Jonathan Alejandro Gonzalez Perez

Corrección De Errores Con Machine Learning En Un Sistema De Distribución De Claves Cuánticas (QKD)

Tesis para obtención de grado de licenciado en ingeniera física

22 de febrero de 2022

Universidad de Guanajuato
División de Ciencias e Ingenieras Campus León
Departamento de Física

UNIVERSIDAD DE GUANAJUATO

Universidad de Guanajuato

Tesis para obtención de grado de licenciado en ingeniera física
División de Ciencias e Ingenieras Campus León
Departamento de Ingeniera Física

Fecha de Envío 20/01/2022;
Fecha de Aprobación 22/02/2022;
Fecha de Defensa 04/03/2022;

Asesores
Dr. Carlos Herman Wiechers Medina (Asesor)
Dr. José Luis Lucio Martínez. (Co-Asesor)

Sinodales

- Dr. Luis Carlos Padierna Garcia
- Dr. Teodoro Cordova Fraga
- Dra. Lorena Berenice Velázquez Ibarra
- Dr. Carlos Herman Wiechers Medina

Especialmente para:

Mi Madre
Mi Padre
Mi hermana

Quienes han estado ahí desde siempre
y entre nosotros nos respaldamos para todo

Y,
a todas las personas que dejaron
su huella en este camino

Тише едешь, Дальше будешь

- Proverbio Ruso

Nada tiene el poder de expandir la mente como la habilidad de investigar de manera sistemática y encontrar la verdad a través de las observaciones de la vida

- Marcus Aurelius

Agradecimientos

Este proyecto de tesis ha sido posible gracias a la ayuda, colaboración y mentoria del Dr. Carlos Herman Wiechers Medina, quién me permitió aprender acerca de temas en óptica cuántica y tecnologías cuánticas, así como también por su ayuda y comprensión en el tutorado. Sin duda, además de ser un gran científico y profesor es también un ejemplo de persona con un gran corazón.

De la misma forma, gracias a los comentarios, sugerencias y apoyo del Dr. José Luis Lucio Martínez, lo cual me permitió elaborar mas acerca de temas fundamentales para el eje principal de este proyecto. Agradezco su amabilidad y atención al momento de compartir sus sugerencias y comentarios.

Por último pero no menos importante, agradezco a la Dra. Lorena Berenice Velázquez Ibarra por su apoyo como sinodal y sus enseñanzas e inspiración durante la carrera. Al Dr. Luis Carlos Padierna García, por su apoyo como sinodal y la inspiración y conocimiento que compartió conmigo, como su alumno, en los cursos de licenciatura. Al Dr. Teodoro Cordova Fraga por su apoyo como sinodal y las enseñanzas e inspiración que compartió conmigo, como su alumno en los cursos de licenciatura.

Resumen

En este proyecto de tesis, se habla de la encriptación cuántica, sus orígenes, el uso que tiene actualmente y como se desarrollará en un futuro. El objetivo principal es crear un algoritmo de corrección de error y reconciliación de llaves cuanticas mediante el uso de métodos y tecnologias de aprendizaje de maquina (Machine Larning) y de Soft Computing como lo son las redes neuronales. Como propuesta final se sugiere el uso de una maquina de paridad de árbol o TPM (Tree Parity Machine) como método de reconciliación para los datos generados por el criptógrafo que se encuentra en el laboratorio de cuántica de la División de Ciencias e Ingenierías de la Universidad de Guanjauto.

El contenido general de este documento gira en torno de los principios de la mecánica cuántica que se aplican para obtener computación cuántica y por consecuente los métodos de encriptación que aprovechan estos principios. Se profundiza en conceptos generales y especificos de computación y programación orientada a métodos de machine learning y las redes neuronales.

El objetivo general del trabajo es obtener una llave de encriptación corregida y sincronizada entre un emisor A y un receptor B, comúnmente a estos sujetos se les llama Alice y Bob. El método que se aborda es mediante el uso de redes neuronales que se rigen por reglas de aprendizaje basadas en el comportamiento de neuronas orgánicas, tal es el caso de las reglas de Hebbian y Anti Hebbian Así como también un Random Walker .

