1} \coloneqq ((2|\psi\rangle\langle\psi|-|I)|\psi'_r\rangle)|-\rangle$.
3. Medir los primeros *n* cúbits.

Para poder encontrar un elemento que cumple una condición específica, dentro de una secuencia de N(para el caso $N = 2^n$) integrantes, es necesario comenzar a desarrollar el algoritmo con un registro de cúbits de tamaño $n + 1$, al cual se le denominará como *v* y será dividido en dos partes.

Antes de comenzar a desarrollar el algoritmo, es necesario tener a disposición una función tal, que su resultado sea 1, $f(x) = 1$, si en la posición ingresada *x* se encuentra el elemento a buscar, y 0 si no está. Así como la operación del bloque a utilizar U_f se define como:

$$U_f(|x\rangle) = (-1)^{f(x)}|x\rangle$$

La ubicación de la operación U_f, puede ser observada en la figura 2.3. Considerando que el estado original de todos los estados posibles posee la misma amplitud, entonces poseen igual probabilidad de ocurrencia, y a medida que ocurren las iteraciones, el algoritmo permite que la amplitud del estado a encontrar aumente $O(\frac{1}{\sqrt{N}})$. La cantidad de iteraciones necesarias es:

$$k_0 = [\frac{\pi}{4}\sqrt{N}]$$

La primera sección ($|\psi\rangle$) de *v* se define como la superposición de todos los estados posibles 2^n en la puerta $H^{\otimes(n)}$, cuyo resultado es una matriz de Hadamard $2^n \times 2^n$. Junto con la segunda sección ($|-\rangle$) se expresan a continuación:

$$|\psi\rangle \coloneqq H^{\otimes(n)}|0\rangle_n = \frac{1}{\sqrt{N}}|0,\dots,0\rangle_n + \cdots + \frac{1}{\sqrt{N}}|1,\dots,1\rangle_n = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle$$

$$|-\rangle \coloneqq H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Para aumentar la amplitud del estado deseado en los primeros *n* cúbits se debe realizar un ciclo desde $r = 1$ hasta $r = k_0$. Este ciclo consiste en:

1. $v'_{r+1} \coloneqq U_f(v_r)$.
2. $v_{r+1} \coloneqq ((2|\psi\rangle\langle\psi| - |I)|\psi'_r\rangle)|-\rangle$.
3. Medir los primeros *n* cúbits.

En el primer paso se aplica la operación U_f en el registro y la función del oráculo *f,* en la cual todos los posibles estados tienen la misma probabilidad $\frac{1}{N}$.

$$v_1 = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle|-\rangle$$

$$U_f(v_1) = U_f\left(\frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle|-\rangle\right) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} U_f(|i\rangle|-\rangle)$$

$$U_f(|i\rangle|-\rangle) = U_f\left(|i\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\right) = \frac{U_f(|i\rangle|0\rangle) - U_f(|i\rangle|1\rangle)}{\sqrt{2}}$$

$$\frac{U_f(|i\rangle|0\rangle) - U_f(|i\rangle|1\rangle)}{\sqrt{2}} = \frac{|i\rangle|0 \oplus f(i)\rangle - |i\rangle|1 \oplus f(i)\rangle}{\sqrt{2}} = (-1)^{f(i)}\left(\frac{|i\rangle|0\rangle - |i\rangle|1\rangle}{\sqrt{2}}\right)$$

$$= (-1)^{f(i)}|i\rangle|-\rangle$$

$$U_f(v_1) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} U_f(|i\rangle|-\rangle) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{f(i)}|i\rangle|-\rangle$$

$$v'_2 = U_f(v_1) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle - \frac{2}{\sqrt{N}}|i_0\rangle = |v'_2\rangle|-\rangle$$

Una vez finalizado el paso 1, se realiza la inversión sobre la media. Esta se desarrolla sobre la primera sección del registro cuántico, mientras que la segunda sección no se altera. El desarrollo de este paso, logrará aumentar la amplitud del estado deseado (i_0) y reducir el de los demás.

$$|\psi_G\rangle = 2|\psi\rangle\langle\psi| - I|\psi'_2\rangle = \frac{2^{n-2} - 1}{2^{n-2}}|\psi\rangle + \frac{2}{\sqrt{2^n}}|i_0\rangle$$

El resultado $|\psi_G\rangle$ corresponde al valor que se obtiene luego de la primera iteración. Su posición en el esquema puede verse en la figura 2.3. En cada iteración la amplitud del estado deseado aumenta.

Como último paso se realiza una medición de los primeros *n* cúbits. Según Grover, terminadas las iteraciones se consigue una probabilidad de por lo menos $O(i_0) = \frac{1}{2}$.

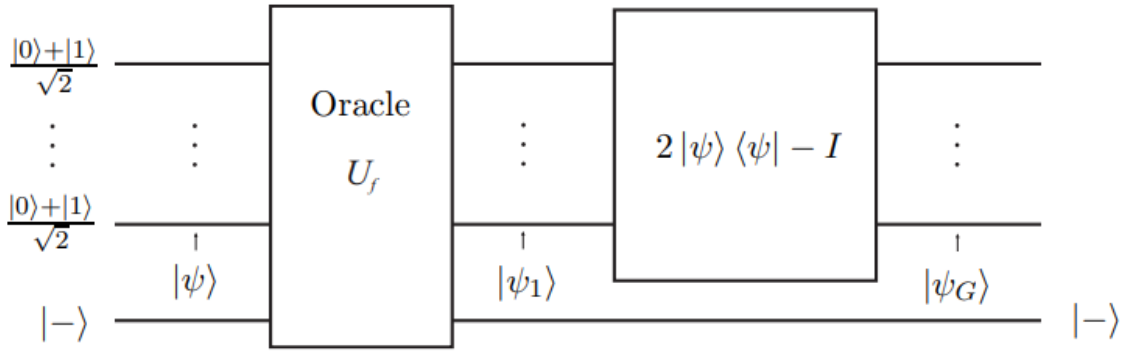

Figura 2.3: Esquema de una iteración en el Algoritmo de Grover, la cual corresponde a la primera iteración. [24]

2.3. Problemas matemáticos de la criptografía asimétrica actual

La criptografía asimétrica actual tiene aplicaciones como: cifrado, firma digital, distribución de claves, etc., y dependen de que su seguridad no pueda ser vulnerada; lo cual se logra mediante problemas matemáticos de dificultad computacional actual irresoluble. La criptografía está pensada con una certeza acerca del oponente que quiera romper su seguridad, y es que este posee acceso a la computación clásica y está limitado por el tiempo polinómico aleatorizado.

Debido a la importancia que los algoritmos poseen en la seguridad de transacciones web y aseguramiento de datos, los problemas que los fundamentan poseen una gran relevancia, al punto de que se ha demostrado teóricamente que la computación cuántica puede afectar la seguridad de los algoritmos.

Dentro de la matemática, los problemas que se pueden solucionar de manera eficaz usando la computación cuántica, se clasifican dentro del Problema de Subgrupos Ocultos (HSP). En el marco HSP, se incluyen problemas como la factorización de enteros positivos, el problema del vector más corto, el problema del logaritmo discreto, el isomorfismo gráfico, entre otros.

El principal instrumento utilizado en la resolución de los HSP, es el muestreo de Fourier, el cual permite encontrar la solución cuando el grupo subyacente es finito y abeliano, mediante el cálculo de la transformada y la medición de Fourier.

Un grupo se considera abeliano o conmutativo, si posee una operación que cumple la propiedad conmutativa en el grupo. [25]

Existen algunos casos, en que los problemas no encajan directamente en la definición de un HSP abeliano, y a pesar de ello se logra usar distintos métodos para encontrar la solución del problema en cuestión.

Actualmente un área de investigación en la matemática, es la solución de HSP no abelianos, es decir, cuando el grupo subyacente es no abeliano. Para ello se debe encontrar un algoritmo eficiente en la computación cuántica, debido a que el muestreo de Fourier no es suficiente para resolver un problema cuando los subgrupos ocultos no son normales y no se puede calcular la intersección de un conjunto de representaciones.

En la tabla 2.2 se muestran los algoritmos, los problemas que aseguran su fortaleza. Debido a que los problemas relacionados con la criptografía asimétrica actual más utilizada, se encuentran dentro de los HSP abelianos.

Algoritmo	Problema Computacional	HSP abeliano
Diffie-Hellman	Problema Diffie-Hellman	Sí
DSA	Problema del Logaritmo Discreto	Sí
ECDSA	Problema del Logaritmo Discreto en la Curva Elíptica Problema de factorización de enteros	Sí
ElGamal	Problema del Logaritmo Discreto Problema Diffie-Hellman	Sí
RSA	Problema de factorización de enteros Problema RSA	Sí

Tabla 2.2: Principales algoritmos asimétricos y su seguridad.

El algoritmo más común para resolver los HSP abelianos es el método estándar, el cual tiene por objetivo encontrar un conjunto de generadores para el subgrupo oculto H, en una serie de pasos polinomiales en $log|G|$.

CAPÍTULO 3: Principales opciones propuestas

Comprendidos los principales algoritmos, su seguridad, aplicación, además de sus problemas con el desarrollo de la computación cuántica, entonces se puede imaginar la devastación que conllevaría en la transmisión de datos su obsolescencia sin un reemplazo. A continuación se presenta las opciones más conocidas y que probablemente a futuro sean el sustituto de la criptografía asimétrica actual, todas se encuentran en etapa de desarrollo, algunas se encuentran más avanzadas que otras.

3.1. Encriptación basada en Código

La criptografía basada en código fue desarrollada desde el inicio de la criptografía asimétrica, la mayoría de los algoritmos más importantes de este tipo se basan en la propuesta original de Robert McEliece.

En 1978 el matemático Robert McEliece, realizó su propuesta tanto para la firma digital como la encriptación, es en esta última donde existe mayor aplicación, sin embargo el gran problema de este algoritmo es la longitud de las claves generadas, por ello existen intentos por mejorar este punto débil.

