CRIPTOGRAFÍA POSTCUÁNTICA BASADA EN RETÍCULOS (LBC)

ARTURO MELERO ORTIZ

GRADO EN INGENIERÍA INFORMÁTICA. FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Arquitectura y Programación de Computadores Cuánticos (APCC)

Fecha
22 de Mayo de 2024

Profesor:

Guillermo Botella Juan

Índice general

Capítulo 1

Contextualización. La Criptografía Postcuántica

La criptografía clásica, que ha constituido el fundamento de la seguridad de la información durante décadas, se basa en la dificultad computacional de ciertos problemas matemáticos. Por ejemplo, el algoritmo RSA depende de la factorización de números grandes, y el algoritmo de Diffie-Hellman se basa en la complejidad del problema del logaritmo discreto. Sin embargo, del estudio de la computación cuántica han surgido algoritmos (como el de Grover y el de Shor) que, aun sin ser computables por las limitaciones de los computadores cuánticos modernos (NISQ), permiten vulnerar la seguridad de los sistemas criptográficos actuales.

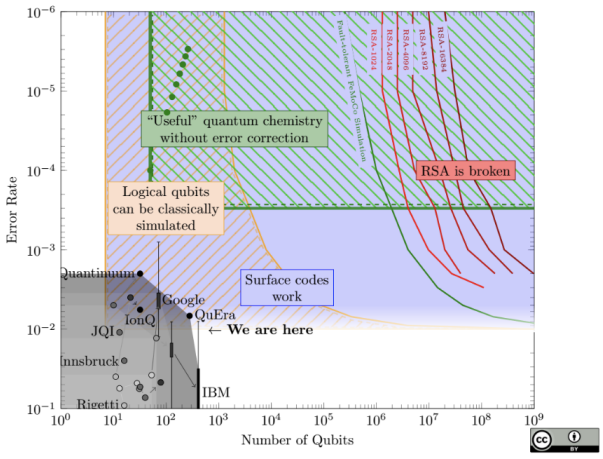

Figura 1.1: *Jaques Samuel. Panorama de la Computación Cuántica en 2023.*

El algoritmo de Grover, desarrollado por Lov Grover en 1996, constituye un ataque a la criptografía simétrica al proporcionar una aceleración cuadrática con respecto a los algoritmos clásicos al problema de buscar elementos concretos en un dominio desordenado. Por su parte, el algoritmo de Shor, desarrollado por Peter Shor en 1994, representa una amenaza

aún más grave para la seguridad de la información. Este algoritmo compromete la seguridad de numerosos sistemas de clave pública como RSA o Diffie-Hellmann, al ser capaz tanto de factorizar enteros como de resolver el problema del logaritmo discreto en tiempo polinómico, concretamente en $O(logN^3)$.

Aunque la respuesta postcuántica al algoritmo de Grover sea relativamente sencilla, pues consiste únicamente en duplicar el tamaño de las claves, la solución postcuántica al de Shor es bastante más problemática, debiendo introducir nuevos esquemas de cifrado. Hasta la fecha, han surgido numerosas propuestas basadas nuevamente en problemas matemáticos avanzados que se especulan intratables incluso para computadores cuánticos. Entre estas propuestas se encuentran la Criptografía Basada en Código (CBC), la Criptografía Basada en Hash (CBH) o la Criptografía Basada en Retículos (LBC - *Lattice Based Cryptography*), en la que se centra el presente documento.

1.1. Origen de la Criptografía Basada en Retículos (LBC)

La LBC ganó popularidad durante la década de 1990 con el trabajo de Miklós Ajtai[1] "*Generating Hard Instances of Lattice Problems*". Publicado en 1996, en él se presenta una familia de funciones unidireccionales o "*one-way function*" a modo de muestra de la dificultad computacional de los problemas basados en retículos. Con la atención de criptógrafos e investigadores, se desarrollaron primitivas criptográficas basadas en problemas como el de encontrar el vector más corto (*Shortest Vector Problem*, SVP) o el vector más cercano (*Closest Vector Problem*, CVP), derivando en la creación de esquemas de cifrado y firma basados en retículos.

En este sentido, la informática teórica Cynthia Dwork demostró que el problema conocido como "*Short Integer Solution* (SIS)" era, al menos, tan computacionalmente difícil de resolver como el caso-peor para cualquier problema basado en retículos[2]. Posteriormente, demostró que la seguridad de una función criptográfica hash era equivalente a la dificultad computacional del SIS. Por su parte, Jeffrey Hoffstein, Jill Pipher y Joseph H. Silverman presentaron en 1998 un esquema de clave pública basado en retículos conocido como NTRU[3] y del que, sin embargo, se desconoce ser al menos tan computacionalmente difícil como resolver el caso-peor de algún problema reticular. El primer esquema de encriptación de clave pública con seguridad probada surgió en 2005 con el trabajo de Oded Regev[4], junto con el problema de aprendizaje de errores (*Learning With Errors*, LWE). Desde entonces, se han realizado numerosas investigaciones para mejorar la seguridad y eficiencia de este esquema, construyendo nuevas primitivas basadas en problemas como LWE y similares.

El interés tras el estudio de la Criptografía Basada en Retículos (LBC) se debe principalmente a su potencial para hacer frente a la amenaza que la computación cuántica representa para la seguridad de la información, por la especulada intratabilidad de los problemas basados en retículos. Señalando este hecho, de los cuatro algoritmos de cifrado seleccionados por

el Instituto Nacional de Estándares y Tecnología - NIST por sus siglas en inglés - durante el congreso del 5 de Julio de 2022 para formar parte del estándar criptográfico postcuántico, tres de ellos se basan en la teoría de retículos[5]. Por estos motivos, se podría decir que la LBC es uno de los candidatos líderes en términos criptográficos, junto con CBC y CBH, mencionados en la sección anterior. Dicho todo esto, cabe mencionar que no hay nada que impida que el día de mañana surja un algoritmo cuántico contra un problema como LWE, de la misma forma que podría proporcionarse un algoritmo polinómico para resolver el problema SAT. La incertidumbre sobre la existencia de algoritmos polinómicos para determinados problemas es y será siempre una constante para el ámbito de la criptografía.