Índice general

Parte I

Teoría y Conceptos

Capitulo 1

Introducción

La ciencia y tecnología han tenido un avance importante en el ultimo siglo. Pasamos de pensar que en el vacío existía una sustancia llamada éter por la cual se propagaban ondas electromagnéticas, a construir teorías y a manipular las piezas mas fundamentales que constituyen la materia. En 1905, Max Planck comienza a sentar las bases e ideas para la mecánica cuántica.

La Mecánica Cuántica es una rama bastante extraña e interesante, cuyos principios aportan mas al mundo de lo que es evidente, una rama que aun sigue en sus primeros pasos y es tan extensa que apenas y podemos lograr entenderla. A partir de algunos principios de esta rama de la física, podemos crear tecnologías y métodos que nos facilitan la vida, no sin antes complicarnos ante el desarrollo y aplicación de estos principios.

También en el siglo pasado comenzamos a crear maquinas autómatas que nos ayudaron a facilitar la vida y simplificar procesos. Alan Turing, el padre de la computación moderna, crea un precursor de la computadora moderna para poder descifrar los mensajes secretos que los alemanes se mandaban entre si, durante la Segunda Guerra Mundial, usando la máquina Enigma. A partir de ahí, la computación comenzó su auge y su ascenso. La era de la información había comenzado.

Hoy en día, el manejo de la información esta presente en todos lados, lo que nos ha llevado a la creación y desarrollo de diferentes herramientas para compartir y proteger la información importante. Un ejemplo muy claro de estas herramientas es el internet, pues dentro de este se comparte una gran cantidad de información día con día.

Para compartir esta información y asegurar que llegue únicamente al destinatario que se desea, el mensaje tiene que ser cifrado, es decir, se debe transformar el mensaje a un código que solo el emisor y receptor entiendan y debe existir una llave que regrese el mensaje a la forma original.

Actualmente existe algo interesante en la implementación física de la información, y es la posibilidad real de extender el unidad básica de

información del 'bit' al 'qubit'. Una nueva magnitud que contiene a la antigua y evoluciona con nuevas reglas dictadas por la física cuántica. Su uso nos conduce a los ordenadores cuánticos y más concretamente a lo que será el tema de este proyecto: los sistemas de criptografía.

1.1 Motivación y objetivos

Estos sistemas de cifrado y aplicaciones de los principios de la mecánica cuántica, en conjunto con algoritmos y métodos de machine learning y soft computing, hacen que la seguridad y eficiencia de un sistema de cifrado cuántico aumenten. Es por eso que en este trabajo se buscará implementar algoritmos y métodos de soft computing como lo son las redes neuronales, específicamente una maquina de árbol de paridad (TPM - Tree Parity Machine) para realizar la reconciliación de las llaves distribuciones de llaves cuánticas (QKD - Quantum Key Distribution).

1.1.1 Antecedentes

La encriptación ha estado presente en la sociedad desde la antigüedad. En Roma se utilizó el sistema de cifrado Cesar que constaba de desplazar n espacios las letras del alfabeto hacia la izquierda o hacia la derecha, generando así un nuevo código. De esta forma el ejercito romano se aseguraba de llevar los mensajes de manera segura. Ya que si algún enemigo robaba el mensaje era poco probable que pudiera descifrar el contenido. Hoy en día, existen métodos mas sofisticados y difíciles de descifrar, pues con las herramientas que tenemos hoy en día, descifrar el código Cesar es una tarea que se realiza en pocos segundos.