Hasta el momento los desarrollos generados a partir de este algoritmo no han logrado satisfacer la demanda de seguridad exigida para el correcto funcionamiento de un algoritmo de clave pública. A continuación se muestra la versión original propuesta por McEliece.

3.1.1. Mc Eliece

Este algoritmo como se ha mencionado se mantiene aún seguro, y a pesar de la longitud de sus claves, es considerado rápido, además hasta el momento la computación cuántica no parece mejorar los ataques existentes, por ello este sistema se mantiene como una opción para la criptografía post-quántica cuando utiliza códigos Goppa. [20]

El esquema se basa en códigos de corrección de errores, en síntesis se utiliza una clave privada la cual es un código aleatorio binario irreductible Goppa y la clave pública es una matriz generadora aleatoria de una versión permutada aleatoriamente de ese código. El texto cifrado es una palabra de código a la que se han agregado algunos errores, y solo el propietario de la clave privada (el código Goppa) puede eliminar esos errores. Este proceso se puede observar en la figura 3.1. [26]

Los códigos Goppa son una clase especial de códigos lineales, el código $\Gamma = \Gamma(\alpha_1, \dots, \alpha_n, \mathrm{g})$ consiste en todos los elementos $\mathrm{c} = (c_1, \dots, c_n)$ en F_2^n que satisfacen:

$$\sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} = 0$$

La dimensión típica de un código Γ, es $n - td$. [27]

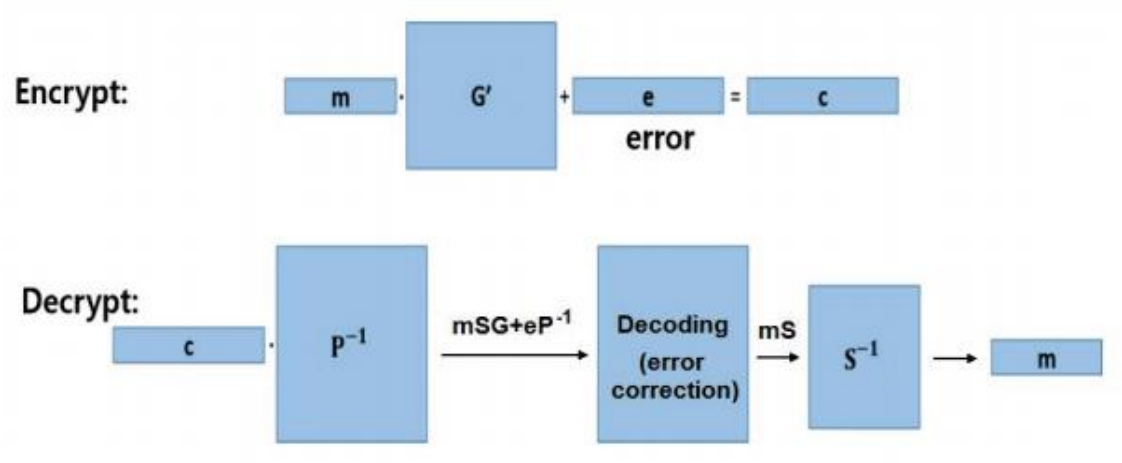

Figura 3.1: Esquema básico del algoritmo McEliece. [26]

El algoritmo propuesto es el siguiente:

1. Se definen los parámetros públicos $n, t, k \in \mathbb{N}$, donde $t \ll n$ y t es la cantidad de errores que se pueden corregir eficientemente.
2. Se escogen 3 matrices, una matriz generadora G, la cual posee dimensiones $k \times n$ para un código lineal binario, de distancia mínima $d \geq 2t + 1$.
3. La segunda matriz S, es aleatoria binaria no singular de dimensiones $k \times k$.
4. La última matriz, es aleatoria de permutación P, de tamaño $n \times n$.
5. Finalmente se calcula la matriz $\hat{G} = SGP$, la cual posee dimensión $k \times n$

Como resultado de los pasos antes descritos, se obtiene la clave pública A, la cuál es $(\hat{G}, t)$, mientras que la clave privada es (S, G, P). En el segundo paso se utiliza un código binario lineal Goppa $\Gamma = \Gamma(\alpha_1, \ldots, \alpha_n, \mathrm{g})$, Γ posee dimensión k, tal que $\mathrm{k} = \mathrm{n} - \mathrm{td}$. [27]

Para la encriptación de un texto m, es necesario seguir los siguientes pasos:

1. Se representa un mensaje de texto plano m como una cadena binaria de longitud k, tal que $m \in \mathbb{F}^k$.
2. Se escoge un vector de error binario aleatorio $z \in \mathbb{F}^n$, de peso t.
3. Calcular $c = m\hat{G} + z$.

Cuando el receptor quiera desencriptar el texto recibido, deberá calcular $\hat{c} = cP^{-1}$, donde P^{-1} es la matriz inversa de P.

$$\hat{c} = cP^{-1} = \left(m\hat{G} + z\right)P^{-1} = mSG + zP^{-1}$$

Aplicando el algoritmo de decodificación en $\hat{c}$, se puede obtener el término mSG. Debido a que G es una matriz generadora del código lineal Goppa Γ especificado en el segundo paso del algoritmo propuesto, mSG es una palabra código de Γ y el error permutado zP^{-1}, tiene un peso de t. [27]

Los valores originales que recomendó McEliece para su algoritmo fueron $n = 1024$, $t = 50$ y $k \geq 524$. El principal problema práctico con estos sistemas es el tamaño de la clave, aproximadamente un megabyte (en forma sistemática) a un alto nivel de seguridad.

Existen algunas modificaciones al sistema original de McEliece, la mayoría intentan solucionar el problema de la longitud de claves y mejorar el rendimiento. Lamentablemente algunas han sido rechazadas debido a los problemas de seguridad que plantean, entre ellas utilizar códigos cíclicos no binarios del tipo Reed-Solomon para producir las claves, los cuales fueron quebrados en 1992 por el ataque Sidelnikov-Shestakov. [27]

En la actualidad el matemático Niederreiter definió su propio criptosistema a partir del algoritmo de McEliece.

3.2. Encriptación basada en Retículas

Dentro de la encriptación de clave pública usando retículas, los algoritmos más conocidos son: GGH/HNF, NTRU y LWE. El primero de ellos fue propuesto por Goldreich. Glodwasser y Halevi, como un análogo de retículas del sistema McEliece, lamentablemente se encuentra quebrado en la práctica.

El segundo y el tercero se mantienen como seguros hasta el momento.

Para comprender este tipo de criptografía, primero se debe entender qué es una retícula. Una retícula es un grupo de puntos de dimensión espacial *n* con una estructura de periodo, se puede decir, que dada una cantidad *n* de vectores linealmente independientemente, $(b_1, b_2, \dots, b_n) \in \mathbb{R}^n$, la retícula formada por los vectores es:

$$\mathcal{L}(b_1, b_2, \dots, b_n) = \{\sum_{i=1}^{n} x_i b_i : x_i \in \mathbb{Z}, 1 \leq i \leq n\}$$

La base de esta retícula son los vectores $(b_1, b_2, \dots, b_n)$, la cual también se puede representar como una matriz $B = [b_1, b_2, \dots, b_n] \in \mathbb{R}^{nxn}$. Por tanto, la retícula formada por esta base también se puede representar como: $\mathcal{L}(B) = \{Bx : x \in \mathbb{Z}^n\}$. Si a la base B se la multiplica por una matriz U cuadrada entera con determinante ±1, la retícula formada por BU será igual a $\mathcal{L}(BU) = \mathcal{L}(B)$, con esto se demuestra que cualquier retícula puede ser formada por diferentes bases, esta particularidad permite que las retículas tengan aplicaciones criptográficas. En la figura 3.2 se observa un esquema básico de la encriptación de un algoritmo basado en retículas. [28]

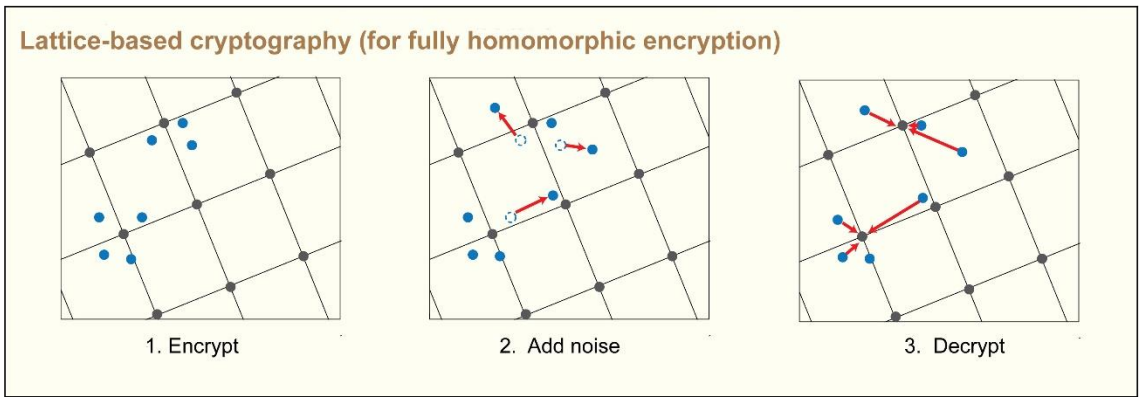

Figura 3.2: Esquema básico de encriptación en la criptografía basada en retículas. [28]

3.2.1. Retículas q-ary

Dentro de la criptografía asociada a las retículas, las retículas q-ary ocupan una posición preponderante.

Estas retículas poseen la característica, que la pertenencia o no de un vector *x* a la retícula $\mathcal{L}$ es determinada por el módulo del vector en un entero *q,* $(x)mod(q)$. Esto también implica, que sea la retícula $\mathcal{L}$ que satisface la regla $q\mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$, para un entero *q,* el cual debe ser un entero múltiplo del determinante de $\mathcal{L}$.