Capítulo 2

Fundamentos de la LBC

En el Capítulo 1 se introdujeron términos como que pueden no resultar familiares para cualquier lector. En esta sección, se exploran los conceptos esenciales para comprender la Criptografía Basada en Retículos (LBC), comenzando por los fundamentos de la Criptografía[6]. En la última subsección, se proporciona una breve introducción a la Teoría de Retículos lo suficientemente vaga para poder comprenderla a un nivel de abstracción elevado, sin profundizar en fundamentos matemáticos complejos.

2.1. Fundamentos de la Criptografía Clásica

La criptografía se define como el estudio del conjunto de técnicas que permiten proteger la información, garantizando además la seguridad en las comunicaciones. Históricamente, el objetivo primordial ha sido el de mantener la **confidencialidad** de los mensajes transmitidos. Para ello, los agentes se ponen de acuerdo en una clave común y se diseñan mecanismos de cifrado y descifrado. Utilizando la clave y conociendo el esquema de cifrado, una de las partes encripta el mensaje y lo transmite, de forma que la otra parte emplea su clave para desencriptarlo.

Definición 1. *Un **criptosistema, esquema de cifrado** o **cifrador** es una quíntupla $C = P, C, K, E, D$, donde P representa el texto plano, C el texto cifrado, K el conjunto de claves y E y D son las aplicaciones de cifrado –encriptado– y descifrado –desencriptado– respectivamente. Además, se garantiza que $\forall k \in K$ existen $e_k(x)$ y $d_k(x)$ tales que $e_k(d_k(x)) = x$, y viceversa. Es decir, aplicaciones inversas entre sí (idempotencia).*

El criptosistema más sencillo para ilustrar este concepto es el conocido como *cifrado de César*, consistente en sustituir cada letra del abecedario por su desplazamiento en el alfabeto un número determinado de posiciones. Los textos de este criptosistema son, pues, $P = C =$ Alfabeto Latino, con $A = 1, B = 2, \ldots, Z = 26$, dado que solo se trasponen las letras. Además, sea la clave $k \in \mathbb{Z}$, entonces el encriptado consiste en en desplazar k espacios a la derecha y el desencriptado a la izquierda:

$$E_k(x) = (x + k) \bmod 26 \qquad D_k(x) = (x - k) \bmod 26$$

Sin embargo, la **seguridad** de este esquema de cifrado no es adecuada. Dado que el espacio de claves es significativamente pequeño, realizando una **búsqueda exhaustiva de claves** es fácil para un atacante en la transmisión descifrar el mensaje, vulnerando la confidencialidad. Al conjunto de técnicas empleadas por una persona no autorizada para descifrar un mensaje se le conoce como **criptoanálisis**. Por este motivo, dada la vulnerabilidad al criptoanálisis de algunos esquemas de cifrado como el cifrado de César era habitual, y se sigue empleando, aplicar técnicas esteganográficas, consistentes en ocultar físicamente el mensaje para que los posibles atacantes no perciban su existencia.

Definición 2. *La **seguridad de un criptosistema** se define como su resistencia al criptoanálisis. Podemos clasificar la seguridad según la complejidad de los algoritmos en:*

1. *Seguridad incondicional. Perfecta, resistente al criptoanálisis con independencia del nivel computacional. El algoritmo One time pad se conoce que es indescifrable.*

2. *Seguridad probable. Romper el criptosistema se reduce a una posibilidad teórica.*

3. *Seguridad computacional. El criptosistema es computacionalmente seguro en el momento actual. Es, por tanto, temporal.*

Con el tiempo, se desarrollaron algoritmos más complejos con el objetivo de incrementar la seguridad de los cifradores al aumentar el tamaño del espacio de claves. Hasta poco antes de la Segunda Guerra Mundial, los esquemas de cifrado consistían en la reordenación de caracteres, bien por sustitución o bien por transposición. Entre estos, se encuentran el cifrado Afín, el cifrador Vigenère y el cifrado de Hill, basados en algoritmos que emplean aritmética modular. Así, por ejemplo, el cifrado Vigenère consiste en aplicar m cifrados de César, por lo que el número posible de claves se incrementa a 26^m.

Dada la existencia de espías y debido a la posible filtración de la información, es habitual que los esquemas de cifrado sean conocidos por los atacantes. La seguridad de la transmisión debe depender, por consiguiente, no tanto del algoritmo de encriptación de cifrado, sino de la ocultación de la clave.

Principio de Kerckhoffs. *La seguridad de un criptosistema debe recaer en la seguridad de la clave, debiéndose suponer conocidos el resto de parámetros del esquema de cifrado.*

Los esquemas de cifrado anteriormente mencionados, donde tanto el emisor como el receptor poseen una misma clave para encriptar y desencriptar los textos, pertenecen a la rama denominada **criptografía simétrica** o **criptografía de clave secreta**. Dada la capacidad computacional de los ordenadores, el tamaño de la clave resulta crucial para mantener la confidencialidad de este tipo de sistemas. En el apartado anterior se mencionaba, por ejemplo, como para paliar la amenaza de la computación cuántica con el algoritmo de Grover a los esquemas de criptografía simétrica se podía, simplemente, duplicar el tamaño de las claves. Sin embargo, la criptografía simétrica presenta un problema adicional: la **distribución de las claves** por un canal no seguro.

