



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE INFORMÁTICA

GRADO EN MATEMÁTICAS

Intercambio de claves: esquemas cuánticos y postcuánticos

Autor: Dimas Muñoz Montesinos

Tutora: María Isabel González Vasco

Curso académico 2017/2018

Resumen

El presente trabajo surge por la motivación de aprender acerca de los algoritmos criptográficos para el intercambio de clave entre dos usuarios usados en la actualidad y los problemas que acarrearán. Se pretende dar una visión superficial sobre el estado actual de la computación cuántica, para poder comparar las técnicas cuánticas de intercambio de clave con un método clásico que se cree resistente a ataques cuánticos (i.e., de los llamados post-cuánticos). Para ello está dividido en dos bloques fundamentales: el protocolo BB84 y el algoritmo RLWE (ver [1]).

El esquema BB84. En el primero, desde una perspectiva teórica, abordaremos el protocolo BB84. Aunque sea algo antiguo nos sirve de punto de partida para profundizar en este campo, puesto que se trata del nacimiento de la criptografía cuántica. Además, se trata de un protocolo que destaca por ser la base de la que parten la mayoría de implementaciones de algoritmos de criptografía cuántica actuales. Se abordarán temas como el funcionamiento del protocolo, las ventajas que ofrece frente a algoritmos criptográficos actuales y algunos problemas de implementación que aún tiene. En multitud de estudios posteriores sobre otros protocolos de criptografía cuántica encontramos referencias al BB84, lo que nos puede dar una noción de su importancia (ver [4]).

Esquemas post-cuánticos para intercambio de claves. El segundo bloque es algo más práctico. Trata de dar una visión global a los algoritmos post-cuánticos y, una vez hecha esta introducción, se centra en el algoritmo *Ring-Learning with errors* (ver [1]). Como veremos más adelante, este es uno de los algoritmos con más potencial para sustituir a los actuales, aunque todavía falta por definir estándares y realizar más estudios que avalen su seguridad. Asimismo, se adjunta junto a este trabajo el código fuente de la implementación hecha en Python 3 del algoritmo RLWE. Dicho algoritmo forma parte de la implementación planteada en el documento *Post-Quantum Key Exchange*.

Índice general

1. Introducción	8
1.1. Qué es la criptografía	8
1.2. Criptografía moderna: Diffie-Hellman y RSA	9
1.3. Comienzos de la computación cuántica	12
1.4. Criptografía cuántica	13
2. Protocolo BB84	16
2.1. Introducción	16
2.2. Funcionamiento	16
2.3. Seguridad del protocolo	20
2.4. Limitaciones e implementaciones	21
3. Esquemas basados en retículos	23
3.1. Introducción a esquemas postcuánticos	23
3.2. Qué es la criptografía basada en retículos	24
3.3. Ring-Learning con errores	25
3.3.1. Notaciones previas	25
3.3.2. El problema RLWE	26
3.3.3. Diffie-Hellman-like basado en RLWE	27

3.3.4. Seguridad de RLWE	28
3.3.5. Implementación en Python	29
4. Conclusiones	32
A. Apéndice	34
A.1. Algoritmo Shor	34
A.2. Definiciones y lemas de RLWE	38
Bibliografía	39

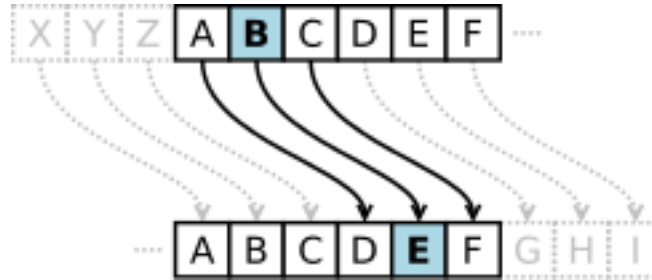
1. Introducción

1.1. Qué es la criptografía

La criptografía es la ciencia que estudia la gestión (almacenamiento, transmisión y procesado) de información en entornos hostiles con garantías de seguridad. Una de las herramientas criptográficas más importantes son los esquemas de cifrado. Aquí hay que hacer una distinción entre dos tipos de cifrado: clave secreta y clave pública.

- Por un lado, el cifrado de clave privada (también conocido como cifrado simétrico) que consta de una única clave compartida por ambas partes. El principal problema de este esquema se encuentra en la distribución de la clave a través de un canal no seguro. No obstante, a pesar de que pueda parecer muy simple, es bastante útil en ciertos ámbitos, como por ejemplo el cifrado digital (ver [3], pág. 3 y 4).
- Por otro lado, el cifrado de clave pública (o cifrado asimétrico) que consta de dos claves: una privada y otra pública. Cada participante dispone de un par de claves: una clave se mantiene secreta (la privada) y la otra clave se distribuye a quien lo desee (la clave pública). Cualquier individuo que posea la clave pública será capaz de cifrar un mensaje y mandárselo al participante que tiene la clave privada. Solamente éste último será el que pueda volver a descifrar el mensaje, ya que se necesita la clave privada para ello.

La criptografía ha sido uno de los avances más importantes en el desarrollo de la humanidad, es casi tan antigua como la escritura. Para poner esto en contexto nos remontaremos a los tiempos de Julio César, cuando él mismo empezó a utilizar un cifrado para realizar comunicaciones militares y que, además, lleva su mismo nombre: cifrado César. Este cifrado, de clave secreta, tan solo consistía en desplazar 3 letras el abecedario de tal forma que, por ejemplo, si se quería escribir la letra A, en su lugar se escribiría la letra D (ver [2]).



Este cifrado puede ser representado usando la aritmética modular. Suponiendo que usemos el alfabeto español, sea $E(x)$ la función de codificación, $D(x)$ la de decodificación y n el desplazamiento:

$$E_n(x) = x + n \bmod 27$$

$$D_n(x) = x - n \bmod 27$$

A modo de anécdota, el mafioso Bernardo Provenzano fue detenido recientemente y, a pesar de todos los avances tecnológicos actuales, se encontró que usaba una pequeña modificación del cifrado César.

Así pues, la criptografía siempre ha jugado un papel fundamental en la transmisión de mensajes de tal forma que no puedan ser leídos y/o modificados por agentes externos. Desde aquí empezaremos a denominar Alice y Bob a los dos individuos que quieren comunicarse entre ellos sin que un tercero sea capaz de interceptar las comunicaciones. A este tercer individuo le llamaremos Eve.

En la criptografía clásica existe el concepto de código indescifrable al que se le llama “Cuaderno de un solo uso” (o “one-time-pad” en inglés). Fue inventado por Gilbert Vernam en 1918, por lo cual también recibe el nombre de cifrado Vernam. Es un tipo de algoritmo en el que un texto plano se combina con una clave aleatoria (que sea como mínimo igual de larga que el texto) usando la operación XOR. Entonces, si la clave es verdaderamente aleatoria y no la reutilizamos, el texto será totalmente aleatorio y no dará ninguna información de lo que contiene (ver [4], pág. 6).

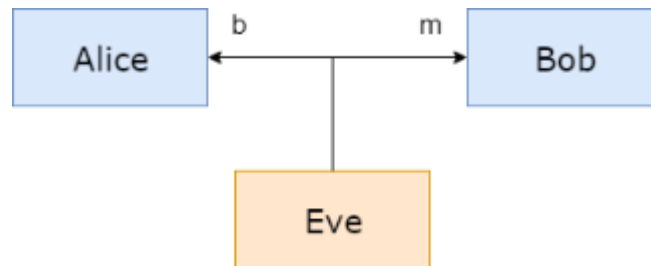
1.2. Criptografía moderna: Diffie-Hellman y RSA

A finales de los años 70 surgen dos de los protocolos criptográficos más utilizados en la actualidad. El primero, en el año 1976, fue el protocolo criptográfico Diffie-Hellman, el cual describe cómo pactar claves entre dos individuos que no han tenido contacto previo, utilizando un canal inseguro, y de manera anónima (no autenticada). Su principal aplicación es la de acordar una clave simétrica con la que posteriormente

cifrar los mensajes entre ambas partes (ver [5]). Además, la simplicidad del protocolo permitió que su uso se extendiera fácilmente. En honor a sus autores, Whitfield Diffie y Martin Hellman, el protocolo recibió este nombre.

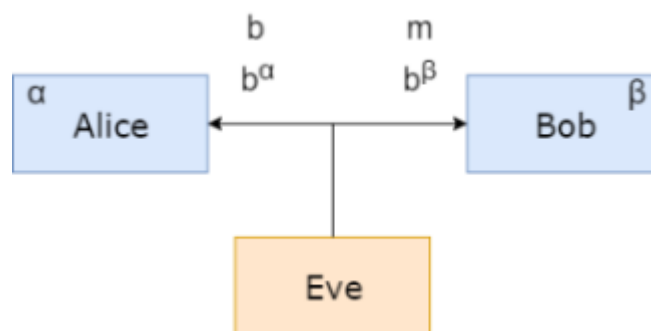
Este protocolo basa su seguridad en la dificultad de calcular un logaritmo discreto en un cuerpo finito. Pero, ¿en qué consiste exactamente este protocolo?

Alice y Bob quieren acordar una clave con la que comunicarse de manera segura. Esta clave la obtendrán a partir de una potencia modular, así que lo primero será ponerse de acuerdo en la base b y el módulo m que van a usar. Eve, que estará atenta a las comunicaciones, podrá anotar esos números ya que los transmiten públicamente.

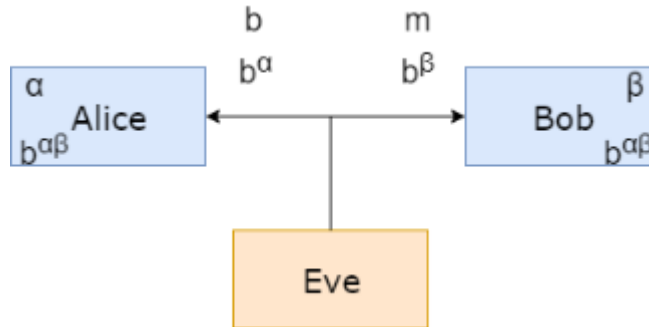


Ahora, Alice y Bob elegirán un número cualquiera cada uno, α y β respectivamente, los cuales los mantendrán en secreto.

Con estos números, por un lado, Alice calculará b^α y le enviará el resultado a Bob. Por otro lado, Bob hará el mismo proceso, calculará b^β y se lo enviará a Alice. Mientras tanto, Eve podrá tomar nota tanto de b^α como de b^β .