Para un $n \leq m$ y una matriz $A \in \mathbb{Z}_q^{nxm}$, se puede definir dos retículas q-ary de dimensión *m.*

$$\Lambda_q(A) = \{y \in \mathbb{Z}^m | \exists s \in \mathbb{Z}^n : y = (A^T s) mod(q)\}$$

$$\Lambda_q^{\perp}(A) = \{y \in \mathbb{Z}^m | Ay = (0) mod(q)\}$$

La primera es generada por las filas de A, mientras que la segunda contiene todos los vectores ortogonales al módulo *q* de las filas A. Existe un pequeño teorema que indica: $\Lambda_q(A) = q.\Lambda_q^{\perp}(A)^*$ y $\Lambda_q^{\perp}(A) = q.\Lambda_q(A)^*$.

Para probar esto se define $\Lambda_q(A) = \mathcal{L}(B)$ y se procede a calcular: [29]

$$q.\Lambda_q^{\perp}(A)^* = \mathrm{q}\left(\mathrm{q}\mathcal{L}(B^{-T})\right)^* = q\mathcal{L}(qB^{-T})^* = q\mathcal{L}(q^{-1}B) = \mathcal{L}(B) = \Lambda_q(A)$$

3.2.2. NTRU

NTRU fue presentado por primera vez en 1996 por tres matemáticos: Hoffstein, Pipher y Silverman. Para NTRU no existe aún un algoritmo en computación cuántica capaz de afectar su seguridad. El ataque más conocido fue descubierto por Howgrave-Graham.

Para desarrollar el algoritmo criptográfico NTRU, primero se deben definir las claves públicas y privadas. Para ello cada parte debe realizar los siguientes pasos:

1. Ambas partes deben acordar un primo *n*, además de dos enteros *p* y *q.*

2. Se define un par de vectores secretos $(f, g) \in \mathbb{Z}^{2n}$, los cuales deben cumplir lo siguiente:
    a. Existe el módulo inverso en p y q, $f.f_p = (1)mod(p)$ y $f.f_q = (1)mod(q)$
3. La clave pública se calcula mediante la siguiente formula: $h = (pf_q.g)mod(q)$, $h = h_0 + h_1x + \cdots + h_{p-1}x^{p-1}$; cada coeficiente pertenece al rango $\{0,1, \dots, q-1\}$.

3.2.2.1. Encriptación NTRU:

El cifrado en este sistema de clave pública, se consigue de la siguiente manera:

1. El mensaje a encriptar se debe colocar en forma de un vector $m \in \{1,0,-1\}^n$.
2. La parte que envía debe elegir de forma aleatoria un vector $r \in_R \{1,0,-1\}^n$.
3. El mensaje a enviar se calcula mediante: $c = (r.h + m)mod(q)$.

Si el receptor quiere obtener el mensaje original, deberá utilizar su clave privada y el mensaje enviado *c* de la siguiente forma:

Se multiplica la clave privada *f* con el mensaje cifrado $m = (f.c)mod(q)mod(p)$, lo cual produce vectores con coordenadas en $[-{}^{q}/_{2}, +{}^{q}/_{2}]$ y $[-{}^{p}/_{2}, +{}^{p}/_{2}]$.

$$(f.c) = \big(f(r.h + m)\big) = \Big(f\big(r\big)\big(pf_q.g\big) + f.m\big)\Big) = (p.rg + f.m)$$

$$(p.rg + f.m)mod(p)mod(q) = (f.m)mod(p)mod(q)$$

Finalmente se multiplica ese resultado por el módulo inverso f_p y se obtiene el mensaje original *m.*

$$f_p(f.m)mod(p)mod(q) = m$$

3.2.3. LWE

Es una de las opciones más populares de criptografía de clave pública a desarrollar dentro de la criptografía post-cuántica. LWE (Aprendiendo de los Errores) se encuentra respaldado por una prueba teórica de seguridad.

La primera versión del algoritmo junto con una prueba de seguridad fueron presentados por el matemático Oded Regev. Es considerado como el algoritmo más eficiente de la criptografía basada en retículas, y se describe en la figura 3.3.

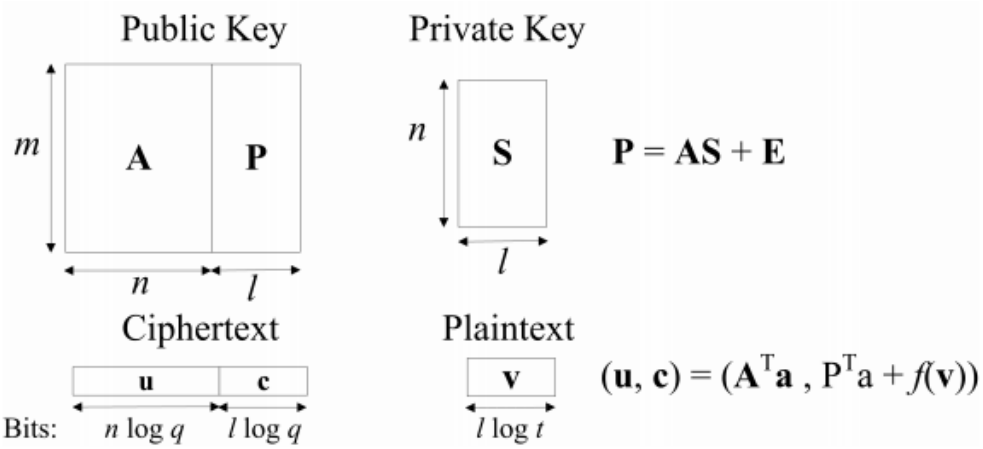

Figura 3.3: Esquema básico del algoritmo LWE. [30]

El esquema para la generación de claves LWE se compone de los siguientes pasos: [30]

1. Se definen los enteros n, m, l, t, r, q y una distribución de probabilidad $\psi_\propto$, donde $\propto$ es un valor real $\propto > 0$.
2. Se escoge de manera aleatoria un vector *S*, tal que $S \in \mathbb{Z}_q^{nxl}$, este vector será la clave privada.
3. Se selecciona un vector $A \in \mathbb{Z}_q^{mxn}$ uniformemente aleatorio y un vector $E \in \mathbb{Z}_q^{mxl}$, de tal manera que cada valor de acuerdo a una distribución $\psi_\propto$,.
4. La clave pública es $(A, P); P = AS + E$.

3.2.3.1. Encriptación LWE:

Una vez determinadas las claves pública y privada se puede encriptar mensajes, los cuales deben estar en el espacio $\mathbb{Z}_t^l$.

1. Dado un elemento del espacio mensaje $v \in \mathbb{Z}_t^l$ y una clave pública (A, P), seleccionar un vector aleatorio $a \in \{-r, -r+1, \dots, r\}^m$.
2. Calcular $u = A^T a$ y $c = P^T a + f(v)$.
3. Enviar el mensaje encriptado $(u, c) \in \mathbb{Z}_q^n x \mathbb{Z}_q^l$.

Para que el receptor pueda obtener el mensaje original, a partir del texto cifrado tan solo deberá evaluar la siguiente ecuación:

$$f^{-1}(c - S^T u)$$

Dentro de los parámetros escogidos, los valores de *n* y *l* son fundamentales en el algoritmo al ser utilizados tanto en la clave pública como la privada, mientras que la función *f* utilizada en el algoritmo, es aquella que mapea el espacio del mensaje $\mathbb{Z}_t^l$ a $\mathbb{Z}_q^l$, multiplicando cada coordenada por $^q/_t$ y redondeando al entero más cercano. Mientras que la función inversa, se

define como el mapeo de los elementos del espacio $\mathbb{Z}_q^l$ a $\mathbb{Z}_t^l$, y cuyo resultado se multiplica por $^t/_q$ y se redondea al entero más cercano.

3.3. Firma digital basada en Retículas

Se han desarrollado varios algoritmos para firma digital basados en retículas, entre ellos los formulados por el matemático Goldreich y la compañía NTRU Cryptosystems, la cual planteó el esquema de firma NTRUSign. Lamentablemente ambos sistemas presentaron serias fallas de seguridad, por lo que fueron descartados.

En el año 2012 Lyubashevsky y Micciancio lograron desarrollar un algoritmo, el cual tiene garantías de seguridad en el peor de los casos basadas en retículas ideales, y es la construcción más eficiente conocida hasta la fecha.

3.3.1. Esquema Lyubashevsky y Micciancio:

Como se ha mencionado este esquema supone el mejor algoritmo propuesto para la firma digital basada en retículas, su seguridad se basa en la imposibilidad de resolver el problema SVP.

Este sistema es considerado como un esquema de firma única basado en hash, es decir, un esquema de firma que permite firmar de forma segura un solo mensaje.

El algoritmo se describirá mediante matrices, aunque existen variantes utilizando secuencia de vectores. Para comenzar a firmar se deben seguir los siguientes pasos:

1. Se definen los enteros k, n, m, q y una función de Hash H (se asume que es computacionalmente difícil encontrar colisiones para H).
2. Se escoge una matriz $S \in \mathbb{Z}_q^{mxk}$, cuyas entradas poseen un valor menor que q. Esta matriz será la clave privada.
3. Se selecciona una matriz $A \in \mathbb{Z}_q^{nxm}$, es decir, es de la dimensión $n\ x\ m$ con enteros módulo q.
4. La clave pública será una matriz $T = (AS)mod(q)$, la cual tiene dimensión $n\ x\ k$.

Antes de proceder a la firma de un mensaje, el firmante selecciona y a partir de una distribución de dimensión m, típicamente una distribución gaussiana discreta. Luego se debe utilizar la siguiente fórmula para obtener la firma del mensaje m'.

$H([Ay]mod(q), m') = c$ y $Sc + y = z$

La firma que se envía al receptor es el par(c, z). Si el receptor de la firma quiere comprobar que la misma es auténtica, debe calcular:

$$H([Az - Tc] mod(q), m') = c$$

Esto debido que $Az - Tc \equiv A(Sc + y) - ASc \equiv (Ay) mod(q)$. Cabe aclarar que la función H se define como $H: \mathbb{Z}_q^n \times \{-1,0,1\}^* \rightarrow \{0,1\}^k$ donde los vectores de salida satisfacen que no más de k entradas son diferentes de cero.