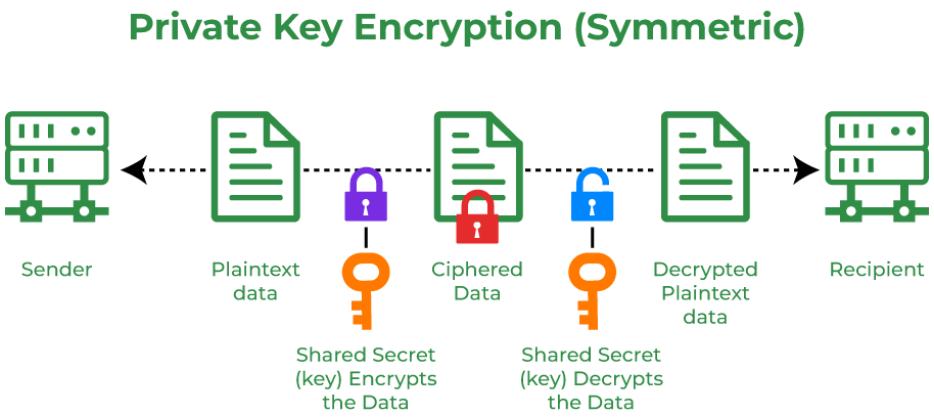

Figura 2.1: *Esquema de clave privada. Fuente: GeeksforGeeks.*

Para solucionar este problema, en la década de 1970 nace la **criptografía asimétrica**, siendo el RSA el primer y más utilizado algoritmo de este tipo. A los esquemas de cifrado pertenecientes a esta rama se les conoce como **criptosistemas de clave pública**. La seguridad de estos sistemas se basa en aplicaciones que son sencillas de computar en una dirección y complejas de deshacer. De esta forma, cada agente en la comunicación posee dos claves, una que hace pública y que sirve para encriptar y otra privada que debe cuidar para desencriptar. Así, sea la comunicación entre Alice (emisor) y Bob (receptor):

1. Bob envía a través del canal no seguro su clave pública.
2. Alice redacta un mensaje que encripta empleando la clave pública de Bob.
3. Bob desencripta el mensaje utilizando su clave privada, manteniendo la seguridad.

Por ejemplificar, RSA emplea la factorización de enteros como mecanismo de encriptación y desencriptación. Cada agente posee como clave privada dos números primos grandes y como clave pública el producto de los mismos. Dado que no se conoce un algoritmo eficiente que resuelva el problema de la factorización, resulta sencillo para Alice encriptar el mensaje y solamente Bob puede desencriptar el mensaje en un tiempo razonable al conocer su propia clave secreta.

La principal ventaja de la criptografía asimétrica es que resuelve el problema de la distribución de claves, pero tiene bastantes desventajas con respecto a la simétrica:

- El mensaje de cifrado ocupa más espacio que el original.
- Para una misma longitud de clave y mensaje se necesita mayor tiempo de proceso.
- Las claves deben ser de mayor tamaño que las simétricas.

Paralelamente, surgen nuevas técnicas criptográficas como la **criptografía híbrida**, que emplea tanto claves simétricas como asimétricas, o la **encriptación homomórfica**, que

permite aplicar operaciones equivalentes sobre textos planos y textos cifrados, permitiendo modificar el texto original sin necesidad de desencriptarlo.

Como último apunte, con el surgimiento de la **sociedad de la información** en las últimas décadas por el uso intensivo de las TICs, se demandan nuevos servicios en las comunicaciones. Por un lado, es necesario la aplicación de **códigos detectores y correctores de errores** para salvaguardar la integridad de los mensajes de errores físicos introducidos durante la transmisión por canales con ruido. En breves palabras, se introducen redundancias en los textos y se emplean algoritmos probabilistas para la detección y corrección. Por otro lado, se demandan nuevos servicios además de la confidencialidad.

Hasta este punto, se ha abordado cómo los criptosistemas proporcionan secretismo frente a ataques pasivos donde la intención del atacante era interceptar y comprender el mensaje. Sin embargo, en ocasiones hay adversarios activos cuyo objetivo es el de alterar la información. De esta forma, se debe garantizar la **integridad** del mensaje en primer lugar. Esto es, asegurar que la información no haya sido manipulada. Una forma de lograrlo es por medio de los *Message Authentication Codes* (MAC): se añade una etiqueta o tag al final del mensaje empleando una función hash criptográfica y la clave privada k, de forma que el tag quedará alterado si el texto original ha sido modificado. Por otra parte, para evitar comunicaciones falsas entre un emisor fraudulento y un receptor víctima se deben proporcionar mecanismos de **autenticación**. Para ello, se emplean **esquemas de firma** que confirman bien la autenticidad de la información o bien la identidad del usuario. En sistemas de clave pública, por ejemplo, la clave privada especifica un algoritmo de firma y un algoritmo de verificación de la misma: *sign-then-encrypt*. Estos esquemas de firma deben garantizar, además, la propiedad de **no repudio**. Es decir, Alice no puede firmar un mensaje y negarlo después, pues la firma se debe poder verificar. En este sentido, MAC permite garantizar la integridad pero no cumple con el principio de no repudio, por lo que se dice que es negable –del inglés, *deniable*–.