Ahora, Alice calculará la potencia α del resultado de Bob b^β . Bob realizará el mismo proceso. De esta forma, ambos participantes estarán en condiciones de calcular la clave K . Esto se debe a la propiedad de las potencias tal que $(a^n)^m = (a^m)^n$ para unos números cualesquiera.



Formalizando esta explicación:

- Sean m primo, $b \in \mathbb{Z}_m^*$.
- Sean $\alpha \in \mathbb{Z}_{m-1}$ elegido al azar por Alice, y $A = b^\alpha \mod m$
- Sean $\beta \in \mathbb{Z}_{m-1}$ elegido al azar por Bob, y $B = b^\beta \mod m$

Tanto Alice como Bob podrán calcular el valor $K = b^{\alpha \cdot \beta} \mod m$ por las propiedades del grupo \mathbb{Z}_m^* . En el caso de Alice podemos ver que (idem en el caso de Bob):

$$B^\alpha \mod m = (b^\beta \mod m)^\alpha \mod m = \overbrace{((b^\beta \mod m)(b^\beta \mod m) \cdots (b^\beta \mod m))}^\alpha \mod m = b^{\beta \cdot \alpha} \mod m = b^{\alpha \cdot \beta} \mod m = K$$

Queda demostrado que ambas partes pueden compartir una clave sin que Eve la conozca pero, ¿podría llegar a calcular α y β ? La respuesta es no, o al menos con los ordenadores actuales, ya que debería invertir la función, es decir, debería calcular $\alpha = \log_m(A)$ o $\beta = \log_m(B)$, lo cual no es computable en un tiempo razonable. Este es el problema del logaritmo discreto en \mathbb{Z}_m^* .

Tan solo un año más tarde, en 1977, Ronald Rivest, Adi Shamir y Leonard Adleman tomaron estas bases y determinaron que la función matemática ideal para crear un sistema de cifrado asimétrico con clave pública era la factorización (recordemos que el protocolo Diffie-Hellman se basa en un cifrado simétrico o de clave privada). Al igual que el protocolo Diffie-Hellman, este algoritmo recibió el nombre RSA en honor a sus autores (ver [6]).

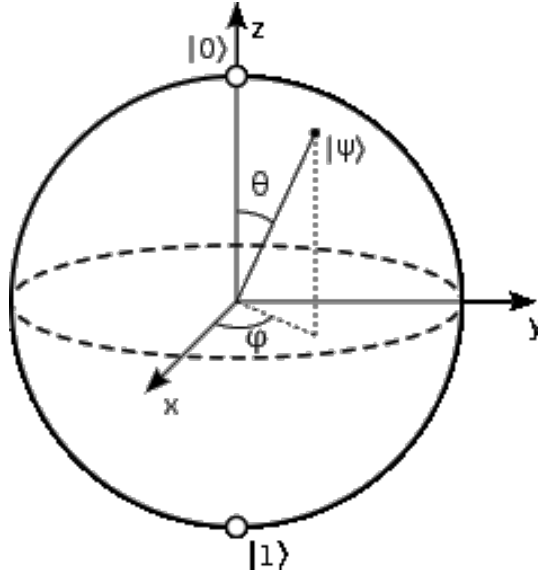
El algoritmo RSA consta de tres partes: generación de claves, cifrado y descifrado. La parte más importante se encuentra en la generación de claves, donde cada individuo debe elegir 2 números primos grandes (actualmente son del orden de 10^{200}) y que no cumplan ciertas características debilitantes (números a partir de los que resulta posible obtener la clave secreta a partir de la pública). Una vez generadas estas claves, los individuos usarán aritmética modular para realizar las operaciones de cifrado y descifrado.

1.3. Comienzos de la computación cuántica

Cuando hablamos de criptografía es casi imprescindible mencionar el trabajo de Alan Turing, considerado uno de los padres de la computación. Por un lado, realizó un trabajo muy importante durante la Segunda Guerra Mundial, ya que rompió el cifrado de la máquina Enigma que usaban los nazis. Por otro lado, la tesis Church-Turing que realiza la afirmación “*todo algoritmo es equivalente a una máquina de Turing*”. Y puesto que todo algoritmo es equivalente a una máquina de Turing, los algoritmos cuánticos también pueden serlo. En 1982, Richard Feynman mostró que una máquina de Turing que simulara fenómenos cuánticos tendría un coste exponencial. Esto es debido a los recursos necesarios para describir clásicamente un espacio de Hilbert cuántico. David Deutsch trabajó sobre estas ideas y, en 1985, describió el primer computador cuántico. Pero no sería hasta 1998 cuando apareció la primera computadora cuántica de 1 qubit, desarrollada por el grupo de Berkeley y dirigido por Isaac Chuang.

En la actualidad (2017) se manejan sistemas cuánticos cada vez más ambiciosos. Por ejemplo, la empresa IBM, junto con la Universidad de Stanford, implementó el algoritmo de Shor (ver Apéndice A.1) en el primer computador cuántico de 7 qubits. Asimismo, la Universidad de Innsbruck creó el primer qbyte con trampas de iones, y se están desarrollando los primeros buses y procesadores cuánticos, aunque parece que aún queda para que veamos estos computadores usándose de manera habitual (ver [7]).

Para poder entender la criptografía cuántica primero hay que hacer una breve introducción a la computación cuántica, comenzando por sus unidades de almacenamiento elementales: los qubits. Se denomina qubit a un sistema cuántico cuyo estado se describe como un punto en una esfera de radio unidad, que suele denominarse *Esfera de Bloch*. Toda operación unitaria sobre el qubit equivale a mover el estado en una trayectoria cualquiera sin salirse de la superficie de la esfera.



Un qubit se puede expresar como:

$$|\psi\rangle = \cos(\theta/2) |0\rangle + e^{i\phi} \sin(\theta/2) |1\rangle$$

Donde $\theta, \phi \in \mathbb{R}$ tal que $0 \leq \theta \leq \pi$ y $0 \leq \phi \leq 2\pi$.

También es posible expresar un qubit como combinación lineal de los estados $|0\rangle$ y $|1\rangle$ de la siguiente forma:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Donde α y β , denominados amplitudes del estado, son números complejos tal que $\alpha = r_\alpha e^{i\phi_\alpha}$, $\beta = r_\beta e^{i\phi_\beta}$. Sus cuadrados, α^2 y β^2 explicitan la probabilidad de que el qubit se “revele” como un 0 o un 1 (respectivamente) al realizar una medición del sistema.

La habilidad de los qubits para estar, antes de la medición, en un continuo de estados (superposición) entre $|0\rangle$ y $|1\rangle$ violenta nuestro sentido común. Y es importante que tengamos en mente que el conocimiento que se adquiere a partir de la medida está ligado a la pérdida de la superposición.

1.4. Criptografía cuántica

La mayoría de sistemas de cifrado actuales se basan en combinaciones numéricas con las que cifrar la información. Utilizan problemas matemáticos difíciles de resolver

por los ordenadores actuales, los cuales deben de ser lo suficientemente complejos como para garantizar que la información no pueda ser descifrada. Cuando pasamos a la computación cuántica aparecen dos características respecto a la criptografía.

En primer lugar, posibilita la realización de tareas exponenciales en un tiempo razonable. Esta característica, que parece ser tan ventajosa para los sistemas actuales, hace posible que podamos vulnerar la seguridad de los esquemas que utilizan en factorización de enteros (es decir, ligados al problema del logaritmo discreto). Esto se debe a que los sistemas criptográficos de los que disponemos se basan en la falta de velocidad de cálculo de los procesadores actuales.

No obstante, al comenzar a utilizar computación cuántica podemos desarrollar herramientas con propiedades muy atractivas en seguridad, como es el caso del protocolo BB84. Para ello se hace uso de propiedades de leyes físicas, lo cual es una gran ventaja sobre la criptografía convencional, que se basa en asunciones computacionales.

Uno de los teoremas de los que hace uso es el teorema del no-clonado, el cual asegura que no existe ningún procedimiento por el cual pueda copiarse un estado cuántico de un sistema a otro sistema idéntico. Este teorema es una consecuencia del famoso principio de incertidumbre de Heisenberg, por el cual se puede medir ya sea la posición de una partícula o su cantidad de movimiento, pero no ambas con una precisión arbitraria. Sin embargo, la física cuántica no prohíbe el “teletransporte” de un estado, el cual no es una clonación, ya que se pierde el estado original.

El teorema de no clonado se formuló por primera vez en 1982. La demostración sorprendió a la comunidad científica debido a su simplicidad, ya que no se entendía como pudo pasar inadvertido a tantas generaciones de físicos; sobre todo por sus importantísimas implicaciones y los grandes esfuerzos de investigación en esta temática que, a lo largo de la segunda mitad del siglo XX, se llevó a cabo por una gran cantidad de científicos. De hecho podemos ver una breve demostración de este teorema:

- Tenemos un sistema que queremos copiar, que viene dado por un ket de estado arbitrario $|\alpha\rangle$
- Ésto lo queremos copiar a un sistema B que esta inicializado con un ket de estado $|\beta\rangle$
- Definimos que el conjunto formado por ambos sistemas estará descrito mediante un ket que es el producto tensorial: $|\alpha\rangle \otimes |\beta\rangle$
- Consideremos una base de estado $|a_i\rangle$, a partir de la que descompondremos el estado a copiar:

$$|\alpha\rangle = \sum_i a_i |a_i\rangle$$

- El procedimiento de copiado vendrá representado mediante un operador U unitario y lineal. Para que el procedimiento de clonado funcione, el operador U tiene que ser capaz de duplicar los vectores de la base: $U|a_i\rangle \otimes |\beta\rangle = |a_i\rangle \otimes |a_i\rangle$

- Entonces, el resultado de aplicar el operador de clonado al estado completo quedará así:

$$U|\alpha\rangle \otimes |\beta\rangle = \sum_i a_i U|a_i\rangle \otimes |\beta\rangle = \sum_i a_i |a_i\rangle \otimes |a_i\rangle$$

- Por otra parte, el estado $|\alpha\rangle \otimes |\alpha\rangle$ se debería descomponer como sigue: $|\alpha\rangle \otimes |\alpha\rangle = \sum_{i,j} a_i a_j |a_i\rangle \otimes |a_j\rangle$. Lo cual no se corresponde con el anterior resultado.
- Por tanto, el procedimiento de clonado funcionará tan sólo si $a_i a_j = a_i \delta_{ij}$, lo cual, en general, no es cierto.