3.4. Firmas Digitales basadas en Hash

Los algoritmos de firma digital basados en Hash, son aquellos que utilizan la función Hash como algoritmo criptográfico para la firma, por ello su seguridad se basa en la dificultad de encontrar colisiones para los datos a ingresar en la función.

Debido a la velocidad de cálculo y la seguridad que caracteriza a las funciones Hash, existen varios grupos que se encuentran trabajando en esta alternativa como opción de criptografía post-cuántica. Entre ellos el Instituto Nacional de Estándares y Tecnología (NIST) de EEUU. y el Grupo de Trabajo de Investigación de Internet (IRTF).

3.4.1. Lamport–Diffie one-time signature scheme (LD-OTS)

En el año 1975 el matemático Lamport aprovechó esta condición para desarrollar un algoritmo de firma de única vez.

Para comenzar se debe seleccionar un entero positivo n, este valor es considerado el parámetro de seguridad del esquema. LD-OTS utiliza dos funciones f y g, ambas funciones g serán conocidas tanto por el destinatario como el firmante.

- f es una función de una sola vía $f: \{0,1\}^n \rightarrow \{0,1\}^n$
- g es una función criptográfica de hash $g: \{0,1\}^n \rightarrow \{0,1\}^n$.

El par de claves LD-OTS a generar son X clave de firma y Y clave de verificación. Para generar X se debe tomar una cadena de $2n$ bits de longitud n elegidas uniformemente aleatorios.

$$X = (x_{n-1}[0], x_{n-1}[1], \dots, x_1[0], x_1[1], x_0[0], x_0[1]) \in_R \{0,1\}^{(n,2n)}$$

Mientras que la clave Y consiste en:

$$Y = (y_{n-1}[0], y_{n-1}[1], \dots, y_1[0], y_1[1], y_0[0], y_0[1]) \in_R \{0,1\}^{(n,2n)}$$

Donde,

$$y_i[j] = f(x_i[j]), 0 \leq i \leq n-1, j = 0,1$$

Entonces, la generación de claves LD-OTS requiere $2n$ evaluaciones de f. Las claves de firma y verificación son cadenas de $2n$ de longitud n.

3.4.1.1. Firma digital LD-OTS

Para firmar un mensaje M, se deben seguir los siguientes pasos:

1. Se debe colocar en la forma $M \in \{0,1\}^*$
2. Luego calcular $g(M) = d = (d_{n-1}, \dots, d_0)$, con ese resultado se calcula la firma.
3. $\sigma = (x_{n-1}[d_{n-1}], \dots, x_1[d_1], x_0[d_0]) \in \{0,1\}^{(n,n)}$

Como resultado se obtiene una firma de longitud n^2, esto debido a que la firma es una secuencia de n cadenas de bits, cada una de longitud n.

Para que el destinatario pueda verificar la autenticidad de la firma σ del mensaje M, debe calcular $g(M) = d$, y verificar:

$$(f(\sigma_{n-1}), \dots, f(\sigma_0)) = (y_{n-1}[d_{n-1}], \dots, y_0[d_0])$$

Para ejemplificar el funcionamiento de LD-OTS, se tomará un valor $n = 4$, $f: \{0,1\}^4 \to \{0,1\}^4$, $x \to (x+2) mod(16)$ y $d = (1,0,0,1)$ el valor de hash del mensaje M. Entonces se escoge la clave de firma X:

$$X = (x_3[0], x_3[1], x_2[0], x_2[1], x_1[0], x_1[1], x_0[0], x_0[1]) \in_R \{0,1\}^{(4,8)}$$

$$X = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \in_R \{0,1\}^{(4,8)}$$

Y se calcula la clave de verificación:

$$Y = (y_3[0], y_3[1], y_2[0], y_2[1], y_1[0], y_1[1], y_0[0], y_0[1]) \in_R \{0,1\}^{(4,8)}$$

$$Y = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \in_R \{0,1\}^{(4,8)}$$

Entonces la firma de d sería:

$$\sigma = (\sigma_3, \sigma_2, \sigma_1, \sigma_0) = (x_3[1], x_2[0], x_1[0], x_0[1]) \in \{0,1\}^{(4,4)}$$

$$\sigma = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

A pesar de que este esquema agilita las operaciones necesarias para encontrar la firma de un mensaje, posee poca aplicación práctica debido a la necesidad de generar claves continuamente para cada firma. Por ello Ralph Merkle propuso una solución para este problema con el árbol de Merkle.

3.4.2. Merkle Signature Scheme (MSS)

El árbol de Merkle, es una solución propuesta por Ralph Merkle al problema de la necesidad de generar un hash diferente por cada firma, en el cual cada nodo tiene dos nodos hijos y un nodo padre, a excepción de los nodos hoja y

raíz. Esta propuesta funciona con cualquier función de hash y esquema de firma única.

Para comenzar la generación del par de claves del MSS se selecciona una función criptográfica de hash $g: \{0,1\}^* \rightarrow \{0,1\}^n$, luego el firmante escoge un valor H, el cuál es el tamaño del árbol y debe ser un número natural mayor que 2, a partir de allí el par de claves generado podrá firmar y verificar 2^H documentos, en la figura 3.4 se muestra un ejemplo de MSS para $H = 2$.

El firmante genera 2^H par de claves únicas: (X_j, Y_j), $0 \leq j < 2^H$. Donde X_j es la clave de firma y Y_j es la clave de verificación, ambos son cadenas de bits elegidas aleatoriamente. Los nodos hoja son el resultado de $g(Y_j), 0 \leq j < 2^H$, mientras que los nodos ramas son el resultado del hash de la concatenación de los nodos hijos. Esto se puede describir matemáticamente como:

$$v_h[j] = g(v_{h-1}[2j] || v_{h-1}[2j+1]), 1 \leq h \leq H, 0 \leq j < 2^{H-h}$$

Finalmente la clave pública del MSS es la raíz del árbol de Merkle, mientras que la clave privada es la secuencia 2^H de claves de firma única.

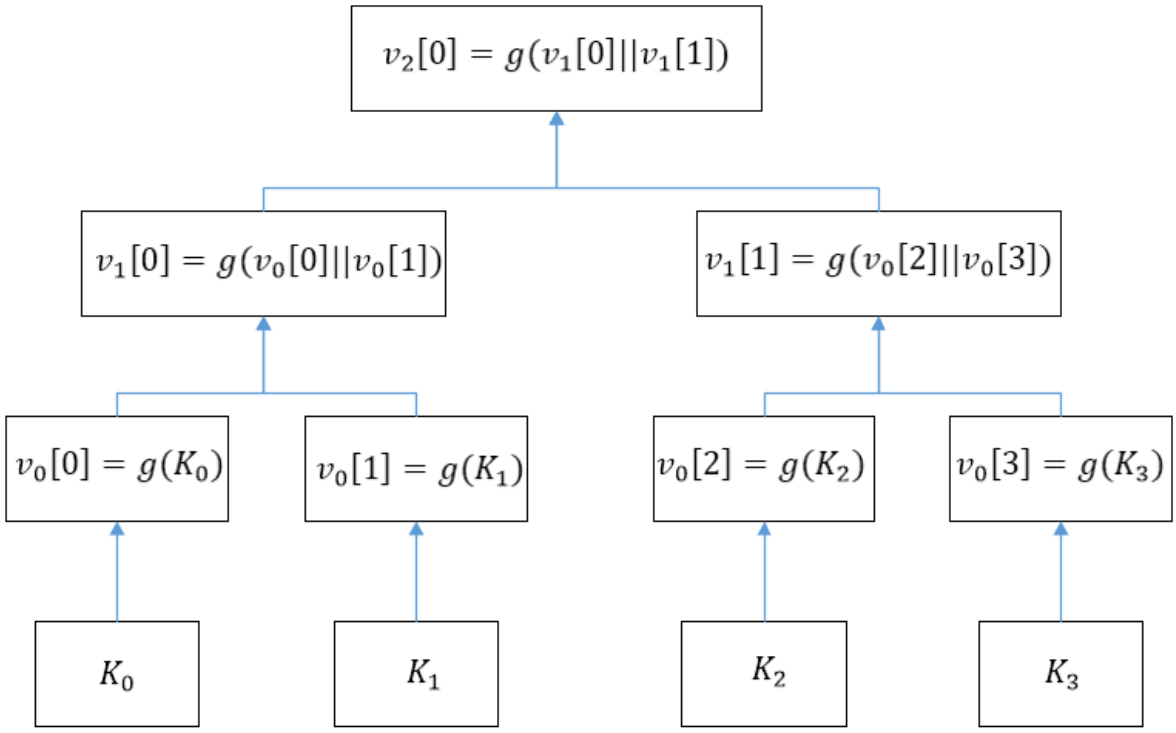

Figura 3.4: Árbol de Merkle para un MSS de tamaño $H = 2$.

3.4.2.1. Firma digital MSS

El proceso de firmar digitalmente usando MSS es el siguiente:

1. Para comenzar a firmar un mensaje M, primero se calcula $d = g(M)$.
2. Luego se genera la firma única σ_{OTS} del resultado d, para ello se selecciona la j-ésima clave de firma $X_j, j \in \{0, \ldots, 2^H - 1\}$.

3. Junto con lo anterior se envía la clave de verificación Y_j.
4. Para probar la autenticidad de Y_j, el firmante incluye el índice j.
5. Finalmente se adjunta la ruta de autenticación para verificar la clave Y_s, esta ruta es la secuencia de verificación de nodos del árbol de Merkle $A_s = (a_0, \dots, a_{H-1})$, desde el nodo hoja hasta la raíz.