2.2. Teoría de Retículos. Problemas.

Definición 3. *Dados n vectores linealmente independientes $b_1, b_2, ..., b_n$ en $\mathbb{R}^m$, se define el retículo generado por dicho conjunto de vectores como*

$$\Lambda(b_1, ..., b_n) = \{\sum x_i b_i \mid x_i \in \mathbb{Z}\}$$

*El **rango** del retículo es n, y se emplean tanto los términos **punto del retículo** y **vector del retículo** para referirse a sus elementos.*

Es decir, sean n vectores con m entradas de los cuales ninguno de ellos puede ser expresado como combinación lineal del resto, se toma la suma de cada uno de ellos multiplicado por algún escalar entero y se obtiene un nuevo vector. De esta forma, los *puntos del retículo* son aquellos a los que se accede mediante un *vector del retículo* desde el origen $(0, 0)$.

Ejemplo 1. *Sea* $n = 2$*, dados los vectores* $b_1 = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$ *y* $b_2 = \begin{bmatrix} -3 \\ 4 \end{bmatrix}$*. Entonces, el retículo formado por los dos vectores se puede describir por el siguiente conjunto de puntos:*

$$L = \{z_1 b_1 + z_2 b_2\} = \begin{bmatrix} 4 & -3 \\ 2 & 4 \end{bmatrix} \times \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}.$$

En la siguiente imagen, se muestran algunos puntos del retículo.

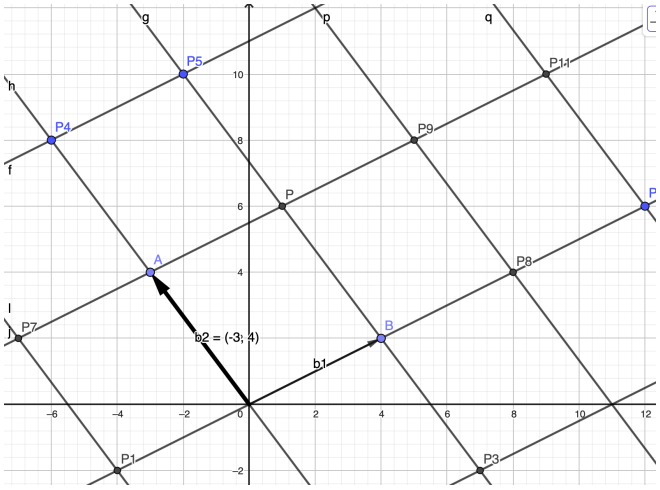

Figura 2.2: *Ejemplo 1. Realizado en Geogebra.*

En general, los retículos se describen como una matriz de tamaño $m \times n$, de forma que el núcleo central de la LBC se fundamenta en realizar manipulaciones sobre dicha matriz. Cabe, entonces, plantearse: ¿por qué emplear esta noción en lugar de utilizar un espacio vectorial? A pesar de su similitud, la restricción de multiplicar exclusivamente por escalares enteros permite definir ciertos problemas computacionalmente difíciles de resolver para los que no se conocen algoritmos ni clásicos ni cúanticos eficientes. Entre estos problemas, se encuentran:

Problema 1. ***(Shortest Vector Problem (SVP))****. Dada la base de un retículo* Λ*, consiste en encontrar el vector no nulo* $v \in \Lambda$ *tal que la norma Euclidiana del vector* v *(*$||v||$*) sea mínima con respecto al origen.*

Considerar este problema para la base proporcionada en el ejemplo anterior puede aparentar ser sencillo. Pero cuando se dispone de vectores de 500 entradas, encontrar la respuesta exacta se convierte en un problema de decisión con optimización, de complejidad exponencial. Existen, además, numerosas variaciones del problema: el SVP aproximado según un factor γ (SVP_γ); considerando un radio r (DSVP); el problema del vector más corto único (USVP), etc.

Problema 2. ***(Closest Vector Problem (CVP))****. Dada la base de un retículo* Λ *y un vector* $v \notin \Lambda$*, consiste en encontrar el vector* $u \in \Lambda$ *más próximo a* v*.*

Ejemplo 2. *Sea el retículo los vectores* $b_1 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$ *y* $b_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ *y un punto externo* $p = (2, 3)$. *Resolver el CVP consiste en encontrar el vector del retículo más cercano a p. Calculando distancias:*

$$\textit{distancia a } (3,1) = \sqrt{(3-2)^2 + (1-3)^2} = \sqrt{5}.$$

$$\textit{distancia a } (1,2) = \sqrt{(1-2)^2 + (2-3)^2} = \sqrt{2}.$$

Por tanto, el vector más cercano a p en el retículo es $(1, 2)$.

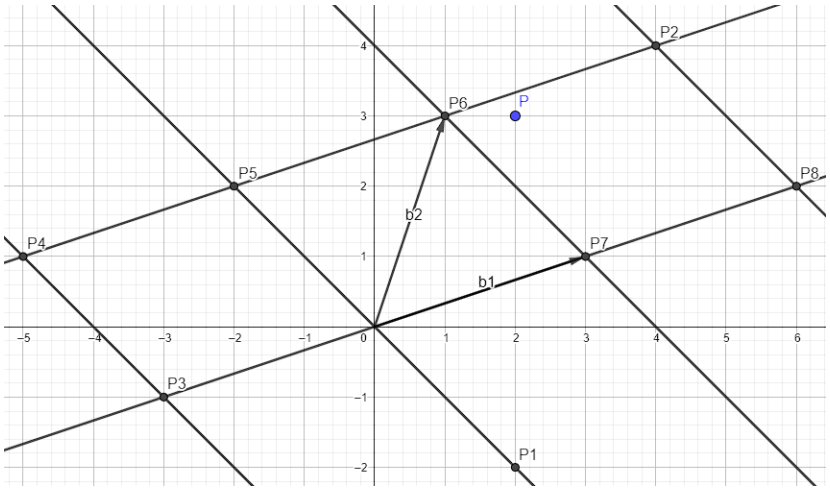

Figura 2.3: *Ejemplo 2. Realizado en Geogebra.*

Problema 3. ***(Bounded Distance Decoding (BDD))***. *Similar a CVP, dado un vector v cuya distancia al retículo* Λ *es como mucho* $\lambda(\Lambda)/2$*, el algoritmo debe proporcionar el vector del retículo máx próximo a este.*