Un método que continúa en desarrollo es el cifrado cuántico. La idea nació a finales de los 60's por Stephen Wiesner [1], aunque fue publicada hasta 1983. La idea de Wiesner describe un mecanismo para crear dinero imposible de falsificar. Incorporando un total de 20 trampas de luz en los billetes de un dolar, y codificando en cada una de ellas uno de dos posibles valores con un fotón polarizado. La seguridad de esta idea reside en que el fotón contenido en cada trampa de luz es polarizado con respecto a una base, y sólo con el conocimiento de esa base se puede recuperar el estado de polarización correcto de cada fotón sin ninguna posibilidad de error. Si no se conoce la

base el resultado del proceso de lectura es completamente aleatorio, produciendo con igual probabilidad el resultado correcto o el incorrecto.

Un año después de esta publicación, Charles Bennet y Gilles Brassard definieron el primer protocolo de criptografía cuántica: el algoritmo BB84 [2]. Basado en los principios de la mecánica cuántica, este protocolo constituye el primer diseño práctico de un sistema de encriptación cuántica. En este algoritmo se establecen los dos primeros niveles de trabajo de un sistema de QKD: el intercambio de clave en bruto y la reconciliación de bases, que en un sistema ideal y libre de errores, son suficientes para realizar el intercambio de una clave. En la práctica, su implementación se ve afectada por imprecisiones del sistema y su entorno.

El contenido del capítulo dos se centra en los fundamentos de la criptografía clásica y los principios de mecánica cuántica que se usan en la criptografía cuántica. El capítulo tres habla acerca de la computación cuántica y como funciona de manera general, se mencionan la encriptación, la reconciliación y los algoritmos de prueba que se usan en la computación cuántica. El capítulo cuatro profundiza en los temas de la computación difusa y el aprendizaje de maquina (soft computing y machine learning en inglés), donde se menciona el funcionamiento de la redes neuronales, las TPM y como se aplican para obtener la llave depurada como resultado. En el capítulo cinco se presenta la metodología de trabajo, el montaje experimental y se explican los códigos de corrección por medio de diagramas de flujo. En el capítulo seis se presentan los resultados obtenidos después de aplicar las TPMs y obtener la llave depurada. Finalmente en el capítulo siete se encuentran las conclusiones y la perspectiva que se obtiene al finalizar este proyecto.

Capitulo 2

Algunos Conceptos Generales

Antes de poder adentrarse a la corrección de errores, es necesario entender ciertos conceptos de mecánica cuántica que se aplican a la criptografía cuántica. Se pretende en este capitulo recordar y explicar fundamentos para entender como funciona el sistema de criptografía utilizado. Además, se describe como se realiza la reconciliación y la corrección de errores.

2.1 Criptografía Clásica

La criptografía se define como la técnica o disciplina que permite escribir apelando a un código oculto [3]. De esta forma, quien no conoce la clave para descifrar el mensaje no puede acceder a su contenido. Estas técnicas existen desde hace muchos siglos. En consecuencia del desarrollo del conocimiento, que se ha adquirido en el ultimo siglo, estas técnicas se han hecho mas sofisticadas y complejas.

El objetivo fundamental de la criptografía es que dos entidades puedan comunicarse a través de un canal inseguro, de forma que un tercero no pueda entender lo que se esta comunicando.

Este objetivo se alcanza mediante técnicas y métodos que se han definido con operaciones matemáticas, una encriptación es perfecta si presenta:

1. Confidencialidad
2. Autenticación
3. Integridad de la información
4. No-autorización

El aspecto de confidencialidad es usado para mantener guardado el contenido de la información de todos, excepto de quien tiene la autorización para poder acceder al contenido. Hoy en día, existen

muchas formas de asegurar la confidencialidad que vemos día a día, algunos ejemplos son las contraseñas de las cuentas de banco, correo o para acceder a las computadoras y teléfonos.

Para asegurar la integridad de los datos, se debe tener la capacidad de detectar la manipulación de datos por partes no autorizadas. Para esto la autenticación se encarga de identificar al usuario que tiene acceso a los datos y al que no lo tiene, a esto se le llama autenticación de identidad. Otro tipo de autenticación es la verificación del origen de los datos, es decir, se debe recibir la información solo de fuentes confiables para asegurarse de la integridad del contenido.