Un ejemplo de la firma de un mensaje M es el siguiente:

$$\sigma_j = (j, \sigma_{OTS}, Y_j, (a_0, \dots, a_{H-1}))$$

Para el caso de j=2 y H=2, la figura 3.5 muestra el árbol de Merkle, y la firma digital es la siguiente:

$$\sigma_2 = (2, \sigma_{OTS}, Y_2, (a_0, a_1))$$

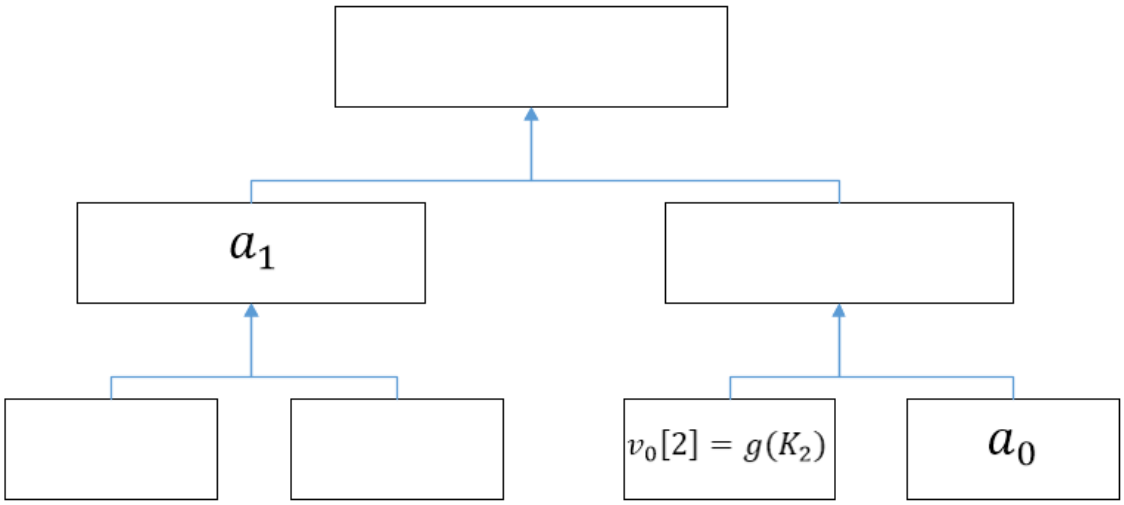

Figura 3.5: Árbol de Merkle para un MSS de tamaño $H = 2$, con los nodos de la ruta de verificación para $g(Y_2)$.

Para la verificación de la firma el receptor debe realizar los siguientes pasos:

1. Primero se utiliza la clave de verificación Y_j para comprobar que la firma única σ_{OTS} del esquema de clave firma única escogido.
2. El verificante construye la ruta $(p_0, \dots, p_H)$ desde la j-ésima hoja hasta la raíz del árbol de Merkle.

Para realizar el camino, se utiliza la información de la firma σ_j y la siguiente función:

$$p_h = \begin{cases} g(a_{h-1} || p_{h-1}), si\ [j/2^{h-1}] \equiv 1\ mod(1) \\ g(p_{h-1} || a_{h-1}), si\ [j/2^{h-1}] \equiv 0\ mod(2) \end{cases}$$

Donde $h = 1, \dots, H$ y $p_0 = g(Y_j)$, la clave de verificación Y_j se comprueba si y sólo si p_H es la clave pública.

3.5. Firmas Digitales basadas en Ecuaciones Cuadráticas de Múltiples Variantes

Los criptosistemas de clave pública múltiples variantes (MPKC) se basan en un grupo de polinomios (usualmente cuadráticos) sobre un campo finito. La seguridad de los criptosistemas MPKC se basa en la dificultad de resolver ecuaciones no lineales en un campo finito.

A pesar, de que en la última década se han desarrollado varios esquemas MPKC muchos han visto comprometida su seguridad. Aunque existen desarrollos tanto para la encriptación como firma digital, es en esta última donde se ha logrado mayores avances. [20]

El inicio de este tipo de criptografía se remonta al año 1988, cuando los matemáticos Matsumoto e Imai introdujeron el esquema de firma C^*, el cual fue quebrado en el año 1995 por el francés Jacques Patarin, no obstante al siguiente año Patarin presentó un sistema basado en la idea de Matsumoto e Imai, el sistema de ecuaciones de campos ocultos (HFE), del cual también se han realizado variantes con mayor o menor fortaleza.

3.5.1. Esquema General de MPKC

La mayoría de las variantes de MPKC, parten de una estructura similar. La cual se puede describir mediante el siguiente algoritmo:

1. Se comienza definiendo dos valores enteros $n, m \geq 1$.
2. Un campo finito k, y q la cantidad de elementos en k.
3. Se definen los mapas afines invertibles S y T, donde $S: k^n \to k^n$ y $T: k^m \to k^m$. Y el mapa cuadrático invertible G, $G: k^n \to k^m$.
4. La clave privada son los tres mapas (S, G, T).
5. La clave pública es la convolución $F: k^n \xrightarrow{S} k^n \xrightarrow{G} k^m \xrightarrow{T} k^m$.

Para firmar un mensaje en plano $x \in k^n$, se debe calcular $y = T^{-1} \circ G^{-1} \circ S^{-1}(m)$. Y para verificar el mensaje se utiliza la clave pública de la siguiente manera: $F(y) = m$. [31] [32]

En esta estructura, el rol que cumplen los mapas afines secretos S y T, es el de asegurar que el mapa F no pueda ser fácilmente invertible. Las primeras versiones de MPKC fueron quebradas por el uso de mapas (S, T) arbitrarios. [31]

Las variantes más seguras actualmente de sistemas MPKC, son: Rainbow, PMI+, HFE^{v-}.

3.5.2 $\mathbf{HFE^{v-}}$

También conocida como Quartz, en esta variante MPKC para firma digital, se define una secuencia de polinomios $(p_1, \ldots, p_m)$ en el anillo polinomial de n variables $\mathbb{F}_2[x_1, \ldots, x_n]$ como clave pública, $m \leq n$.

Cada polinomio p_i, tiene la forma $a_i + \sum_j b_{i,j}\, x_j + \sum_{j<k} c_{i,j,k}\, x_j x_k$, los coeficientes no tienen una estructura pública obvia. Mientras que la firma de un mensaje es una cadena de n-bit $(s_1, \dots, s_n) \in {\mathbb{F}_2}^n$. Se estima que los valores que permitirían asegurar la resistencia de este tipo de MPKC contra ataques cuánticos serían $m = 240, n = 272\ y\ q = 2^{256}$.

3.6. Soluciones basadas en Álgebra no conmutativa y no asociativa

Además de las opciones vistas hasta el momento, existen otras alternativas las cuales por su estructura algebraica no se consideran como parte de las familias antes descritas. [20]

Una de estas alternativas es la utilización de protocolos basados en estructuras no conmutativas (NCC) y no asociativas (NAC). Algunos ejemplos de este tipo de soluciones son: Anillos no conmutativos, semigrupos, cuasigrupos, grupos policíclicos, entre otros. La ventaja de esta solución, es que puede ser utilizada para desarrollar protocolos de intercambio de claves, transporte de claves, protocolos de autenticación, cifrado o firma digital. [33] [34]

Es importante señalar que debido a su versatilidad, las estructuras algebraicas NCC y NAC poseen varias operaciones computacionalmente complejas que sustentan su seguridad matemática. Entre ellas se pueden mencionar:

- Problema de la Búsqueda del Elemento Conjugador (CSP)
- Problema de Doble Clase Lateral (DCP)
- Problema de Descomposición (DP)
- Problema de la Descomposición Simétrica (SD)
- Problema de la Descomposición Simétrica Generalizada (GSDP)

Para estos problemas, no existe por el momento un algoritmo capaz de brindar una solución en tiempo polinómico, sin embargo, tampoco se ha logrado demostrar que poseen una complejidad No Polinómica (NP). A pesar de ello, el campo de estructuras algebraicas no conmutativas y no asociativas es uno de los que más atención está recibiendo en los últimos años.

Una de sus grandes ventajas es el poder adaptar estas ideas a la criptografía actual vista en el capítulo 1. Por ello se han desarrollado varios trabajos usando esta clase de soluciones. A continuación, se muestra cuatro trabajos sobre soluciones basadas en NCC y NAC, los tres primeros comparten entre otras cosas, el uso del problema GSDP como función trampa de una vía.

El problema GSDP se puede generalizar como:

"Dado un grupo no conmutativo G y un subgrupo conmutativo escondido S, conociendo $(x, y) \in G^2$ y $(m, n) \in \mathbb{Z}^2$, encontrar $z \in S$, donde $y = z^m x z^n$". [35]

Para lograr el uso del problema GSDP en los trabajos enumerados a continuación, también se define un grupo general lineal $GL(8, F_{251})$.

3.6.1. One Round Conference Key Distribution for Multiple Entities: A Post-Quantum Approach [33]

En el desarrollo mencionado en [33], se propone una solución al problema de concretar una clave de sesión aleatoria para n-partes, la cual a su vez es resistente a ataques criptográficos.

La base matemática se fundamenta en un grupo general lineal de matrices cuadradas modulares y no singulares.

Al igual que en el algoritmo Diffie-Hellman, existen elementos conocidos por las partes y otros que cada una mantiene en secreto. En esta solución, las entidades partícipes acuerdan utilizar los valores d y p, los cuales corresponden al orden de las matrices cuadradas y al valor del campo primo F_p. Además de hacer públicas dos matrices (P, G) y dos valores enteros aleatorios (m, n). Estos elementos aleatorios son miembros del conjunto Z_p^*.

Cada una de las n-partes deberá elegir una matriz diagonal M_i de orden d y un par de enteros (m_i, n_i).