2. Protocolo BB84

2.1. Introducción

El protocolo BB84, publicado en 1984 por Charles Bennett y Gilles Brassard, supone el nacimiento de la criptografía cuántica (ver [4]). Con éste se describe cómo compartir una clave de cifrado entre dos individuos sin que un tercero pueda conocerla, lo cual se suele conocer como *Distribución de Clave Cuántica* (*Quantum Key Distribution* o *QKD* en inglés).

A diferencia de la criptografía clásica, cuando comenzamos a emplear la mecánica cuántica se pone de manifiesto el teorema del no-clonado que mencionamos anteriormente, lo que imposibilita que Eve sea capaz de realizar una copia de un estado cuántico desconocido. Esto supone una gran ventaja en temas de seguridad, ya que antes tan solo tenía que guardar una copia de los datos hasta que hubiese un algoritmo o hardware nuevo que fuese capaz de revelar el contenido de los datos. Por ejemplo, el gobierno canadiense almacena de forma secreta los datos del censo durante 92 años, por lo que en unas décadas podríamos ser capaces de resolver el problema de la factorización y violar las medidas de seguridad actuales.

Como veremos más adelante, una de las ventajas fundamentales de este protocolo es que si Eve quiere realizar un ataque *Man in the Middle* (un ataque muy común en la actualidad) al tratar de leer el estado cuántico lo altera, y al reenviarlo a Alice (o Bob) es detectado, por lo que descartarán la clave al haberse visto comprometida. En el caso clásico de un ataque de este tipo, el hecho de que Eve lea el canal no es detectable en la mayoría de casos.

2.2. Funcionamiento

Durante la siguiente sección describiremos el protocolo QKD de BB84 sobre canales sin ruido, y más adelante veremos la seguridad que aporta a las comunicaciones. Realmente, los canales están inevitablemente sometidos a ruidos, por lo que esta des-

criptación no es del todo realista. No obstante, se han realizado otras pruebas: Peter Shor y J. Preskill realizaron la primera prueba de la seguridad del protocolo BB84 en canales con ruido (ver [8]). Esta prueba se basa en la seguridad de otro protocolo QKD que, a su vez, confía en los *Códigos de Corrección de Errores Cuánticos* (en inglés *Quantum Error Correcting Codes*, tema principal de la *Teoría de Información Cuántica*).

El procedimiento del protocolo BB84 se divide en las siguientes dos fases:

1. Fase de comunicación cuántica

- Alice y Bob generan, cada uno, una secuencia aleatoria entre 2 bases: B_0 y B_1 . Los elementos de estas bases los podemos describir como rectilíneos para B_0 (horizontal y vertical) y diagonales para B_1 (45° ó 135°).

$$B_0 = \{|0\rangle, |1\rangle\} \text{ y } B_1 = \left\{ \frac{|0\rangle + |1\rangle}{2}, \frac{|0\rangle - |1\rangle}{2} \right\}$$

Además, Alice genera una secuencia aleatoria de bits que será usada en el siguiente paso.

- A partir de la secuencia aleatoria de bases y la de bits, Alice polariza los fotones y los envía a Bob. Por ejemplo, si la primera base es B_0 y el primer bit 0, el fotón sería polarizado a partir del primer elemento de la base B_0 . Si el bit hubiese sido 1, el fotón se habría polarizado en función del segundo elemento de la base (resp. para B_1).
- Bob recibe los fotones polarizados y, a partir de la secuencia aleatoria de bases que ha generado él, realiza la medición de los mismos.

2. Fase de discusión pública

- Una vez enviados todos los fotones, cada participante comparte con el otro la secuencia de bases que ha usado.
- Se rechazan aquellos fotones que hayan sido medidos con una base distinta. En promedio se rechazarán la mitad.
- Ahora, para verificar que no haya habido interferencias en el canal, o incluso que haya un tercero manipulando la conversación, Alice manda a Bob una fracción del resultado de la medición para comprobar que los valores coinciden (Bob hace lo mismo).
- Alice y Bob comprueban la tasa de error. Si la tasa de error es superior a un cierto umbral se aborta la comunicación y se vuelve a empezar.
- Pueden repetir este proceso varias veces para llegar a tener una clave de mayor longitud. Asimismo, pueden aplicar técnicas de post-procesamiento clásicas, sobretudo para la corrección de errores, para llegar a una clave final.

Bits aleatorios de Alice	0	1	1	0	1	0	0	1
Bases aleatorias de Alice	+	+	x	+	x	x	x	+
Fotones enviados por Alice	↑	→	↘	↑	↘	↗	↗	→
Bases aleatorias de Bob	+	x	x	x	+	x	+	+
Fotones medidos por Bob	↑	↗	↘	↗	→	↗	→	→
Bits medidos por Bob	0	0	1	1	0	0	1	1
Clave compartida	0	-	1	-	-	0	-	1

Hay que remarcar que Alice y Bob únicamente comparten la secuencia de bases que ha usado cada uno después de haber enviado la clave, ya que si no estarían mandando información útil a Eve.

No obstante, la parte más complicada de este protocolo se da en el post-procesamiento clásico para generar la clave final. Por supuesto, *a priori*, dado un procedimiento particular para un post-procesamiento clásico es muy complicado saber si la clave será segura o qué clave va a generar. Afortunadamente, hay determinadas pruebas de seguridad que proporcionan a Alice y Bob una idea sobre qué protocolo usar. El protocolo clásico de post-procesamiento a menudo consiste en una *prueba de manipulación* y en la *generación de la clave*.

En la *prueba de manipulación*, Alice y Bob pueden elegir aleatoriamente una fracción de las señales para la prueba. Por ejemplo, al transmitir las polarizaciones de esas señales, pueden calcular la tasa de error de bit de las señales de prueba. Si la tasa de error de bit de la señal probada es mayor que un valor umbral prescrito, Alice y Bob abortan. Por otro lado, si la tasa de error de bit es menor o igual al valor prescrito, continúan con el paso de *generación de la clave* con las señales restantes.

Los pasos para la *generación de la clave* se pueden dividir en las siguientes etapas:

1. Pre-procesamiento clásico

A pesar de que se trata de un paso opcional, usarlo nos proporciona una mayor tolerancia a errores de bit. Para ello se puede usar algún tipo algoritmo de detección de errores o algún proceso aleatorio. Un ejemplo de algoritmo de detección (y corrección) de errores es *B-step*, donde Alice permuta aleatoriamente todos sus bits y envía a Bob la paridad de cada par adyacente.

Por ejemplo, partiendo de $\vec{x} = (x_1, x_2, \dots, x_{2N})$, Alice crea la cadena $\vec{x}_1 = (x_{\sigma(1)} + x_{\sigma(2)} \bmod 2, x_{\sigma(3)} + x_{\sigma(4)} \bmod 2, \dots, x_{\sigma(2N-1)} + x_{\sigma(2N)} \bmod 2)$, donde σ es una permutación aleatoria elegida por Alice. Además, Alice informará a Bob cuál ha sido la permutación elegida. De manera similar, Bob hará el mismo proceso con \vec{y} , usando la misma permutación σ , y le enviará el resultado de la permutación a Alice. Es decir, Bob le envía a Alice la siguiente secuencia: $\vec{y}_1 = (y_{\sigma(1)} + y_{\sigma(2)} \bmod 2, y_{\sigma(3)} + y_{\sigma(4)} \bmod 2, \dots, y_{\sigma(2N-1)} + y_{\sigma(2N)} \bmod 2)$. Una vez hecho esto, Alice y Bob mantendrán el primer bit si y sólo si las paridades del par coinciden. Por ejemplo, si se da que $x_{\sigma(2k-1)} + x_{\sigma(2k)} \bmod 2 =$

$y_{\sigma(2k-1)} + y_{\sigma(2k)} \bmod 2$, entonces Alice mantiene $x_{\sigma(2k-1)}$ (Bob equivalentemente para $y_{\sigma(2k-1)}$) como su nuevo bit. En caso contrario, ambos desechan los pares $(x_{\sigma(2k-1)}, x_{\sigma(2k)})$ y $(y_{\sigma(2k-1)}, y_{\sigma(2k)})$ respectivamente.

Como resultado de este proceso de detección de errores, la probabilidad de error se reduce de $O(p)$ a $O(p^2)$, donde p es la probabilidad de que un evento $x_i \neq y_i$ ocurra.

2. Corrección de errores

Las claves x e y de Alice y Bob pueden ser diferentes, así que es necesario reconciliarlas. Por ejemplo, un método simple sería dejar que Alice mantuviese su clave x y que Bob modificase y por x .

El paso de reconciliación es muy importante en el protocolo BB84 ya que aquí es donde se calculará la tasa de error y se podrá detectar la presencia de un agente externo. No obstante, los errores pueden proceder de distintas fuentes, incluyendo hardware no-ideal, ruido ambiental y, por supuesto, Eve. En principio, para garantizar la seguridad, podemos asumir que todos los errores han sido provocados por Eve.

Alice y Bob realizan una estimación de la tasa de error. Para ello toman un subconjunto de bits de la clave y los comparan (debemos tener en cuenta que esto se hace por un canal público). Si todos los bits son iguales quiere decir que ambos tienen la misma versión de la clave, lo cual lo pueden verificar a través de un hash. Por supuesto, los bits que se han usado durante esta comparación serán automáticamente descartados en la clave final, antes de aplicar la función hash.

Si hay errores en la clave, entonces tendremos que identificarlos y corregirlos. Esta reconciliación requiere de compartir información a través de canales públicos, por lo que “filtra” información sobre la clave. Este es el principal motivo por el que el método usado en la reconciliación de información es una elección crítica de diseño. Algunos de los protocolos de reconciliación más comunes son: *Cascade*, *Winnow* y *LDPC* (*Low Density Parity Codes*).

Las principales características de estos métodos son las siguientes:

- **Cascade:** Es muy similar al preprocesamiento que hemos visto. A modo resumido, el proceso de *Cascade* divide la clave en bloques y calcula la paridad de cada uno de ellos, pero no es capaz de detectar cantidades pares de errores. Es decir, solo corrige errores en bloques que tengan más de 2 errores y con paridad impar. En conclusión: solo es capaz de reducir (exponencialmente) la cantidad de errores que hay en la clave.
- **Winnow:** Este método se basa en multiplicaciones de matrices para corregir los errores. De forma similar a *Cascade*, se divide la clave en bloques, pero la comparación de paridades también se hace a través de matrices. Sin embargo, *Winnow* tiene un problema: puede introducir nuevos errores si la cantidad de errores por bloque es demasiado alta. Afortunadamente, el autor original de este método (Buttler) propone una permutación inicial para evitar este problema (ver [14]).