Dificultad o dureza y su efecto en la criptografía. Los problemas mencionados y sus derivados pertenecen a NP-Difícil; sin embargo, existen instancias menos problemáticas que otras. Sin entrar en demasiados detalles, encontrar una solución a dichos problemas resulta más sencillo cuanto mayor sea el grado de ortogonalidad de los vectores de la base del retículo. Empeorando el panorama, existen algoritmos de reducción de bases de retículos como el de Lenstra-Lenstra-Lovász (LLL) o la ortogonalización de Gram-Schmidt que permiten obtener *buenas bases* a partir de las *malas*. El algoritmo de László Babai, por ejemplo, se basa en estas reducciones para resolver el problema SVP. Por este motivo, los criptosistemas originalmente introducidos por Ajtai y Dwork[2], Goldreich, Goldwasser y Halevi[7] y Regev[8] han sido reemplazados por otros similares basados en la dureza de un conjunto diferente de problemas de retículos. En la siguiente subsección, se presentan algunos de ellos. Esta presentación se complementará en el siguiente capítulo, donde se proporciona una guía para realizar una implementación sencilla de un esquema de cifrado de uno de ellos.

2.2.1. Problemas de Retículos Modernos

Por los motivos anteriormente citados, la mayoría de las propuestas de esquemas de cifrado modernos se basan en los siguientes problemas.

Problema 4. ***(Short Integer Solution (SIS))****. Introducido por Ajtai, dada una matriz elegida uniformemente $A \leftarrow \mathbb{Z}_q^{n\times m}$ y un parámetro real β, encontrar el vector no nulo $e \in \mathbb{Z}^m$ tal que*

$$Ae = 0 \ mod\ q \ y\ ||e|| \leq \beta.$$

Una matriz se dice ser elegida uniformemente cuando cada una de sus entradas se selecciona de entre un rango de valores con la misma probabilidad.

Problema 5. ***(Inhomogeneous Short Integer Solution (SIS))****. Dada una matriz elegida uniformemente $A \leftarrow \mathbb{Z}_q^{n\times m}$, un vector $u \leftarrow \mathbb{Z}_q^n$ y un parámetro real β, encontrar el vector no nulo $e \in \mathbb{Z}^m$ tal que*

$$Ae = u \ mod\ q \ y\ ||e|| \leq \beta.$$

Una matriz se dice ser "elegida uniformemente" cuando cada una de sus entradas se selecciona de entre un rango de valores con la misma probabilidad.

A continuación, se plantea el problema que utilizaremos más tarde durante la implementación de un esquema de cifrado.

Problema 6. ***(Learning With Errors (LWE))****. Sean $n, q \in \mathbb{N}$, y sea χ una distribución sobre enteros "pequeños" módulo q. Sea $s \leftarrow \mathbb{Z}_q^n$ un vector (secreto) uniformemente aleatorio. Entonces, la suposición de LWE consiste en que, para $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times n}$ una matriz (pública) uniformemente aleatoria y $\mathbf{e} \leftarrow \chi^n$, entonces $(\mathbf{A}, \mathbf{As} + \mathbf{e})$ es difícil de distinguir de algo completamente aleatorio. Se asume que $LWE_{n,q,\chi}$ es un problema intratable – indescifrable.*

Regev demostró que, para para ciertos módulos q y determinadas distribuciones de error acotadas χ, la suposición de $LWE_{n,q,\chi}$ anterior es verdadera mientras algunos de los problemas basados en retículos para su correspondiente caso-peor sean *difíciles* de resolver empleando algoritmos cuánticos[9]. En cualquier caso, la dureza de este problema depende fuertemente en la elección de los parámetros n, q, χ. Además, la introducción de los parámetros de error impiden que algoritmos como el de reducción Gaussiana *rompan* el problema. Por último, mencionar que existe una variación de este problema incorporando anillos de polinomios conocido como **RLWE**. Un último problema famoso es el de la asunción **NTRU**[10], aunque no profundizaremos en él.

Capítulo 3

Implementación en Python de esquema de cifrado basado en retículos

Existen numerosas propuestas de esquemas de cifrado basados en retículos. Al ser un área de investigación reciente, es común que se propongan esquemas que se consideran *seguros* solo para ser descartados poco tiempo después. Así, por ejemplo, el esquema de encriptación **GGH** (1997), basado en el SVP y para el cual se desarrolló también un esquema de firma, sucumbió al criptoanálisis realizado por Nguyen un par de años más tarde. Por su parte, se han desarrollado esquemas de encriptación homomórfica (Gentry, Brakerski y Vaikuntanathan), de firma (CRYSTALS-Dilithium, Falcon, NTRUSign, qTesla) y esquemas para el intercambio de claves (CRYSTALS-Kyber, FrodoKEM –basado en LWE–, NewHope –basado en RLWE–, NTRU Prime, etc.).

Dada su relevancia, se proporciona una guía sencilla para implementar un esquema de clave pública y otro de clave privada muy simplificados teniendo en cuenta todas las consideraciones previas. Para el intercambio de claves, se emplea una versión simplificada y poco optimizada de FrodoKEM.