- M_i diagonal con valores seleccionados aleatoriamente en $(Z_p^*)^d$
- (m_i, n_i) seleccionados aleatoriamente en $(Z_p^*)^2$

Con estos valores cada entidad comparte con las otras, un valor que utilizarán para conseguir la clave de sesión compartida, esta operación se conoce como ronda de intercambios y sólo se registra una vez sin importar cuántas n-entidades intervengan.

$$M_i' = {M_i}^{m_i}.G.{M_i}^{n_i}$$

Finalmente se obtendrá el valor de sesión mediante las siguientes operaciones:

$$X = M_1' * M_2' * \ldots * M_{n-1}' * {M_i'}^{m_i+n_i}$$

$$K = X^m.P.X^n$$

En la primera multiplicación, se destaca que la matriz M_i' se multiplica una sola vez al final, en el caso que un posible atacante quiera obtener la clave de sesión a partir de conocer la matriz M_i', primero deberá resolver el problema GSDP.

En el desarrollo mencionado [33], se muestra una ejemplificación matemática del mismo con tres partes que intervienen y acuerdan usar $d = 8$ y $p = 251$. El autor menciona que en el caso de necesitar aumentar el valor de p, *"se debe asegurar que el orden multiplicativo lo permita"*.

3.6.2. Post-Quantum Cryptography: A Zero-Knowledge Authentication Protocol [36]

Como se mencionó anteriormente las soluciones basadas en NAC y NCC no se limitan a una sola clase de uso de criptografía asimétrica, en este desarrollo

se muestra una solución para la autenticación entre partes, que se clasifica como protocolo de conocimiento cero.

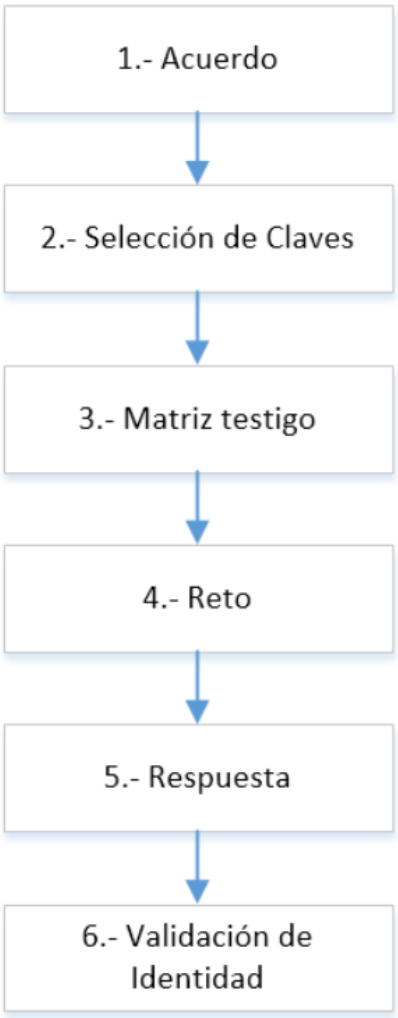

Figura 3.6: Diagrama de flujo del algoritmo de autenticación ZKP, establecido en el trabajo. [36]

El algoritmo cuenta con seis pasos que se muestran en la figura 3.6, en el primero de ellos las entidades participantes acuerdan el grupo general lineal, el grupo no conmutativo, y el subgrupo conmutativo (en el trabajo descrito $GL(8, F_{251})$, M_8 y P_8 respectivamente). A partir de allí se acordarán dos matrices (P, G) que pertenecen al grupo y dos valores enteros positivos (m, n).

En el siguiente paso, cada entidad participante seleccionar una clave privada, para ello genera una matriz diagonal aleatoria con valores no repetidos escogidos en $\mathbb{Z}_p^*$ ($\mathbb{Z}_{251}^*$ en [36]). Luego utilizando la matriz P, se genera una clave privada $(A, B, C, ..)$, y la clave pública $(G_A, G_B, G_C, \dots)$ de cada entidad se forma a partir de los valores m, n, G y la clave privada.

$$\text{Cada entidad escoge } (\lambda_1, \dots, \lambda_8) \in_R \mathbb{Z}_{251}^*$$

$$A = PD_AP^{-1}, B = PD_BP^{-1}, C = PD_CP^{-1} \dots$$

$$G_A = A^m G A^n, G_B = B^m G B^n, G_C = C^m G C^n, \dots$$

Un posible atacante podría intentar obtener las claves privadas a partir de las claves públicas, pero a pesar de que conozca (m,n) y (P,G), las claves privadas se encuentran seguras por el problema GSDP, para el cual como se ha mencionado no existe una solución en tiempo polinómico.

En el tercer paso una entidad ocupa el rol de verificador y el otro de sujeto de prueba, estos roles pueden ser intercambiados. La entidad que prueba su identidad crea una matriz testigo (S) y se lo envía al verificador.

$$k \in_R \mathbb{Z}^+$$

$$S = A^k G_A A^{-m}$$

El valor k, sólo lo conoce el sujeto de prueba, al igual que la clave secreta.

Cuando el verificador recibe la matriz testigo S, crea el reto para identificar a la otra entidad, con ese fin genera un bit b, una pregunta Q y se los envía al sujeto de prueba.

$$b \in_R \{0.1\}$$

Si $b = 0$ luego $H \in_R M_8$ y $Q = B^m H B^n$

Si $b = 1$ luego $Q = B^m S G_A B^n$

En el quinto paso, la entidad prueba su identidad respondiendo el reto y le envía la respuesta R al verificador de la siguiente forma:

Si $b = 0$ luego $R = S^{-m} Q S^{-n}$

Si $b = 1$ luego $R = A^{-k} Q A^{-n}$

Finalmente, se verifica la respuesta R.

Si $b = 0$ acepta si $Q = S^m R S^n$

Si $b = 1$ acepta si $G_B G = B^{-m} R B^{-n}$

En el caso que los valores concuerden se aceptará la identidad, caso contrario se repetirán los pasos desde el tercero al último, hasta que el verificador se encuentre seguro de la identidad de la otra parte.

Como se ha mencionado este protocolo de autenticación es considerado ZKP (Protocolo de Conocimiento Cero), esto debido a que satisface tres propiedades: integridad, solidez y conocimiento cero.

En el caso que ambas partes sean honestas, el verificador podrá determinar la entidad a prueba sin importar cuántas veces se haga el reto, tan sólo quien posea la clave privada los superará. Esta es la condición de integridad, la cual se comprueba de la siguiente forma:

Si $b = 0$ entonces $S^m R S^n = S^m S^{-m} Q S^{-n} S^n = Q$.

Si $b = 1$ entonces $B^{-m} R B^{-n} = B^{-m} A^{-k} Q A^{-n} B^{-n} = B^{-m} A^{-k} B^m S G_A B^n A^{-n} B^{-n} = B^{-m} A^{-k} B^m A^k G_B A^{-m} A^m G B^n A^{-n} B^{-n} = G_B G$.

Para el caso de la solidez, una entidad puede fingir ser otra e inventar una clave privada que pertenece al subgrupo no conmutativo, a partir de allí podrá tener éxito en la mitad de los casos, cuando b sea igual a 0. Afortunadamente un 50% de casos de éxito es muy bajo como para verificar una entidad, por lo cual el deshonesto sujeto de prueba se descarta.

En el caso del conocimiento cero, se indica que se cumple sólo si existe un algoritmo simulador capaz de imitar los informes de sesión válidos con valores falsos, totalmente indistinguibles de los verdaderos. Al igual que en la integridad si un atacante utiliza un algoritmo simulador no podrá demostrar la identidad de un tercero a menos que posea su clave privada.

De acuerdo con los valores seleccionados en el primer paso, se puede determinar la seguridad del algoritmo, en el trabajo expuesto se utilizó un grupo lineal $GL(8, F_{251})$, lo cual brinda una seguridad de 64 bits. Si se necesita aumentar la seguridad se puede seleccionar un grupo $GL(16, F_{251})$, el cual brinda una seguridad de 127 bits.

3.6.3. Post-Quantum Cryptography: Generalized ElGamal cipher over GF(251^8)** [37]

En este desarrollo se parte del algoritmo ElGamal, el cual fue mencionado en el capítulo 1. La diferencia por la cual se conjetura que puede resistir ataques de tipo cuánticos, es por la base sobre la cual se desarrolla.

Al ser un algoritmo de encriptación, la transacción se desarrolla entre dos partes. Como se observa en la tabla 3.1, comienzan acordando valores comunes para ambas partes, estas matrices pertenecen al grupo no conmutativo M_8. A continuación, cada parte genera dos valores secretos, así como una matriz diagonal con valores únicos, la cual se conjuga al igual que en el paso de *"selección de claves"* del desarrollo 3.6.2.

Para crear un valor clave común K para ambas partes, primero crean tokens los cuales son intercambiados en el paso *"Interchange tokens".* Con este valor cualquiera de las partes está en capacidad de encriptar un mensaje y enviarlo a la otra para ser desencriptado.

En el caso expuesto en la tabla 3.1, Alice desea encriptar un mensaje y enviarlo a Bob, por ello Bob debe actualizar su token de sesión B' y enviarlo a Alice. El cual será utilizado en la creación de los valores (y_1, y_2) que contienen el mensaje encriptado. Cabe destacar que el mensaje H debe ser parte del grupo no conmutativo M_8.

A pesar de la dificultad que *a priori* se muestran en los procedimientos matemáticos expuestos en la tabla 3.1. En el trabajo desarrollado en [37], se muestra un ejemplo práctico del desarrollo planteado desde la definición de (P, G) hasta la desencriptación del mensaje.