- **LDPC:** Algunas de las grandes restricciones de los 2 métodos anteriores es que solo son capaces de corregir 1 error por bloque, y tienen que compartir multitud de información sobre paridades hasta llegar a una clave común. Así que otra alternativa es *LDPC*, el cual es un proceso más complejo que los dos anteriores (necesita una mayor capacidad de computación y de memoria), pero es capaz de corregir todos los errores de la clave con un solo intercambio de información.

3. Amplificación de privacidad

El último paso, después de corregir los errores, se conoce como *Amplificación de privacidad*. El objetivo de este paso es eliminar cualquier información expuesta durante la fase de reconciliación y garantizar que Eve no obtenga suficiente información hasta el punto de poder reconstituir una parte significativa de ella.

Después de la reconciliación de errores y el mantenimiento de la privacidad, Alice y Bob pueden estar seguros de que tienen la misma versión de la clave y de que los bits obtenidos por Eve se han reducido al mínimo.

Como una medida de seguridad final, Alice y Bob discuten sobre el canal clásico la selección de una función hash universal. Esta función hash, que reduce aún más el tamaño de la clave final, se aplica a la clave reconciliada. De esta forma, incluso si Eve fuera una espía muy eficiente y tuviese una copia razonable de la clave, calcular este hash amplificaría cualquier error en la versión de Eve, ya que pequeños cambios en la entrada de una función hash provocan grandes cambios en la salida. Por lo tanto, incluso si Eve solo tenía algunos errores en su versión de la clave, después de la amplificación de privacidad sus errores se amplificarían hasta el punto en que incluso una búsqueda de fuerza bruta sería inviable.

A modo de apunte, una función hash universal es aquella en la que la probabilidad de colisión entre dos textos diferentes es despreciable (ver [10]).

2.3. Seguridad del protocolo

Para ilustrar la seguridad de este algoritmo pongamos un ejemplo sencillo de un ataque MitM (del inglés, *Man in the Middle*): supongamos que Eve intercepta cada uno de los fotones que envía Alice. Eve no conoce las bases usadas por Alice, por lo que usará a su suerte una secuencia aleatoria para medir los fotones. Después, Eve reenvía a Bob los fotones que acaba de medir, el cual volverá a medirlos usando otra secuencia aleatoria de bases. Como se han realizado 2 mediciones con bases aleatorias (y tenemos dos posibles bases: + ó x) tendremos que, en promedio, habrá una tasa del 25 % de error. Evidentemente, esto es detectable por Alice y Bob.

Como hemos mencionado, la conexión se aborta si la tasa de error es demasiado alta. Así que si tenemos un grupo de espías interceptando nuestras comunicaciones las 24h no seremos capaces de acordar una clave común, pero al menos sabremos que hay

alguien ahí. El bloqueo de la comunicación es una de las críticas que ha recibido BB84 a lo largo del tiempo.

Otras consideraciones que tenemos que tener en cuenta es que, a pesar de tener presente el teorema de no-clonado, los emisores de fotones pueden tener fallos que permitan que Eve sea capaz de detectar cuándo se envían pares de fotones, de manera que ella se queda uno y el otro lo deja pasar. Esto es totalmente indetectable para las partes que están estableciendo una comunicación, ya que se trata de partículas distintas, pero de esto hablaremos más en detalle en la siguiente sección.

2.4. Limitaciones e implementaciones

Si suena demasiado bien para ser verdad es bastante probable que no lo sea. En el protocolo BB84 se asumen varias cosas:

- Alice emite un único fotón cada vez.
- El canal entre Alice y Bob puede tener ruido pero no hay pérdida de información.
- Bob tiene detectores de fotones perfectamente eficientes.
- La alineación de la base entre Alice y Bob es perfecta.

Si se dan estas cuatro condiciones, entonces nos encontramos que el protocolo es incondicionalmente seguro al intercambio de claves tal y como se ha mostrado en diferentes pruebas matemáticas (como la mencionada de Shor y Preskill en el año 2000, ver [8]).

Sin embargo, muchas de estas condiciones no son válidas en los sistemas del mundo real. Por ejemplo, el protocolo confía en la transmisión de un único fotón cada vez, ya que si se mandase duplicado Eve tendría la oportunidad de interceptar uno de ellos y medirlo sin que el otro se viese afectado. Puesto que son fotones totalmente independientes, ni Alice ni Bob se darán cuenta de este duplicado y proseguirán con el procedimiento del protocolo y Eve podrá tener la oportunidad de medirlo más adelante.

Hoy en día, generar fotones *on-demand* de forma confiable no es práctico, así que en su lugar se genera y se atenúa un fotón y, en promedio, solo hay 0,1 fotones por paquete (es decir, 1 de cada 10 paquetes tendrá un duplicado). Así se reduce significativamente la eficiencia del protocolo, pero es necesario para limitar el conocimiento que adquiere Eve.

En el lado del receptor, los fotones deben ser detectados con precisión. Desafortunadamente, los detectores de fotones tienen un ratio de detección limitado, pueden

tener una baja eficiencia en la detección y pueden dar falsas mediciones. Incluso cuando Eve no está presente, las características físicas del canal cuántico pueden introducir errores que afecten a la polarización de los fotones cuando están viajando por él. El resultado de estas limitaciones técnicas es que en la clave final se introducen errores incluso cuando no hay ningún agente externo malicioso. Por supuesto, hay que ser capaz de corregir estos errores antes de usar un algoritmo criptográfico, ya que ambas partes tienen que tener copias idénticas de la clave.

Después de que saliese a la luz el protocolo BB84, con la posible existencia de un método incondicionalmente seguro para compartir claves, surgieron multitud de investigaciones con formas de atacar el protocolo. Así surgió un nuevo ámbito: *Hacking Cuántico*. Los creadores del protocolo BB84 dieron algunas formas para reducir las vías por las que se podía obtener información de la clave y, por tanto, mitigar la información filtrada. Otros investigadores propusieron nuevos protocolos que minimizaran los problemas del BB84, pero introducían otros nuevos o necesitaban dispositivos hardware con las mismas limitaciones.

Actualmente hay numerosas líneas de investigación en este campo. Algunas de ellas han conseguido buenos resultados generando claves para distancias de unos 50km en fibra óptica. No obstante, los cables de fibra óptica tienen una atenuación según la distancia, por lo que es esta pérdida lo que no nos permite crear conexiones seguras de punto a punto en distancias superiores a unas pocas decenas de kilómetros.

Debido a esto, uno de los mayores retos en rendimiento y coste es el desarrollo del hardware necesario para la generación de claves. A parte del canal de comunicación, es crucial que los detectores de fotones sean precisos. Los últimos desarrollos para sistemas QKD están mostrando resultados bastante esperanzadores, con detectores que alcanzan una eficiencia del 93 %.

3. Esquemas basados en retículos

3.1. Introducción a esquemas postcuánticos

A pesar de que podamos duplicar o triplicar la velocidad de un ordenador actual con computación cuántica, conocemos determinados algoritmos cuyo tiempo de ejecución no se vería significativamente alterado con el conocimiento actual sobre algoritmos cuánticos. Aquí encontramos ciertos esquemas criptográficos que podemos implementar hoy en día, y que nos ofrecen alguna esperanza de que el mundo pueda hacer la transición a sistemas cuánticos sin necesidad de anticipar un cambio de paradigma radical. Tan solo tendríamos que migrar a esquemas implementables en la actualidad cuyos problemas matemáticos subyacentes no se resuelven significativamente más rápido con algoritmos cuánticos.

Algunos algoritmos criptográficos que son candidatos prometedores están basados en retículos o, también llamados, enrejados (*Lattice-based*): *Learning con errores*, *Ring-Learning con errores* ó *Ring-Learning con errores de firma*. Algunos de estos esquemas, como el cifrado NTRU (ver [11]), han sido estudiados durante muchos años sin que nadie encuentre un ataque factible. Pero existen más tipos de algoritmos; enumeramos a continuación algunos:

- Criptografía basada en retículos, de la cual acabamos de hablar.
- Criptografía de ecuaciones cuadráticas y multivariantes. Normalmente solo se admite en firmas. Un ejemplo es el esquema Rainbow.
- Criptografía basada en hash (*hash-based* en inglés). Incluye sistemas criptográficos como las firmas Lamport y el esquema de firmas Merkle (ver [12]). Concretamente, estas últimas, creadas en 1970, son de especial importancia como posibles sucesores a las firmas digitales RSA y DSA actuales. Perdió importancia en su momento ante las firmas RSA, pero luego volvió a tomar mayor renombre al ser un sistema de firmas resistente a los ataques cuánticos.

La criptografía moderna, en la mayoría de casos, aún no usa estos algoritmos. La

mayoría de los esquemas criptográficos post-cuánticos son simplemente demasiado nuevos como para confiar la seguridad de los sistemas en ellos, como es el caso de la criptografía basada en retículos. Con un poco de suerte y de preparación, los expertos en seguridad realizarán el cambio a estos algoritmos antes de que los ordenadores cuánticos evolucionen más. Sin embargo, si fallan las consecuencias pueden ser nefastas.

En este sentido, el NIST (siglas de *National Institute of Standards and Technology*) está realizando esfuerzos en establecer estándares para estos algoritmos. Para ello ha abierto un concurso en el que se solicita y evalúan el estándar propuesto por los diferentes participantes para uno o más algoritmos post-cuánticos. Esta iniciativa del NIST ha surgido por los grandes avances que se están dando en el campo de la computación cuántica. Hace unos años, plantearse construir computadores cuánticos a gran escala era poco factible. No obstante, hoy en día hay numerosos ingenieros que predicen que en los próximos 20 años puede que tengamos computadores cuánticos entre nosotros que sean capaz de romper los esquemas de clave pública.

3.2. Qué es la criptografía basada en retículos

Cuando hablamos de retículos nos referimos a un subconjunto discreto de vectores. Decimos que un retículo tiene dimensión n si no está contenido en ningún espacio propio de \mathbb{R}^n . Visualmente, un retículo es un conjunto de puntos situados regularmente a lo largo de un espacio. Como en los sistemas algebraicos, un retículo es un grupo Abelian libre generado libremente.