3.1. Preliminares

El problema LWE puede interpretarse como un problema de decodificación de códigos lineales aleatorios; es decir, un código de corrección de errores. En particular, como el problema de, dada la descripción de la matriz aleatoria ($\mathbf{A}$) y un mensaje encriptado con ruido ($\mathbf{A}s+e$), obtener la decodificación s. Aunque esta interpretación es ligeramente distinta a la definición planteada con anterioridad, en la mayoría de contextos se acepta la equivalencia de estos problemas. Reformulado el problema, se proponen funciones auxiliares para facilitar las operaciones en aritmética modular (mod q).

```python
# Funciones para la generación de utilidades:

def gen_vec_unif(n, q): return [random.randint(0, q-1) for _ in range(n)]

def gen_matriz_unif(n, q): return [[random.randint(0, q-1) for _ in range(n)] for _ in range(n)]

def gen_vec_acotado(n, B): return [random.randint(-B, B) for _ in range(n)]


# Operaciones modulares con matrices:

def modular_mult(m1, m2, q): return (np.dot(m1, m2) % q).tolist()

def modular_add(m1, m2, q): return [(m1[i] + m2[i]) % q for i in range(len(m1))]

def modular_sub(m1, m2, q): return [(m1[i] - m2[i]) % q for i in range(len(m1))]
```

Figura 3.1: *Captura de pantalla 1. Funciones auxiliares.*

La generación de vectores y matrices uniformes, que se realiza por la descripción del propio problema, tiene como propósito asegurar que no haya patrones predecibles para dificultar el criptoanálisis. A este respecto, se debe notar que la aleatoriedad del programa no es real, pues se utiliza la aleatoriedad por defecto de Python. Para efectos didácticos, se puede obviar este hecho.

3.2. LWE. Esquema de Clave Privada

Recapitulando:

1. LWE se basa en la suposición de que es difícil resolver sistemas de ecuaciones lineales de la forma $b = As + e$, con A la base del retículo, s la clave secreta, e un vector de errores con valores pequeños y $b = As + e$.
2. La seguridad se basa en la dificultad de distinguir entre la distribución original y la distribución con el error.

Para la implementación de un sistema de clave privada para LWE, se sigue el siguiente esquema:

1. Alice o Bob eligen una clave privada s, empleando una distribución aleatoria uniforme, y la comparten entre sí.

2. Alice redacta un mensaje y lo encripta. Para encriptarlo, selecciona una matriz uniforme A y un vector de error e acotado en B. Utilizando la clave privada s:

$$E_s(m) = (A, As + e + (q//2)m)$$

   La última operación se corresponde con el escalado del mensaje para la toleración del error introducido e. Aunque existen otras técnicas más sofisticadas para este fin,

un escalado de este estilo es suficiente. En este caso, dado que el mensaje encriptado está en código binario ($m \in \{0,1\}^n$), se emplea $q/2$. Para el caso general, con $m \in \{0,...,p-1\}^n$, se emplearía $(q/p)m$.

3. Bob recibe el mensaje encriptado y lo desencripta. Para ello, substrae As del segundo componente y reescala el mensaje. Para mapear $(q/2)m + e \mapsto m$, se divide entre $q/2$ para conseguir $m + e/(q/2)$, redondeando el resultado. Dado que las coordenadas del error se asumieron "pequeñas", el reescalado funciona, obteniendo el valor original. (Concretamente, las coordenadas de e deben tener un valor absoluto $\leq q/4$).

El código correspondiente a la implementación de este esquema puede encontrarse en el Apéndice A.

3.3. LWE. Esquema de Clave Pública

La construcción del esquema de clave pública se vale de los mismos conceptos, pero debiendo proporcionar la manera de emplear claves públicas de forma segura. El esquema de clave pública para LWE es más sencillo de visualizar en términos del intercambio de claves de Diffie-Hellman.

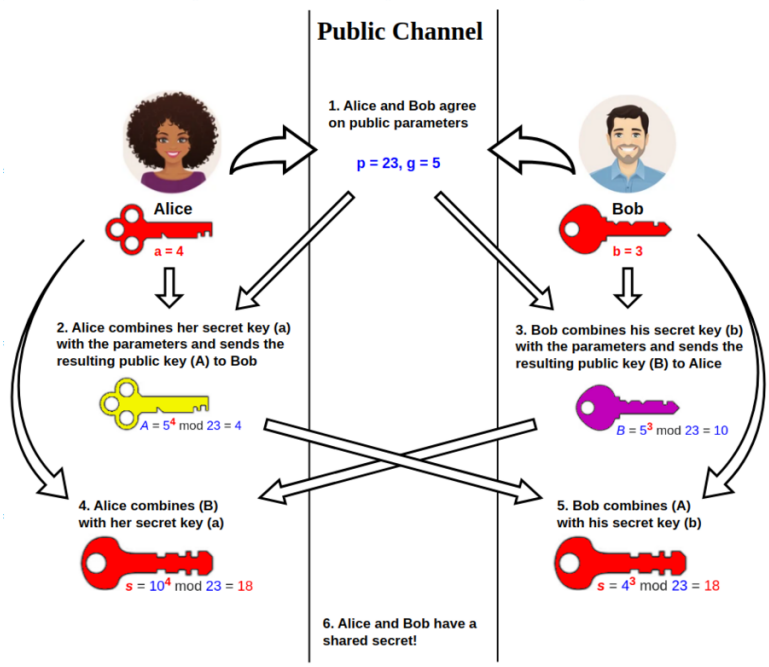

Figura 3.2: *Diffie-Hellman Key Exchange. Fuente: Wikipedia.*

A efectos prácticos, dado un generador g de un subgrupo cíclico de orden p, el esquema de cifrado funciona de la siguiente forma:

1. Se elige un primo grande p y un generador g de un subgrupo cíclico de orden p. Estos parámetros se hacen públicos.

2. Alice y Bob eligen valores aleatorios $r, s \leftarrow \mathbb{Z}_p$ que almacenan como sus claves secretas.

3. Utilizando el generador g, cada parte calcula un valor y se lo envía a la otra parte.

   - Alice calcula $A = g^r$ y se lo envía a Bob.
   - Bob calcula $B = g^s$ y se lo envía a Alice.