	Alice	Bob
Any entity begins	$P \in_R M_8$ $G \in_R M_8$	
Generating private elements	$k_1, k_2 \in_R \mathbb{Z}^*_{251}$ $\forall \neq \lambda_1, \ldots, \lambda_8 \in_R \mathbb{Z}^*_{251}$ $D_A = (\lambda_1, \ldots, \lambda_8)$ $A = PD_AP^{-1} \in P_8$	$r_1, r_2 \in_R \mathbb{Z}^*_{251}$ $\forall \neq \mu_1, \ldots, \mu_8 \in_R \mathbb{Z}^*_{251}$ $D_B = (\mu_1, \ldots, \mu_8)$ $B = PD_BP^{-1} \in P_8$
Interchange tokens	$A' = A^{k_1}GA^{k_2} \implies$	$B' = B^{r_1}GB^{r_2} \implies$
First common key K is obtained	$K = A^{k_1}B'A^{k_2}$ $m = K^{\nearrow}_{8,1}.K^{\swarrow}_{1.8}$ $n = K^{\searrow}_{1,1}.K^{\nwarrow}_{8.8}$	$K = B^{r_1}A'B^{r_2}$ $m = K^{\nearrow}_{8,1}.K^{\swarrow}_{1.8}$ $n = K^{\searrow}_{1,1}.K^{\nwarrow}_{8.8}$
	$K = A^{k_1}T_BA^{k_2} = A^{k_1}(B^{r_1}G_0B^{r_2})A^{k_2}$ $= B^{r_1}(A^{k_1}G_0A^{k_2})B^{r_2} = B^{r_1}T_AB^{r_2} = K$	
Alice start a new cipher session updating recursively parameters	$K = K^{m.n}$ $m = K^{\nearrow}_{8,1}.K^{\swarrow}_{1.8}$ $n = K^{\searrow}_{1,1}.K^{\nwarrow}_{8.8}$ $P = K^mPK^n$ $G = K^mGK^n$ $A = PD_AP^{-1}$ $A' = A^mGA^n \implies$	
Bob acknowledges and update parameters		$K = K^{m.n}$ $m = K^{\nearrow}_{8,1}.K^{\swarrow}_{1.8}$ $n = K^{\searrow}_{1,1}.K^{\nwarrow}_{8.8}$ $P = K^mPK^n$ $G = K^mGK^n$ $B = PD_BP^{-1}$ $B' = B^mGB^n \implies$
Alice ciphers an H message to Bob	$H \in M_8$ $J \in_R P_8$ $C = (y_1, y_2) \implies$ $y_1 = J^mGJ^n$ $y_2 = H(J^mB'J^n)$	
Bob deciphers H		$H = y_2(B^my_1B^n)^{-1}$
	$H = y_2(B^my_1B^n)^{-1}$ $= H(J^mB'J^n)(B^my_1B^n)^{-1}$ $= H(J^mB^m)G_1(B^nJ^n)(B^my_1B^n)^{-1}$ $= H(B^m(J^mG_1J^n)B^n)(B^my_1B^n)^{-1}$ $= H(B^my_1B^n)(B^my_1B^n)^{-1}$ $= H$	

Tabla 3.1: Procedimiento para la creación de una clave común, el cifrado y descifrado de un mensaje. [37]

Si un posible atacante quisiera descifrar una clave privada de una entidad participante mediante fuerza bruta, sería necesario encontrar la matriz diagonal dentro del grupo no conmutativo M_8, el cual posee un cardinal de 10^{19} aproximadamente 2^{64}. Con los algoritmos actuales es imposible atacar este algoritmo por fuerza bruta.

Finalmente, en la demostración realizada en [37] el procedimiento completo toma alrededor de 85 ms, valor que puede llegar a ser optimizado.

3.6.4. Post-Quantum Cryptography: S_{381} Cyclic Subgroup of High Order [38]

En este desarrollo a diferencia de los anteriores la base matemática es un grupo de permutación aleatoria, no un grupo general lineal. Por ello la seguridad no se sustenta en el problema GSDP. Una buena razón para usar grupos de permutación aleatoria es la facilidad de construir un grupo aleatorio de alto orden y luego un generador de un subgrupo cíclico.

Para ejemplificar el uso de un grupo cíclico, se expone en [38] un caso en el cual dos entidades emplean el protocolo Diffie-Hellman para intercambio de claves y ElGamal para la encriptación y desencriptación de un mensaje. El procedimiento completo se ilustra mediante un ejemplo matemático en [38].

En este caso que se trabaja con un grupo cíclico de orden 381, se debe utilizar una dimensión de listas de tamaño 16, como se muestra en la figura 3.7. Primero se lista los primeros 16 primos, luego se realizan dos listas más, una con la suma y la otra con la multiplicación de los números en la primera lista.

Los últimos valores de la segunda y tercera lista, son el orden del grupo simétrico y el orden del subgrupo cíclico respectivamente, estos parámetros son públicos.

```
Dim= 16

Prime list= {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53}

Partition sum=
 {2, 5, 10, 17, 28, 41, 58, 77, 100, 129, 160, 197, 238, 281, 328, 381}

Primorial list=
 {2, 6, 30, 210, 2310, 30030, 510510, 9699690, 223092870,
  6469693230, 200560490130, 7420738134810, 304250263527210,
  13082761331670030, 614889782588491410, 32589158477190044730}
```

Figura 3.7: Listas de dimensión 16. [38]

Al utilizar el protocolo Diffie-Hellman, que se expone en la figura 3.8, el problema que brinda seguridad al intercambio de claves es el DLP. Los valores que escogen Alice y Bob para elevar el generador p pertenecen al grupo definido por los reales iguales o inferiores al orden de p.

Para definir p, se realiza una permutación con los valores entre 1 y 381.

(a) PUBLIC VALUES (preparation)
S_{381}: permutation group (non-commutative); $|S_{381}| \equiv 381! \sim 3.596379714 \times 10^{819}$
$p \in_R S_{381}$ generator of the <p> subgroup; $|\langle p \rangle| \equiv \Omega = 32589158477190044730$
(b) PRIVATE VALUES
ALICE_power $(a) \in_R \mathbb{Z}_\Omega$; ALICE private exponent
BOB power $(b) \in_R \mathbb{Z}_\Omega$; BOB private exponent
(c) CALCULATED TOKENS interchanged
ALICE_Token $(t_a) = p^a$
BOB_Token $(t_b) = p^b$
(d) ALICE calculates the session key
ALICE_key $(k) = (t_b)^a == p^{ba}$
(e) BOB calculates the session key
BOB_key $(k) = (t_a)^b == p^{ab}$

Figura 3.8: Diffie-Hellman utilizado en el trabajo. [38]

Mientras que la versión del cifrador ElGamal, posee dos versiones, una en la cual basa su seguridad en el DCP y la otra en DP. Para ambos problemas no existen por el momento algoritmos capaces de resolverlos en tiempo polinómico, pero al igual que GSDP tampoco se ha demostrado que son resistentes a ataques cuánticos.

La variante que se muestra en la figura 3.10 diferencian su procedimiento de la figura 3.9, en el uso de una permutación extra, la cual impacta en todos los pasos descritos, desde el cálculo de valores privados hasta la desencriptación de un mensaje.

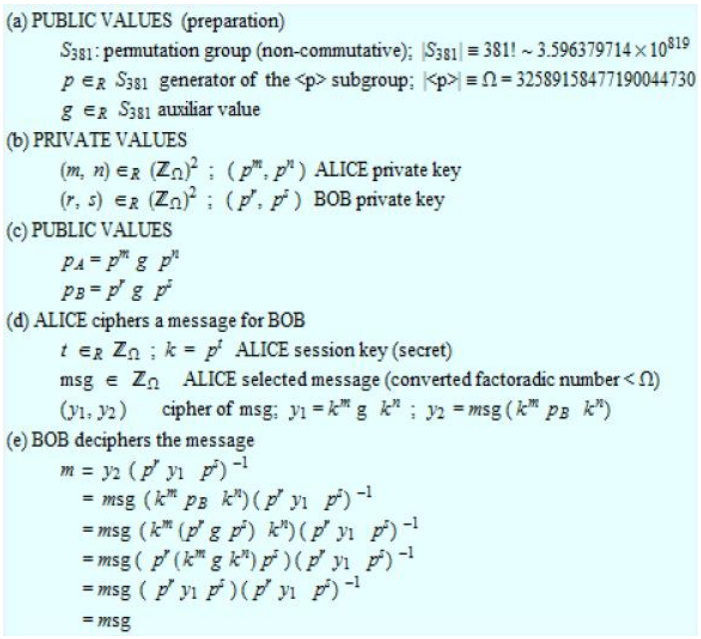

Figura 3.9: Variante de ElGamal usando DCP. [38]

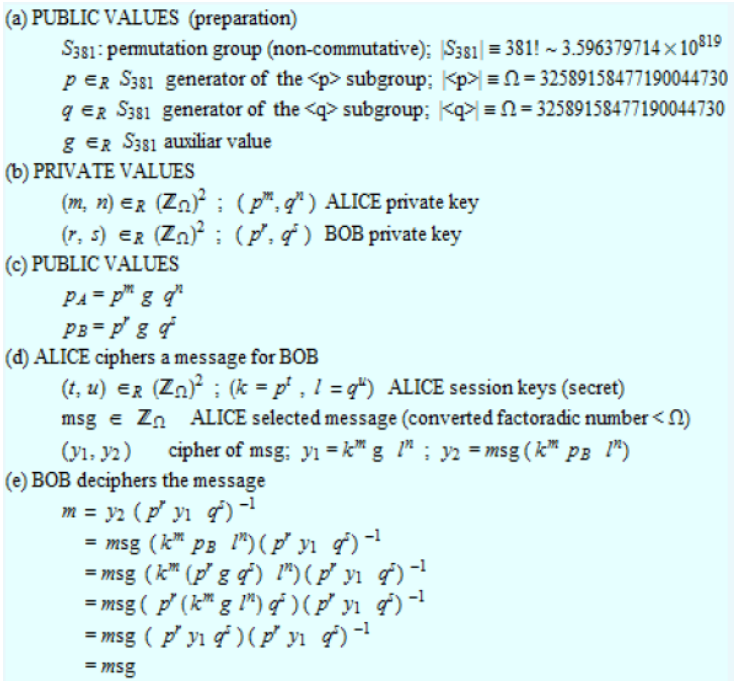

Figura 3.10: Variante de ElGamal usando DP. [38]

En la ejemplificación matemática descrita en [38], se mostró el uso tanto de Diffie-Helman, como la variante descrita de ElGamal descrita en la figura 3.9.

Con este trabajo se logró desarrollar una solución basada únicamente en operaciones combinatorias simples, sin necesidad de bibliotecas aritméticas o de números grandes.