Denominamos base de un retículo L a un conjunto B de vectores (ver definición en el Apéndice A.2) tal que cualquier punto de L puede ser expresado como una combinación lineal (con coeficientes enteros) de elementos de B (ver Apéndice A.2). Si la dimensión del retículo L es por lo menos 2, siempre habrá infinitas bases diferentes en el retículo. No obstante, el texto plano, el texto cifrado y la clave deben tener un tamaño finito, por lo que los retículos criptográficos no son subconjuntos de \mathbb{R}^n sino de \mathbb{K}^n , donde \mathbb{K} es un cuerpo finito. En otras palabras, para cualquier base de L , el conjunto de todas las combinaciones lineales con coeficientes enteros de los vectores de la base forman un retículo.

La seguridad de un sistema criptográfico basado en retículos se reduce a dos problemas combinatorios:

- El problema del vector más corto: encontrar el vector más corto en un retículo L .
- El problema del vector más cercano: dada una base B de L y un vector $v \notin L$, encontrar el vector $v' \in L$ que está más cerca de v .

Se cree que estos problemas son difíciles para los casos generales, como la mayoría de las bases B . Pero si los vectores de la base son cortos y *casi* ortogonales, entonces ambos problemas se vuelven resolubles. A este proceso de encontrar una base en un retículo se le denomina *reducción de base del retículo* (*lattice basis reduction* en inglés), y el algoritmo más famoso para esto es el de Lenstra-Lenstra-Lovász (también llamado LLL).

Por último, cabe mencionar que los sistemas criptográficos basados en retículo se dividen en dos grupos:

- Eficientes, pero probablemente no seguros. Su eficiencia es comparable al algoritmo mejor conocido. En este grupo encontramos, por ejemplo, el algoritmo NTRU, uno de los sistemas criptográficos más famosos en este campo. Éste se divide en dos algoritmos: *NTRUEncrypt*, usado en el cifrado de datos, y *NTRUSign*, para realizar firmas digitales. Además de lo mencionado previamente, este sistema tiene otra ventaja respecto a RSA en el coste de generación de claves privadas. Mientras que las operaciones de RSA se incrementan cúbicamente según aumenta el tamaño de la clave, en NTRU aumenta cuadráticamente.
- Probablemente seguros, pero menos eficientes. Por ejemplo, los algoritmos basados en el problema combinatorial de *Aprendizaje con errores* (*Learning with errors* en inglés). En la siguiente sección hablaremos en más profundidad de un algoritmo de este tipo.

3.3. Ring-Learning con errores

También llamado RLWE (*Ring-Learning with errors*), es un problema computacional que sirve de base para nuevos algoritmos criptográficos diseñados para protegerse ante computadores cuánticos y, también, proporcionan la base del cifrado homomórfico.

3.3.1. Notaciones previas

Antes de dar algunas definiciones que usaremos para describir el funcionamiento del protocolo, será necesario conocer los elementos con los que trabajaremos:

- El problema RLWE se construye sobre la aritmética de polinomios con coeficientes de un cuerpo finito (o cuerpo de Galois). El típico polinomio $p(x)$ lo podemos expresar como:

$$p(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}$$

- En el contexto de RLWE, notaremos como \mathbf{Z} al anillo de enteros racionales, y a $R = \mathbf{Z}[X]/\Phi_m(X)$ al anillo de enteros del cuerpo ciclotómico m , es decir, $\Phi_m \in \mathbf{Z}[X]$ es el polinomio ciclotómico m . A lo largo de este capítulo, nos limitaremos al caso de que m es una potencia de 2. Esto quiere decir que $\Phi_m(X) = X^n + 1$ para $n = 2^l$, $l > 0$ y $m = 2n$.
 - Sea q un entero, y sea $R_q = R/qR \cong \mathbf{Z}_q[X]/(X^n + 1)$ con $\mathbf{Z}_q = \mathbf{Z}/q\mathbf{Z}$.
 - Si χ es una distribución sobre R , entonces $x \xleftarrow{\$} \chi$ es un muestreo de $x \in R$ según χ . La distribución χ será, normalmente, una distribución discreta Gaussiana sobre R .
 - Sea S un conjunto y $\mathcal{U}(S)$ la distribución uniforme sobre dicho conjunto, entonces definimos como $x \xleftarrow{\$} S$ a un muestreo uniforme y aleatorio.
 - Si \mathcal{A} es un algoritmo probabilístico, $y \xleftarrow{\$} \mathcal{A}(x)$ significa que ejecuta el algoritmo \mathcal{A} sobre una entrada x y asigna el resultado a y .
 - $\text{Adv}_Y^{\text{xxx}}(\mathcal{A})$ es la ventaja de un algoritmo \mathcal{A} en romper la noción de seguridad xxx de un esquema o protocolo Y .
- En criptografía, la ventaja de un adversario se usa como medida sobre con que efectividad se podría atacar un algoritmo criptográfico.

En los siguientes apartados haremos uso de diferentes definiciones y lemas de los que podemos ver explicaciones detalladas en el Apéndice A.2.

Otro de los conceptos que es necesario en el problema RLWE es la idea de generar polinomios pequeños con respecto a alguna norma. Normalmente se usa la norma infinita, ó también llamada norma uniforme. La norma infinita de un polinomio es simplemente el coeficiente más grande del polinomio cuando estos se ven como enteros. Así pues, si tenemos que $\|a(x)\|_\infty = b$ quiere decir que la norma infinita del polinomio $a(x)$ es b , donde b será el coeficiente más grande de $a(x)$.

3.3.2. El problema RLWE

El problema de RLWE se puede separar en dos partes: el problema de *búsqueda* y el problema de *decisión*. Para describir ambos problemas vamos a necesitar:

- $a_i(x)$ un conjunto aleatorio de polinomios conocidos de R_q , con coeficientes de $\mathbf{Z}_q[X]$
- $e_i(x)$ un conjunto aleatorio de polinomios desconocidos en relación a un límite b en el anillo $\mathbf{Z}_q[X]/\Phi(X)$
- $s(x)$ un polinomio desconocido en relación a un límite b en el anillo $\mathbf{Z}_q[X]/\Phi(X)$

$$\blacksquare \quad b_i(x) = (a_i(x) \cdot s(s)) + e_i(x) \in R_q \times R_q$$

Así pues, por un lado, el problema de *búsqueda* se centra en hallar el polinomio desconocido $s(x)$ a partir de una lista de pares de polinomios $(a_i(x), b_i(x))$.

Por otro lado, el problema de *decisión* se basa en determinar si, a partir de una lista $(a_i(x), b_i(x))$, los polinomios $b_i(x)$ se han construido con $b_i(x) = (a_i(x) \cdot s(s)) + e_i(x)$ o han sido generados aleatoriamente en $\mathbf{Z}_q[X]/\Phi(X)$ con coeficientes de $\mathbf{Z}_q[X]$.

La dificultad de este problema es parametrizarlo a partir del polinomio cociente $(\Phi(X))$, su grado n , el cuerpo \mathbf{Z}_q y el límite más pequeño b . En muchos de los algoritmos RLWE basados en clave pública, la clave privada será el par formado por los polinomios $s(x)$ y $e(x)$, y la correspondiente clave pública estará formada por los pares de polinomios $a(x)$ (tomado aleatoriamente de $\mathbf{Z}_q[X]/\Phi(X)$) y el polinomio $b(x)$, donde $b(x) = (a(x) \cdot s(x)) + e(x)$. Entonces, dados $a(x)$ y $b(x)$ no debe ser computacionalmente factible poder recuperar el polinomio $s(x)$. En el caso en el que nos encontramos, en el que $\Phi(x)$ es un polinomio ciclotómico, la dificultad de resolver el problema de búsqueda se reduce a encontrar un vector corto (pero no necesariamente el más corto) en un retículo formado por elementos de $\mathbf{Z}[X]/\Phi(X)$, representado como un vector de enteros (conocido como el problema del vector aproximado más corto).

Una de las mayores ventajas que tiene la criptografía basada en RLWE sobre la que está basada en el LWE original se encuentra en el tamaño de las claves pública y privada. Por ejemplo, para un nivel de seguridad de 128 bits, un algoritmo criptográfico RLWE usaría una clave pública de unos 7000 bits de longitud, mientras que el correspondiente algoritmo en LWE necesitaría 49 millones de bits. No obstante, las claves RLWE son más largas que las que se usan actualmente en otros algoritmos, como el RSA o el de curva elíptica de Diffie-Hellman, que requieren claves públicas de 3072 y 256 bits respectivamente.

3.3.3. Diffie-Hellman-like basado en RLWE

En este apartado describiremos un protocolo *Diffie-Hellman-like* de intercambio de claves basado en el problema de RLWE. Para simplificar llamaremos a *Diffie-Hellman-like* como DHI.

Para poder obtener un protocolo de intercambio de clave exacto deberemos aplicar ciertas correcciones de errores al cálculo de la clave por parte de Alice. Para ello, emplearemos los mecanismos de corrección de errores de Peikert (usaremos las definiciones y lemas descritas en el Apéndice A.2).

Uno de los aspectos que más debemos tener en cuenta es que la seguridad final del algoritmo dependerá de la capacidad de generar polinomios aleatorios que, además, sean pequeños respecto a la norma infinita. Para un polinomio, la norma infinita es simplemente el valor del coeficiente más grande cuando éstos se toman en \mathbf{Z} en lugar de

\mathbf{Z}_q (es decir, el conjunto $\{-(q-1)/2, \dots, 0, \dots, (q-1)/2\}$). Para llegar a este polinomio, existen dos métodos bastante comunes:

- *Distribución uniforme.* Los coeficientes de los polinomios son obtenidos a través de una distribución uniforme de un conjunto de coeficientes pequeños. Sea b un entero mucho más pequeño que q , entonces si tomamos coeficientes aleatoriamente de $\{-b, -b+1, \dots, 0, \dots, b-2, b-1, b\}$, el polinomio será pequeño respecto al límite b .
- *Distribución discreta Gaussiana.* Para un valor impar de q , los coeficientes son obtenidos aleatoriamente de un conjunto $\{-(q-1)/2, \dots, 0, \dots, (q-1)/2\}$, cuya media será 0 y desviación típica σ .