4. Una vez que Alice recibe B de Bob y Bob recibe A de Alice, cada uno calcula la clave compartida. Alice calcula $K_A = B^r$ y Bob $K_B = A^s$. Dado que $K = g^{rs}$, ambos obtienen la misma clave secreta ($K = K_A = K_B$).

Este proceso es seguro debido a la dificultad del problema del logaritmo discreto, que es la base matemática del algoritmo de Diffie-Hellman. Incluso si un atacante intercepta los valores A y B, sin conocer r o s, es computacionalmente difícil determinar la clave compartida K.

Para el problema LWE, s es la clave secreta y $As + e$ la pública. Aplicando el esquema anterior, se puede calcular $r^t As$ de dos formas, una como $r^t(As)$ y otra como $(r^t A)s$. En el Apéndice B se encuentra el código correspondiente a este esquema. Nótese que, debido a la simplificación del esquema, sólamente permite la encriptación/desencriptación de 1 bit. En realidad, FrodoKEM realiza algo similar a descomponer el texto plano en bloques y encriptar *múltiples cosas*, añadiendo bits de redundancia. Así mismo, aclarar que, para garantizar el correcto funcionamiento del algoritmo, debe cumplirse la siguiente relación entre los parámetros: $2nB^2 + B < q/4$. Esto es debido a que el error introducido en el esquema público es mayor. Aún así, si se cumplen determinadas condiciones relacionadas con los parámetros e, e', e'', r, sk la desencriptación puede fallar, aunque al ser elegidos de forma uniformemente aleatoria es improbable que ocurra.

Para facilitar la comprensión del código, se proporciona la ejemplificación de un intercambio de mensajes entre Alice y Bob. Supongamos que Alice quiere enviar un mensaje a Bob:

1. Generación de Claves (Bob):

   - $A, sk, As, e, b = (As + e) \bmod q$. Publica (A, b) y guarda sk.

2. Encriptación del mensaje (Alice):

   - Obtiene (A, b) de Bob. Genera r, e', e''
   - Calcula y envía $u = (r^t A + e') \bmod q$ y $v = (r^t b + e'' + (q/2) * m) \bmod q$.

3. Desencriptación del mensaje (Bob):

   - Recibe (u, v), calcula $r^t As$ empleando u como $r^t As = u * sk \bmod q$.
   - Ajusta v restando $r^t As$, es decir, $v = (v - r^t As) \bmod q$.
   - Decodifica m a partir de v.

Capítulo 4

Conclusiones Finales

La criptografía postcuántica representa un campo emergente de la criptografía que busca desarrollar sistemas resistentes a los ataques de los futuros computadores cuánticos. A diferencia de otros sistemas que dependen de la factorización de enteros o del problema del logaritmo discreto, la LBC se mantiene robusta frente a los algoritmos cuánticos conocidos, como el algoritmo de Shor. Esta resistencia inherente los posiciona como candidatos viables para su adopción en un futuro donde la computación cuántica sea una realidad tangible.

Además de su fortaleza en términos de seguridad, los criptosistemas basados en retículos han demostrado ser eficientes en cuanto a la velocidad de encriptación y desencriptación, aunque presentan desafíos en términos de tamaño de las claves y de los datos cifrados. Siguiendo esta línea, las investigaciones han mostrado avances significativos en la reducción de estos tamaños mediante técnicas como la compresión de claves y la optimización de los algoritmos de codificación y decodificación. Los esfuerzos actuales, como el proceso de estandarización llevado a cabo por el NIST, son pasos esenciales para la integración de estas nuevas tecnologías en la infraestructura de seguridad global.

En conclusión, aunque existen desafíos por superar, la criptografía basada en retículos se perfila como una solución robusta y eficiente para garantizar la seguridad de la información en la era postcuántica.

Nota. Otras fuentes utilizadas para la redacción del documento y desarrollo del trabajo incluyen: la introducción a la criptografía postcuántica de Agrawal[11]; LBC para principiantes[12]; las aportaciones de Mark Schultz[13]; resolución de problemas de retículos y la seguridad LBC[14].

Bibliografía

[1] Miklós Ajtai. Generating hard instances of lattice problems. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 99–108, 1996.

[2] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. Technical report, Revision of: TR96-065, May 1997. Accepted on: 7th May 1997 00:00, Downloads: 3023.

[3] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In *Algorithmic Number Theory*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

[4] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 84–93. ACM, ACM, 2005.

[5] National Institute of Standards and Technology (NIST). History of pqc standardization selected algorithm updates, 2022.

[6] Douglas R. Stinson. *Cryptography: Theory and Practice*. CRC Press, Boca Ratón, FL, 3rd edition, 2006.

[7] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *Annual International Cryptology Conference*, pages 112–131. Springer, 1997.

[8] Oded Regev. New lattice-based cryptographic constructions. *Journal of the ACM (JACM)*, 51(6):899–942, 2004.

[9] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6), 2009.

[10] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 333–342, 2009.

[11] Shweta Agrawal. Post quantum cryptography: An introduction. 2024.

[12] Dong Pyo Chi, Jeong Woon Choi, Jeong San Kim, and Taewan Kim. Lattice based cryptography for beginners – a supplementary note to the following. *Division of General Studies, UNIST; Department of Mathematics, Seoul National University; Fusion Technology R&D Center, SK Telecom; Department of Applied Mathematics, Kyung Hee University; Institute of Mathematical Sciences, Ewha Womans University*, 2024.