Conclusiones

Una vez concluido el trabajo final de especialización, se pueden obtener las siguientes conclusiones:

- En el mundo de la criptografía, una idea nunca puede ser totalmente descartada. Muchas de las opciones actuales vistas en el capítulo 3 a los ataques de computación cuántica, son ideas que se propusieron en el inicio de la criptografía asimétrica, las cuales quedaron en el olvido con la aparición de RSA.
- Para que una idea criptográfica sea tomada en cuenta como algoritmo asimétrico, es necesario no sólo establecer condiciones de fortaleza matemática, además de ello rapidez de cálculo y limitar la extensión de las claves pública y privada.
- A pesar de que existan varias opciones al momento para hacer frente a los ataques cuánticos, no todas estas opciones de criptografía asimétrica poseen la misma credibilidad.
- En la actualidad, el tipo de criptografía asimétrica con mayores condiciones para resistir ataques cuánticos es la criptografía basada en Hash. La cual posee por el momento mayor madurez y pruebas de seguridad. Esto se refleja en las tablas c.1 y c.2.
- A pesar de que el campo de las estructuras algebraicas no asociativas y no conmutativas no se encuentra entre las principales opciones para reemplazar a la criptografía asimétrica actual, existen desarrollos que demuestran que las estructuras algebraicas NCC y NAC, pueden ser planteadas en entornos que no necesitan de grandes recursos computacionales, manteniendo niveles de seguridad aceptables.
- En el caso que se demuestre que el problema GSDP es resistente a ataques cuánticos, la conjetura expuesta en el punto 3.6.1 puede resolver de manera más sencilla el intercambio de claves entre n-partes en relación con la solución de funciones bilineales propuesta por el matemático francés Joux.
- Los trabajos expuestos en los puntos 3.6.3 y 3.6.4 muestran que no existe una única forma de desarrollar soluciones utilizando NCC y NAC.
- Las soluciones propuestas en el punto 3.6, son menos propensas a la manipulación de sus resultados que la criptografía actual, esto se debe a que no es necesario elementos de un tercero, como bibliotecas de precisión extendida o el sometimiento de su seguridad a generadores de números pseudo-aleatorios.

A continuación, se muestran dos tablas (c.1 y c.2) extraídas de la bibliografía general, que evidencian el estado al momento, de las principales opciones vistas en el capítulo 3.

Tipo	Ventajas	Desventajas
Encriptación basada en Código (Usando Códigos Goppa)	Alta confianza en seguridad, encriptación muy rápida, textos cifrados cortos	Largas claves públicas
Encripción basada en Retículas (Usando NTRU o relacionados)	Textos cifrados y claves cortos, Encriptación muy rápida	Requiere mayor análisis de seguridad
Firma basada en Retículas	Claves cortas y firmas rápidas	Requiere mayor análisis de seguridad, ataques de canal lateral en gausianos discretos
Firmas MPKC	Firmas muy cortas	Requiere mayor análisis de seguridad
Firmas basadas en Hash	Alta confidencialidad, descripción simple	Administración de estados

Tabla c.1: Ventajas y Desventajas de las opciones descritas en el capítulo 3.

Tipo	Firma	Cifrado	Largo de Claves (en Byte)	Madurez Criptográfica
Basado en Hash	Si	No	20	Alta
MPKC	Si	No	10.000	Baja, media para esquemas conservadores
Basado en Retículas (General)	Posible	Si	Menor que 100	Media
Basado en Retículas (NTRU)	Posible	Si	100.000	Media
Basado en Código	Costoso	Si	100.000	Alta con precauciones de implementación

Tabla c.2: Principales características de los algoritmos descritos en el capítulo 3.

Bibliografía específica

[1] R. Oppliger, Contemporany Cryptography, Norwood, Massachusetts: Artec House, 2005.

[2] G. Renault, «Les bases de la Cryptologie».

[3] H. Scolnik, *Criptografía Asimétrica y RSA,* Buenos Aires, 2017.

[4] J. S. Fernández, «Repositorio Universidad de Cantabria,» Universidad de Cantabria, 2013. [En línea]. Available: https://repositorio.unican.es/xmlui/bitstream/handle/10902/3101/Jennifer%20Santamaria%20Fernandez.pdf?sequence=1. [Último acceso: 3 Marzo 2018].

[5] W. Stallings, Cryptography and Network Securtity, Fourth Edition, Prentice Hall, 2005.

[6] I. Guwahati, «National Programme in technology enchanced learning,» 2 Enero 2013. [En línea]. Available: http://nptel.ac.in/courses/106103015/36. [Último acceso: 3 Marzo 2018].

[7] «PLATAFORMA DE APRENDIZAJE KZGUNEA,» [En línea]. Available: http://e-forma.kzgunea.eus/mod/book/tool/print/index.php?id=3949. [Último acceso: 2 Febrero 2018].

[8] K. B. Z. D. P. G. M. G. J. A. H. N. H. D. S. E. T. L. V. David Adrian, «Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice,» weakdh.org.

[9] D. Ireland, «RSA Algorithm,» 2018.

[10] National Institute of Standards and Technology, «NIST,» Julio 2013. [En línea]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf. [Último acceso: 23 Abril 2018].

[11] Rivest Ronald L, MIT Laboratory for Computer Science.; Burt Kaliski, RSA Laboratories, «RSA Problem,» 2003.

[12] S. P. y. M. Hellman, «An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance,» *IEEE Transactions and Information Theory 24,* pp. 106 - 110, 1978.

[13] H. D. y. L. D. Barrera Ángel, «Criptografía,» 2015. [En línea]. Available: https://criptografia.webnode.es/algoritmos-asimetricos/algoritmo-elgamal/. [Último acceso: 29 Julio 2018].

[14] Wenbo Mao Hewlett- Packard Company, Modern Cryptography: Theory and Practice, Prentice Hall PTR, 2003.

[15] A. N. S. Institute, «Public Key Cryptography For the Financial Services Industry: ECDSA,» 20 Septiembre 1998. [En línea]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.202.2977&rep=rep1&type=pdf. [Último acceso: 7 Mayo 2018].

[16] G. C. Kessler, «An Overview of Cryptography,» 22 Febrero 2018. [En línea]. Available: https://www.garykessler.net/library/crypto.html#pgp. [Último acceso: 8 Mayo 2018].

[17] T. Kenta, «Hitachi,» 26 Mayo 2014. [En línea]. Available: http://www.hitachi.com/rd/portal/contents/story/pbi/index.html. [Último acceso: 5 Mayo 2018].

[18] «1965: "Moore's Law" Predicts the Future of Integrated Circuits,» [En línea]. Available: http://www.computerhistory.org/siliconengine/moores-law-predicts-the-future-of-integrated-circuits/. [Último acceso: 3 Noviembre 2018].

[19] P. C. W. Davies, «Quantum tunneling time,» American Association of Physics Teachers, Sidney, 2004.

[20] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner y D. Smith-Tone, «Report on Post-Quantum Cryptography,» National Institute of Standards and Technology, 2016.

[21] P. Dirac, «A new notation for quantum mechanics,» vol. 35, nº 3, pp. 416-418, 1939.

[22] H. E. Caicedo-Ortiz, «Algoritmo de factorización para un,» México DF., 2010.

[23] M. Mosca, «Quantum Algorithms,» Waterloo, 2008.

[24] C. Lavor, «Grover's Algorithm: Quantum Database Search,» Petrópolis, Brazil, 2008.

[25] Y. Ershov, «Abelian group,» Encyclopedia of Mathematics, [En línea]. Available: https://www.encyclopediaofmath.org/index.php/Abelian_group. [Último acceso: 13 Diciembre 2018].

[26] Y.-S. K. y. J.-S. N. Wijik Lee, «Code-Based Post-Quantum,» Seoul National University, Seoul, 2017.

[27] T. L. a. C. P. Daniel J. Bernstein, «Attacking and defending the McEliece cryptosystem,» Springer-Verlag, Berlin, 2008.

[28] Computer Science, «Chasing real security: An interview with Allison Bishop Lewko,» Columbia University, Nueva York, 2015.

[29] L. W. Jesko Hüttenhain, «Topics in Post-Quantum Cryptography,» 2011.

[30] J. H. S. K. a. s. Yu Yao, «A Sub-0.5V Lattice-Based Public-Key Encryption,» University of Virginia, Charlottesville, 2011.

[31] Y. Hashimoto, «Multivariate Public Key Cryptosystems,» de *Mathematical Modelling for Next-Generation*, Tokyo, Springer, 2018.

[32] J. B. y. N. Bindel, «Post-Quantum Cryptography,» Technische Universität Darmstadt, Hessen, 2015.

[33] J. P. Hecht, «One Round Conference Key Distribution for Multiple Entities: A Post-Quantum Approach,» Buenos Aires, 2017.

[34] J. P. Hecht, «A Post-Quantum Set of Compact Asymmetric,» Universidad de Buenos Aires, Buenos Aires, 2015.

[35] D. X. L. W. Z. Cao, New public-key cryptosystems using polynomials over non-commutative rings, Shanghai: Shanghai Jiao Tong University, 2007.

[36] J. P. Hecht, «Post-Quantum Cryptography: A Zero-Knowledge Authentication Protocol,» Universidad de Buenos Aires, Buenos Aires, 2017.

[37] J. P. Hecht, «Post-Quantum Cryptography: generalized ElGamal cipher over GF(251^8),» Universidad de Buenos Aires, Buenos Aires, 2017.

[38] J. P. Hecht, «Post-Quantum Cryptography: S381 Cyclic Subgroup of High Order,» International Journal of Advanced Engineering Research and Science (IJAERS) , Buenos Aires, 2017.

Bibliografía General

- Handbook of Applied Cryptography – Alfred Menezes.
- Post-Quantum Cryptography – Daniel J. Bernstein & Johannes.
- Post quantum cryptography: Implementing alternative public key schemes on embedded devices – Stefan Heyse