Parámetros públicos

Decisión de los parámetros q, n, χ

$$a \xleftarrow{\$} \mathcal{U}(R_q)$$

Alice

$$s, e \xleftarrow{\$} \chi$$

$$b \leftarrow as + e \in R_q$$

Bob

$$s', e' \xleftarrow{\$} \chi$$

$$\xrightarrow{b} b' \leftarrow a's' + e' \in R_q$$

$$e'' \xleftarrow{\$} \chi$$

$$v \leftarrow bs' + e'' \in R_q$$

$$\bar{v} \xleftarrow{\$} \text{dbl}(v) \in R_{2q}$$

$$\xleftarrow{b', c} c \leftarrow \langle \bar{v} \rangle_{2q, 2} \in \{0, 1\}^n$$

$$k_A \leftarrow \text{rec}(2b's, c) \in \{0, 1\}^n$$

$$k_B \leftarrow \lfloor \bar{v} \rfloor_{2q, 2} \in \{0, 1\}^n$$

Cuadro 3.1: Protocolo DH1 sin autenticar de intercambio de clave basado en RLWE

Usar un protocolo del tipo de DH1 tiene bastantes ventajas. Una de ellas es que la mayoría de las redes actuales usan protocolos como TLS, que está basado en DH (*Diffie-Hellman*). Otros sistemas criptográficos se construyeron a partir de asunciones de DH.

3.3.4. Seguridad de RLWE

La seguridad de la criptografía moderna, en particular de la criptografía de clave pública, se ha basado en la presumible intratabilidad de resolver determinados problemas matemáticos. Esto ha sido así si el tamaño del problema es suficientemente largo y se introducen componentes aleatorios. El ejemplo clásico que se ha usado desde los años 70 es el problema de factorización, el cual se cree que es computacionalmente intratable si los números primos son suficientemente largos y aleatorios. Este es el caso

del algoritmo RSA, pero como vimos anteriormente, si usamos el algoritmo de Shor (ver Apéndice A) podríamos llegar a obtener los dos números primos iniciales.

Martin R. Albrecht publicó en 2015 un documento donde explicaba algunos scripts Sage para calcular el tiempo de ejecución de diferentes algoritmos para resolver clásicamente el problema de LWE. Con estos parámetros, el mejor ataque clásico conocido consiste en resolver LWE por BDD (problema de decodificación de distancia acotada, en inglés *Bounded Distance Decoding*) reduciendo BDD a uSVP (problema del vector más corto, en inglés *Unique Shortest Vector Problem*) usando la técnica de Kannan. El tiempo de ejecución calculado por Albrecht es de al menos 2^{163} operaciones.

Para un ataque cuántico, el problema se vuelve algo más difuso. Por un lado, encontramos que el algoritmo de búsqueda de Grover tiene una complejidad cuadrática para el problema de búsqueda, pero este no es necesariamente el algoritmo que reduce la seguridad. Por ejemplo, Laarhoven describe un algoritmo cuántico para encontrar el vector más corto en un tiempo de $2^{1,799n+O(n)}$ (comparado con el mejor algoritmo clásico conocido que tarda $2^{2,465n+O(n)}$).

3.3.5. Implementación en Python

Para la implementación descrita a continuación se han tomado los parámetros $n = 1024$, $q = 2^{32} - 1$, $\sigma = 8/\sqrt{2\pi}$. Esta elección nos proporciona una seguridad de al menos 128 bits contra algunos ataques conocidos sobre algoritmos criptográficos basados en retículos, con una ventaja menor a 2^{-128} sobre adversario no cuánticos.

En el código se puede ver cómo se generan polinomios de forma aleatoria: el parámetro A de manera uniforme y el resto de polinomios usados con una distribución Gaussiana. Esta es una de las partes más críticas del algoritmo, ya que si, por ejemplo, estuviésemos obteniendo los coeficientes en base a un conjunto predefinido, tendríamos un serio problema de seguridad: la obtención de los polinomios secretos s y e se volvería un problema computable.

Para el cálculo de los polinomios y las operaciones sobre ellos he usado la librería *numpy*, la cual ya incluye funciones para generar polinomios siguiendo diferentes distribuciones (en este caso solo me interesaban las distribuciones uniforme y Gaussiana) y las operaciones de suma, resta y multiplicación de polinomios. Asimismo, también tiene otras operaciones, concretamente las de redondeo, que han sido necesarias para poder completar el algoritmo.

```

Administrador: Windows PowerShell
PS F:\Documentos\URJC\TFG MAT\python> python .\rlwe-kex-dimas.py

-Params---
n: 1024
q: 4294967295
A: 1024 | [3.27245288e+09 2.60239488e+09 1.89234996e+09 ... 3.76815300e+09
1.36009395e+09 3.25183462e+08]

-Alice---
s: 1024 | [-1. 3. -4. ... 4. 1. 4.]
e: 1024 | [ 1. -2. -11. ... -2. -7. 1.]
b: 2047 | [1.02251442e+09 2.91999645e+09 1.41495775e+09 ... 5.53570636e+08
1.47059198e+09 1.30073385e+09]

-Bob---
s': 1024 | [-1. 0. -2. ... 0. 2. -6.]
e': 1024 | [ 0. -4. -7. ... -3. -7. -5.]
b': 2047 | [1.02251442e+09 1.69257241e+09 1.52678867e+08 ... 1.58610636e+09
1.07973780e+09 2.34386652e+09]
e'': 1024 | [-2. -2. 3. ... -4. 3. 2.]

-Computed---
v: 3070 | [3.27245287e+09 1.37497084e+09 8.34980707e+08 ... 3.91472743e+09
2.36785044e+09 7.85531502e+08]
dbl_v: 3070 | [6.54490575e+09 2.74994168e+09 1.66996141e+09 ... 7.82945486e+09
4.73570087e+09 1.57106300e+09]
c: 3070 | [1. 1. 0. ... 1. 0. 0.]
recA: 3070 | [6.54490575e+09 2.74994169e+09 1.66996138e+09 ... 7.82945486e+09
4.73570087e+09 1.57106300e+09]

Keys---
Ka: 3070 | [0. 1. 0. ... 0. 1. 0.]
Kb: 3070 | [0. 1. 0. ... 0. 1. 0.]
Equals: 100.0 %
PS F:\Documentos\URJC\TFG MAT\python>

```

Figura 3.1: Ejecución del algoritmo con los parámetros descritos

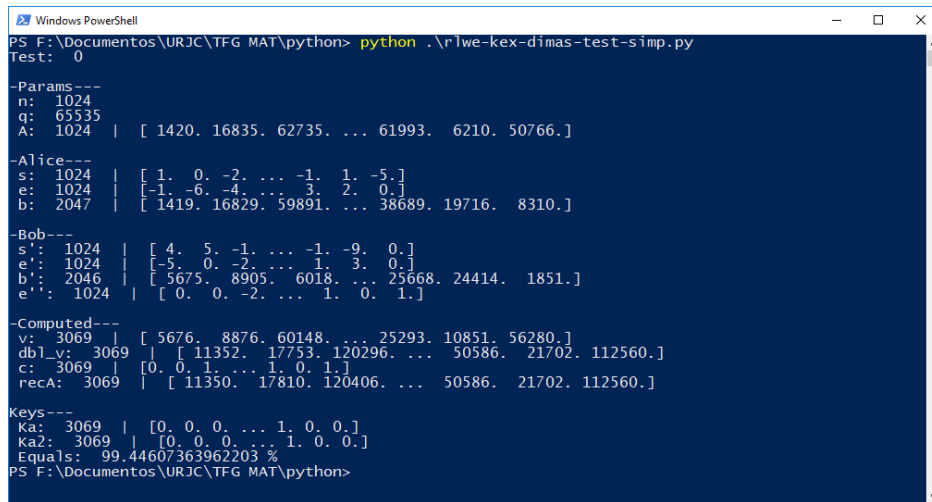
Otro de los puntos interesantes se da en el momento de calcular la clave K . La función rec tiene dos parámetros: $2b's$ (al que llamaremos $recA$) y c . Podemos apreciar que el valor que tiene Alice para $recA$ parece ser el mismo que el de Bob para dbl_v , donde $dbl_v = \langle \bar{v} \rangle_{2q,2}$. Esto puede inducir a error a la hora de implementar el algoritmo, porque no son exactamente iguales.

Alice	Bob
$s, e \xleftarrow{\$} \chi$	$s', e' \xleftarrow{\$} \chi$
$b \leftarrow as + e \in R_q$	$\xrightarrow{b} b' \leftarrow a's' + e' \in R_q$
	$e'' \xleftarrow{\$} \chi$
	$v \leftarrow bs' + e'' \in R_q$
	$\xleftarrow{b'} \bar{v} \xleftarrow{\$} \text{dbl}(v) \in R_{2q}$
$k_A \leftarrow \lfloor 2b's \rfloor_{2q,2} \in \{0, 1\}^n$	$k_B \leftarrow \lfloor \bar{v} \rfloor_{2q,2} \in \{0, 1\}^n$

Cuadro 3.2: Variación errónea del protocolo DHI

Si ejecutamos el algoritmo y comparamos la clave generada, por parte de Alice, con la función de reconciliación y con la de redondeo modular vemos que hay una similitud superior al 99 %. Así pues, tal y como podemos ver en el *lema 2* del *Apéndice A.2*, es necesario el cálculo de c para que Alice pueda calcular correctamente la clave.

$$\text{rec}(2b's, c) \neq \langle 2b's \rangle_{2q,2}$$



```

Windows PowerShell
PS F:\Documentos\URJC\TFG MAT\python> python .\rlwe-kex-dimas-test-simp.py
Test: 0

-Params--
n: 1024
q: 65535
A: 1024 | [ 1420. 16835. 62735. ... 61993. 6210. 50766.]

-Alice---
s: 1024 | [ 1. 0. -2. ... -1. 1. -5.]
e: 1024 | [-1. -6. -4. ... 3. 2. 0.]
b: 2047 | [ 1419. 16829. 59891. ... 38689. 19716. 8310.]

-Bob---
s': 1024 | [ 4. 5. -1. ... -1. -9. 0.]
e': 1024 | [-5. 0. -2. ... 1. 3. 0.]
b': 2046 | [ 5675. 8905. 6018. ... 25668. 24414. 1851.]
e'': 1024 | [ 0. 0. -2. ... 1. 0. 1.]

-Computed---
v: 3069 | [ 5676. 8876. 60148. ... 25293. 10851. 56280.]
dbl_v: 3069 | [ 11352. 17753. 120296. ... 50586. 21702. 112560.]
c: 3069 | [ 0. 0. 1. ... 1. 0. 1.]
recA: 3069 | [ 11350. 17810. 120406. ... 50586. 21702. 112560.]

Keys---
Ka: 3069 | [ 0. 0. 0. ... 1. 0. 0.]
Ka2: 3069 | [ 0. 0. 0. ... 1. 0. 0.]
Equals: 99.44607363962203 %
PS F:\Documentos\URJC\TFG MAT\python>

```

Figura 3.2: Ka se ha generado de la forma usual y $Ka2$ usando el redondeo modular

4. Conclusiones

Como se menciona en el resumen inicial, el trabajo se ha centrado en presentar el protocolo BB84 y el algoritmo post-cuántico RLWE. En los archivos adjuntos se pueden encontrar todos los scripts en Python que se han usado en la realización de este TFG.