[13] Mark Schultz. Blog de mark schultz, presente. PhD Student in CS at UCSD.

[14] Thijs Laarhoven, Joop van de Pol, and Benne de Weger. Solving hard lattice problems and the security of lattice-based cryptosystems. September 10 2012.

Apéndice A

Esquema de Clave Privada en Python (LWE)

```
class LWEPrivKey:

    # INICIALIZACIÓN
    def __init__(self, n, q, B):
        self.n = n           # Nº de entradas de los vectores del retículo
        self.q = q           # Módulo para las operaciones modulares
        self.B = B           # Límite superior para los elementos del vector e

    # MÉTODO PARA LA GENERACIÓN DE CLAVES
    def key_gen(self):
        # Genera clave secreta s, con s[1..n] y elementos {0, ..., q-1}
        self.s = gen_vec_unif(self.n, self.q)

    # MÉTODO PARA ENCRIPTAR EL MENSAJE: E(m) = (A, As + e + (q/2)m)
    def enc(self, m):
        # Generamos la matriz A y el vector de errores e
        A = gen_matriz_unif(n, self.q)
        e = gen_vec_acotado(self.n, self.B)
        # Calculamos b:= As + e
        b = modular_mult(A, self.s, self.q)
        b = modular_add(b, e, self.q)
        # Escalamos el mensaje m -> (q//2)m
        scaled_m = [(self.q//2)*m[i] % self.q for i in range(self.n)]
        # Sumamos (q//2)m a b = As + e
        b = modular_add(b, scaled_m, self.q)
        return (A,b)
```

```
    # MÉTODO PARA DESENCRIPTAR EL MENSAJE: D(A,b) = m
    def dec(self, ctxt):
        (A,b) = ctxt[0], ctxt[1]
        # Recalculamos As
        As = modular_mult(A, self.s, self.q)
        # Recuperamos el mensaje escalado m_escalado = b - As
        b = modular_sub(b, As, self.q)
        # Reescalamos para recuperar el mensaje original (q//2)m + e -> m
        m = [0 for _ in range(self.n)]
        for i in range(self.n):
            m[i] = round(b[i] / (self.q//2)) % 2
        return m


# EJEMPLO DE USO: -----

# Generamos la clave
lwe = LWEPrivKey(n, q, B)
lwe.key_gen()

# Generar un mensaje aleatorio binario de longitud n
mensaje = [random.randint(0, 1) for _ in range(n)]
print("Mensaje original:", mensaje)

# Cifrar el mensaje
ctxt = lwe.enc(mensaje)
print("Texto cifrado:", ctxt)

# Desencriptar el mensaje
mensaje_recuperado = lwe.dec(ctxt)
print("Mensaje recuperado:", mensaje_recuperado)


# RESULTADO: -----

Mensaje original: [1, 1, 0, 1, 0]
Texto cifrado: ([[1604, 2391, 1310, 661, 897], [1258, 769, 975, 272, 567],
                [1625, 1850, 1124, 666, 347], [2815, 1638, 2493, 404, 1410],
                [296, 183, 2863, 378, 1888]], [2768, 2131, 2743, 812, 766])
Mensaje recuperado: [1, 1, 0, 1, 0]
```

Apéndice B

Esquema de Clave Pública en Python (LWE)

```
class LWEPubKey:

    # INICIALIZACIÓN
    def __init__(self, n, q, B):
        self.n = n          # Nº de entradas de los vectores del retículo
        self.q = q          # Módulo para las operaciones modulares
        self.B = B          # Límite superior para los elementos del vector e

    # MÉTODO PARA LA GENERACIÓN DE CLAVES
    def key_gen(self):
        A = gen_matriz_unif(self.n, self.q)
        self.sk = gen_vec_acotado(self.n, self.B)
        As = modular_mult(A, self.sk, self.q)
        e = gen_vec_acotado(self.n, self.B)
        b = modular_add(As, e, self.q)
        self.pk = (A,b)

    # MÉTODO PARA ENCRIPTAR EL MENSAJE:
    def enc(self, m):
        (A,b) = self.pk
        # Generamos r
        r = gen_vec_acotado(self.n, self.B)
        e_prime = gen_vec_acotado(self.n, self.B)
        # Calculamos u = A^tr + e'
        At = np.transpose(A).tolist()
        u = modular_mult(At, r, self.q)
        u = modular_add(u, e_prime, self.q)
        # Producto interno r^tb
```

```
        rAs = sum(r[i]*b[i] for i in range(len(r))) % self.q
        # Añadimos otro termino de error
        e_prime_prime = gen_vec_acotado(1, self.B)[0]
        # Calculamos v=rAs+e"+(q/2)m módulo q
        v = rAs + e_prime_prime % self.q
        v = (v + (self.q//2)*m) % self.q
        return (u,v)

    # MÉTODO PARA DESENCRIPTAR EL MENSAJE:
    def dec(self, ctxt):
        (u, v) = ctxt[0], ctxt[1]
        s = self.sk
        # Recalcular (r^tA)s
        rAs = sum(u[i]*s[i] for i in range(len(u))) % self.q
        # Ajusta v restando rAs
        v = (v - rAs) % self.q
        # Decodificación: (q//2)*m + error_terms
        return round(v/(self.q//2)) % 2



# EJEMPLO DE USO: -----

# Creamos instancia
lwe = LWEPubKey(n,q, B)
# Generamos claves (pública y privada)
lwe.key_gen()
# Mensaje a cifrar
m = random.randint(0,1)
print("Mensaje:", m)
# Encriptamos el mensaje
c = lwe.enc(m)
# Desencriptamos el mensaje
t = lwe.dec(c)
print("Desencriptado: ", t)


RESULTADO: --

Mensaje: 0
Desencriptado:  0
```