Tras la realización de este trabajo podemos confirmar que será necesario realizar una transición a estos nuevos algoritmos, ya que una vez que los ordenadores cuánticos se comercialicen, potencialmente cualquier persona tendrá la capacidad de romper los datos cifrados de hoy en día. No obstante, esta transición no es necesario que sea inmediata, pues hemos visto que los protocolos QKD aún se enfrentan a multitud de problemas. El protocolo BB84 (y sus derivados) aun necesitan del desarrollo de herramientas más precisas para la transmisión de fotones en largas distancias y, por supuesto, la detección de los mismos. Sin las herramientas adecuadas aún tendrán que esperar para que se extienda su uso.

Una de las partes más difíciles ha consistido en la explicación del algoritmo RLWE. Apenas se pueden encontrar documentos que hablen y describan con precisión en qué consiste. Tan solo encontré una explicación teórica a partir de la cual he procedido a implementarla sobre Python, simulando como se comportarían los individuos que quieren acordar una clave de cifrado. Una vez hecho el script, he podido estudiar en más detalle el algoritmo, tratando de realizar simplificaciones sobre él o modificando los cálculos que se deben hacer. Esto me ha permitido tener una visión más global.

Con toda esta información podemos hacer una breve valoración de las implicaciones que conllevarían dar el salto a usar un protocolo QKD o usar algoritmos post-cuánticos. Es muy probable que el principal inconveniente para decidimos por usar QKD es que debemos hacer una inversión inicial en nuevos dispositivos hardware, posiblemente muy costosos por tratarse de una tecnología novedosa. Además no solo nosotros deberíamos realizar dicha inversión, ya que el otro participante con el que queremos establecer una conexión también debe estar en posesión de dispositivos de similares características. Y todo ello suponiendo que los dispositivos sea suficientemente eficientes como para no tener brechas de seguridad que nos obliguen a renovarlos cada cierto tiempo. No obstante, si los costes son razonables, el uso de estos nuevos dispo-

sitivos se extendería fácilmente y sería una opción muy a tener en cuenta, ya que nos guiamos por leyes físicas, inmutables y estudiadas en detalle por multitud de físicos y matemáticos.

Mientras tanto sería aconsejable dar el salto a un algoritmo post-cuántico. Dicho salto no tiene que ser necesariamente a corto plazo, pues los ordenadores cuánticos aún siguen en desarrollo. No obstante, desde el NIST ya comenzamos a ver que comienzan a surgir líneas de investigación para estandarizar dichos esquemas y algoritmos criptográficos. Podemos esperar a que se estandaricen y se realicen estudios en profundidad sobre la seguridad que ofrecen ante un ataque con un computador cuántico. Además, como hemos visto en el caso de RLWE, es relativamente sencillo comenzar a usarlo, puesto que se trata de una variante del bien conocido *Diffie-Hellman*.

A. Apéndice

A.1. Algoritmo Shor

Este algoritmo, formulado en 1994, lleva este nombre debido a su creador, Peter Shor. Se trata de un algoritmo cuántico para la factorización en números primos de números compuestos, es decir, números que son resultado de multiplicar dos más pequeños.

Su potencia se cimienta en determinar el periodo de una función adecuada. Aunque su estudio presenta un grado de complejidad relativamente alto, es muy interesante analizar el nuevo enfoque que la mecánica cuántica ofrece para solucionar el problema de factorización.

Algoritmos cuánticos como el de Shor nos muestran por qué es necesario el estudio de nuevos esquemas criptográficos, ya que tendremos el poder de romper los esquemas clásicos de criptografía.

El algoritmo se compone de dos partes. La primera parte convierte el problema de factorización en el problema de encontrar el periodo de una función, y puede ser implementado de forma clásica (por un computador clásico). La segunda parte trata de encontrar el periodo usando la transformada cuántica de Fourier, y es el responsable de la aceleración cuántica (ver [13]).

* * *

El problema consiste en, dado un número compuesto e impar N , hallar un entero d que divida N tal que $1 < d < N$. Resulta obvio que es necesario que N sea impar, ya que para todo número par es trivial ver que tiene a 2 como factor primo. De hecho podemos usar el test de primalidad para verificar que de verdad se trata de un número compuesto.

Además, debemos verificar que N no sea una potencia de primos. Es decir: $\forall k \leq \log_2(N)$ tenemos que $\sqrt[k]{N} \notin \mathbb{Z}$.

Puesto que hemos visto que N no es una potencia de un número primo, es producto de dos coprimos mayores que 1. Como consecuencia del Teorema Chino del Resto, el número 1 tiene, por lo menos, 4 raíces distintas y con módulo N (dos de ellas son 1 y -1). Así pues, el objetivo del algoritmo es encontrar una raíz cuadrada b tal que $b \notin \{-1, 1\}$, tal que b llevará a cabo la factorización de N , como en otros algoritmos de factorización como la criba cuadrática (algoritmo de Pomerance).

Por tanto, el problema de encontrar b se reduce a encontrar un número a de periodo par con una determinada propiedad (que explicaremos más adelante). La parte cuántica del algoritmo es usada para encontrar el periodo de elementos a elegidos al azar, ya que este es un problema muy complejo para un computador clásico.

Pasemos a explicar la **parte clásica** del algoritmo.

1. Tomamos un número $a < N$ cualquiera al azar.
2. Calculamos $\text{mcd}(a, N)$ (podríamos usar el algoritmo Euclidiano).
3. Si llegamos a $\text{mcd}(a, N) \neq 1$ significa que hemos encontrado un factor no-trivial de N y habremos terminado.
4. En otro caso, habría que usar la subrutina de búsqueda del periodo (ver la parte cuántica) para hallar r , el periodo de la siguiente función:

$$f(x) = a^x \bmod N$$

Es decir, el orden r de a en \mathbb{Z}_N^+ , que es el entero más pequeño $r > 0$ tal que $f(x + r) = f(x)$, lo que es lo mismo: $f(x + r) = a^{x+r} \bmod N \equiv a^x \bmod N$

5. Si r es impar, volver al paso 1.
6. Si $a^{r/2} \equiv -1 \bmod N$, volver al paso 1.
7. Si $\text{mcd}(a^{r/2} + 1, N)$ y $\text{mcd}(a^{r/2} - 1, N)$ son (ambos) factores no-triviales de N , hemos terminado.

Por otro lado, tenemos la **parte cuántica** (subrutina de búsqueda del periodo). Los circuitos cuánticos usados para este algoritmo tienen un diseño para cada elección de N , y cada elección del número aleatorio a usado en $f(x) = a^x \bmod N$.

Dado N , hallar $Q = 2^q$ tal que $N^2 \leq Q < 2N^2$, que implica que $Q/r > N$. Los registros de qubits (tanto de entrada como de salida) necesitan almacenar superposiciones de valores de 0 a $Q - 1$, y, por tanto, tener q qubits cada uno. Utilizar lo que podría parecer el doble de qubits que sea necesario garantiza que hay al menos N diferentes x que producen la misma $f(x)$, incluso cuando el periodo r se aproxima a $N/2$.

Así pues, el algoritmo sigue así:

1. Se inicializan los registros a:

$$|\psi\rangle = \frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x\rangle$$

Este estado inicial es una superposición de Q estados.

2. Construir $f(x)$ como una función cuántica y aplicarla al anterior estado, de lo cual obtendremos:

$$|\psi\rangle = \frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x, f(x)\rangle$$

En este resultado todavía mantenemos la superposición de Q estados. Equivalentemente podemos escribir:

$$|\psi\rangle = \frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x, a^x \bmod N\rangle$$

3. En este paso, aplicamos la transformada cuántica de Fourier al registro de entrada. Esta transformada (operando en una superposición de $Q = 2^q$ estados) usa una raíz de la unidad Q tal que $\omega = e^{\frac{2\pi i}{Q}}$ para distribuir la amplitud de cualquier estado $|x\rangle$ entre todos los Q de $|y\rangle$, y hacerlo de manera diferente para cada x distinto.

Sea y uno de los posibles enteros r módulo Q tal que yr/Q es un entero. Entonces:

$$U_{QFT} |x\rangle = \frac{1}{\sqrt{Q}} \sum_y \omega^{xy} |y\rangle$$

Lo que nos lleva al estado final:

$$\frac{1}{Q} \sum_x \sum_y \omega^{xy} |y, f(x)\rangle$$

Esto es una superposición de más de Q estados, pero menos que Q^2 , ya que hay menos de Q valores distintos de $z = f(x)$. Sea:

- $\omega = e^{\frac{2\pi i}{Q}}$ sea una raíz de la unidad Q
- r el periodo de la función f
- x_0 el valor más pequeño de x tal que $f(x) = z$ (tenemos que $x_0 < r$)
- b indexa estos x , desde 0 hasta $\lfloor (Q - x_0 - 1)/r \rfloor$ tal que $x_0 + rb < Q$

Entonces ω^{ry} es un vector unitario en el plano complejo (ω es una raíz de la unidad, y r e y son enteros), y el coeficiente de $Q^{-1} |y, z\rangle$ en el estado final es:

$$\sum_{x: f(x)=z} \omega^{xy} = \sum_b \omega^{(x_0+rb)y} = \omega^{x_0 y} \sum_b \omega^{rby}$$

Donde cada término en esta suma representa a un diferente camino de llegar al mismo resultado.

4. Realizamos una medición, de lo que obtendremos alguna salida y en el registro de entrada y z en el registro de salida. Puesto que f es periódica, la probabilidad de medir algún par y y z es dada por:

$$\left| \frac{1}{Q} \sum_{x: f(x)=z} \omega^{xy} \right|^2 = \frac{1}{Q^2} \left| \sum_b \omega^{(x_0+rb)y} \right|^2 = \frac{1}{Q^2} \left| \sum_b \omega^{bry} \right|^2$$

Ahora los análisis muestran que esta probabilidad es más alta cuanto más cercano es el vector unitario ω^{ry} al eje X (real y positivo), o cuanto más se acerca yr/Q a un entero. A menos que r sea una potencia de 2, no será un factor de Q .

A partir de aquí, los siguientes pasos se pueden realizar de forma clásica.

5. Puesto que yr/Q es cercano a algún entero c , el valor conocido y/Q es cercano al valor desconocido c/r . Realizando expansión de fracciones continuas en y/Q nos permitirá encontrar aproximaciones d/s que puedan satisfacer las siguiente condiciones:

$$\begin{aligned} & \blacksquare s < N \\ & \blacksquare \left| \frac{y}{Q} - \frac{d}{s} \right| < \frac{1}{2Q} \end{aligned}$$

Dada estas condiciones (y asumiendo que d/s sea irreducible), s es muy probable que sea el periodo r , o por lo menos un factor de la función.

6. Verificamos si $f(x) = f(x+s) \Leftrightarrow a^s \equiv 1 \pmod{N}$. Si es así, hemos terminado.
7. En otro caso, obtenemos más candidatos para r usando múltiplos de s , o usando otros s con d/s próximos a y/Q . Si algún candidato es válido, hemos terminado.
8. Si no hemos llegado a un resultado válido tendremos que volver al paso 1 de esta subrutina.

* * *

En contraste con hallar y multiplicar grandes números primos, no conocemos ningún algoritmo clásico y eficiente para la factorización de números grandes, por lo que tenemos que recurrir a algoritmos cuánticos como el de Shor.

Llamamos algoritmo eficiente si el número de operaciones elementales es asintóticamente polinómica según aumenta la longitud del dato de entrada (medido en bits). Este es el caso del algoritmo de Shor, cuya complejidad es $O(\log^3(N))$. Y por el contrario, por ejemplo, el algoritmo clásico de la criba cuadrática necesita

$$O\left(\exp\left(\sqrt[3]{\frac{64}{9}} N (\ln N)^{2/3}\right)\right)$$

operaciones para factorizar un número binario de N bits y, además, esto escala exponencialmente cuando aumenta la longitud del dato de entrada.

A.2. Definiciones y lemas de RLWE

Definición 1 (Base de un retículo). *Una base de un retículo \mathbf{L} es un conjunto $\mathbf{B} = (b_1, \dots, b_n)$ tal que:*

$$\mathbf{L} = \mathbf{L}(\mathbf{B}) = \mathbf{B} \cdot \mathbf{Z}^n = \left\{ \sum_{i=1}^n c_i b_i \mid c_i \in \mathbf{Z} \right\}$$

Definición 2 (El problema de decisión de RLWE). *Sean n, q, R, R_q como los mencionados en la notación de RLWE. Sea χ una distribución sobre R , y sea $s \xleftarrow{\$} \chi$. Definimos $O_{\chi,s}$ como el oráculo que hace lo siguiente:*

1. *Tomar $a \xleftarrow{\$} \mathcal{U}(R_q)$, $e \xleftarrow{\$} \chi$*
2. *Devuelve $(a, as + e) \in R_q \times R_q$*

El problema de decisión RLWE para n, q, χ es distinguir $O_{s,\chi}$ de un oráculo que devuelve muestras uniformes aleatorias de $R_q \times R_q$. En particular, si \mathcal{A} es un algoritmo, define la ventaja:

$$\text{Adv}_{n,q,\chi}^{\text{drlwe}}(\mathcal{A}) = \left| \Pr(s \xleftarrow{\$} \chi; \mathcal{A}^{O_{s,\chi}}(\cdot) = 1) - \Pr(\mathcal{A}^{\mathcal{U}(R_q \times R_q)}(\cdot) = 1) \right|$$

La distribución χ a la que se hace referencia será, normalmente, una distribución discreta Gaussiana sobre R .

Definición 3 *Sea un entero $q > 0$, definimos como función modular de redondeo:*

$$\lfloor \cdot \rfloor_{q,2} = \mathbf{Z}_q \rightarrow \mathbf{Z}_2, \quad x \mapsto \lfloor x \rfloor_{q,2} = \left\lfloor \frac{2}{q} x \right\rfloor \mod 2$$

Y definimos como la función de redondeo cruzado:

$$\langle \cdot \rangle_{q,2} = \mathbf{Z}_q \rightarrow \mathbf{Z}_2, \quad x \mapsto \langle x \rangle_{q,2} = \left\lfloor \frac{4}{q} x \right\rfloor \mod 2$$

Ambas funciones se pueden extender a los elementos de R_q . Para $f = f_{n-1}X^{n-1} + \dots + f_1X + f_0 \in R_q$ definimos:

$$\lfloor f \rfloor_{q,2} = \left(\lfloor f_{n-1} \rfloor_{q,2}, \lfloor f_{n-2} \rfloor_{q,2}, \dots, \lfloor f_0 \rfloor_{q,2} \right)$$

$$\langle f \rangle_{q,2} = \left(\langle f_{n-1} \rangle_{q,2}, \langle f_{n-2} \rangle_{q,2}, \dots, \langle f_0 \rangle_{q,2} \right)$$

Estas funciones que acabamos de definir son usadas por Peikert para definir el mecanismo de reconciliación. Si el módulo q es impar es necesario trabajar con \mathbf{Z}_{2q} en vez de \mathbf{Z}_q . Como usamos un valor de q impar tenemos que introducir la función de doblaje aleatorio dbl tal que $dbl : \mathbf{Z}_q \rightarrow \mathbf{Z}_{2q}, x \mapsto 2x - e$, donde e es tomada aleatoriamente de $\{-1, 0, 1\}$ con probabilidades $p_{-1} = p_1 = 0,25$ y $p_0 = 0,5$

Lema 1 *Para un impar q , si $v \in \mathbf{Z}_q$ es uniformemente aleatorio y $\bar{v} \xleftarrow{\$} dbl(v) \in \mathbf{Z}_{2q}$, entonces $\lfloor \bar{v} \rfloor_{2q,2}$ es uniformemente aleatorio dado $\langle \bar{v} \rangle_{2q,2}$*

La función de doblaje aleatorio dbl se extiende a los elementos $f \in R_q$, aplicándola a cada uno de los coeficientes, resultando en un polinomio en R_{2q} , que a su vez se puede tomar como entrada de las funciones de redondeo $\lfloor \cdot \rfloor_{2q,2}$ y $\langle \cdot \rangle_{2q,2}$

Además, necesitamos una función de reconciliación. Esta función se define para recuperar $\lfloor v \rfloor_{q,2}$ de un elemento $w \in \mathbf{Z}_{2q}$ cercano a un elemento $v \in \mathbf{Z}_q$, dados únicamente w y la función de redondeo cruzado $\langle \bar{v} \rangle_{q,2}$.

Definición 4 Sean $I_0 = \{0, 1, \dots, \lfloor \frac{q}{2} \rfloor - 1\}$, $I_1 = \{-\lfloor \frac{q}{2} \rfloor, \dots, -1\}$ y $E = [-\frac{q}{4}, \frac{q}{4}]$. Definimos la función de reconciliación $rec : \mathbf{Z}_{2q} \times \mathbf{Z}_2 \rightarrow \mathbf{Z}_2$ como:

$$rec(w, b) = \begin{cases} 0, & \text{si } w \in I_b + E \text{ mod } 2q \\ 1, & \text{en otro caso} \end{cases}$$

Lema 2 *Con q siendo impar, sea $v = w + e \in \mathbf{Z}_q$ con $w, e \in \mathbf{Z}_q$ tal que $2e \pm 1 \in E \pmod{q}$. Entonces $rec(2w, \langle \bar{v} \rangle_{2q,2}) = \lfloor \bar{v} \rfloor_{2q,2}$*

Teorema 1 (Dificultad del problema de decisión DHI ó DDHI) *Sea un entero impar q , n un parámetro, y χ una distribución en R_q . Si el problema de decisión de RLWE para q, n, χ es difícil, entonces el problema DDHI para q, n, χ es aún más difícil. Concretamente:*

$$\text{Adv}_{n,q,\chi}^{\text{ddh}}(\mathcal{A}) \leq \text{Adv}_{n,q,\chi}^{\text{drlwe}}(\mathcal{A} \circ \mathcal{B}_1) + \text{Adv}_{n,q,\chi}^{\text{drlwe}}(\mathcal{A} \circ \mathcal{B}_2)$$

donde \mathcal{B}_1 y \mathcal{B}_2 son algoritmos de reducción.

Bibliografía

- [1] Joppe W. Bos and Craig Costello and Michael Naehrig and Douglas Stebila, *Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem*, 2015 IEEE Symposium on Security and Privacy, pp. 553–570, 2014.
- [2] José Ramón Soler Fuensanta *Una introducción a la criptografía clásica*
- [3] Ido Bregman, *Public Keys and Private Keys in Quantum Cryptography*, 2008.
- [4] Hoi-Kwong Lo y Yi Zhao, *Quantum Cryptography*, 2008.
- [5] Sivanagaswathi Kallam, *Diffie-Hellman: Key Exchange and Public Key Cryptosystems*, 2015.
- [6] R.L. Rivest, A. Shamir, y L. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, 1976.
- [7] Vicente Moret Bonillo, *Fundamentales de computación cuántica*, 2013.
- [8] Peter W. Shor y John Preskill, *Simple Proof of Security of the BB84 Quantum Key Distribution Protocol*, 2000.
- [9] Thomas Baignères, *Quantum Cryptography: On the Security of the BB84 Key-Exchange Protocol*
- [10] Michael R. Grimaila, Jeffrey Morris y Douglas Hodson, *Quantum Key Distribution*
- [11] Daniele Micciancio y Oded Regev, *Lattice-based Cryptography*
- [12] Georg Becker, *Merkle Signature Schemes, Merkle Trees and Their Cryptanalysis*
- [13] Hernando Efraín, *Algoritmo de factorización para un computador cuántico*
- [14] W. T. Buttler, S. K. Lamoreaux, J. R. Torgerson, G. H. Nickel, C. H. Donahue y C. G. Peterson, *Fast, efficient error reconciliation for quantum cryptography*, 2003