Cuando la autenticación falla, se genera una no-autorización que impide un acceso a una entidad desconocida, o el acceso a una información de una fuente dudosa. Así, se protege la integridad de los datos y de la información que pueda poseer el mensaje o el receptor.

2.1.1 Elementos Criptográficos

Para entender la criptografía un poco mas a fondo, se introducirán algunos conceptos básicos de criptografía.

El Texto Llano es aquel mensaje que se quiere comunicar sobre un canal inseguro. Para que este texto se pueda compartir de manera segura, se somete a un proceso de cifrado que nos genera un Texto Cifrado, que solo se puede leer con una Llave de cifrado. Este texto ahora es seguro de compartir en un canal publico, siempre y cuando la única persona que posea la llave sea el destinatario.

Un criptosistema o esquema de ecriptación es un conjunto formado por (P.C,K,E,D) donde:

- P, Conjunto de textos llanos.
- C, Conjunto de textos cifrados.
- K, Conjunto de claves de encriptación.
- E, Familia de funciones de encriptación.
- D, Familia de funciones de desencriptación.

Las funciones de encriptación actúan como:

$$E_k : \mathrm{P} \rightarrow \mathrm{C}, \ \forall \ \mathrm{k} \in \mathrm{K}. \tag{2.1}$$

Las funciones de desencriptación actuan como:

$$D_k : \mathrm{C} \rightarrow \mathrm{P},\ \forall\ \mathrm{k} \in \mathrm{K}. \tag{2.2}$$

La condicion de que un sistema constituya un criptosistema es que se verifique la propiedad

$$\forall \mathrm{e} \in \mathrm{K},\ \exists\ \mathrm{d} \in \mathrm{K}/D_{\mathrm{k}}(E_{\mathrm{e}}(\mathrm{p})) = \mathrm{p},\ \forall \mathrm{p} \in \mathrm{P}. \tag{2.3}$$

Un cripto sistema se denomina simétrico cuando d y e son iguales. por el contrario, se denomina asimétrico si d y e son diferentes.

El ataque a un criptosistema se denomina criptoanálisis. Hay que considerar diferentes tipos de criptoanálisis:

- Ataque de texto cifrado.
- Ataque con texto llano conocido.
- Ataque con texto llano escogido.

En el criptosistema (P,C,K,E,D) denotamos por $P_{rp}(\mathrm{p})$ a la probabilidad de que un texto llano p aparezca en P. De igual manera $P_{rk}(\mathrm{k})$ es la probabilidad de encontrar una clave en P.

Entonces la probabilidad de que un texto p aparezca codificado por la clave k es denotada por $P_r(p,k) = P_{rp}(p) \cdot P_{rk}(k)$ Función que define una distribución de probabilidad en el espacio producto $\mathrm{P} \times \mathrm{K}$.

Decimos que un criptosistema es secreto perfecto si.

$$P_r(p|c) = P_r(p),\ \forall p \in \mathrm{P},\ \forall c \in \mathrm{C}. \tag{2.4}$$

Es decir, que la probabilidad de un cierto texto cifrado y la probabilidad de que un texto llano haya sido cifrado son independientes.

El Teorema de Shannon dicta que:
El criptosistema tendrá seguridad perfecta, sí y sólo sí la distribución de probabilidad en el espacio de llaves es uniforme. Además, sí para cualquier texto llano p y texto cifrado c, existe una llave única k con $E_k(p) = c$.

2.2 Fundamentos físicos y matemáticos de la criptografía cuántica.

Las bases que se manejan en la rama de la criptografía cuántica parten de principios e ideas propuestos alrededor del año 1984 [4, 5]. El objetivo de la criptografía cuántica es aprovechar los principios peculiares que la mecánica cuántica propone y describe para lograr un mayor grado de seguridad al momento de compartir información de manera cifrada.

2.2.1 Oscilador Armónico Cuántico.

El Hamiltoniano del oscilador armónico clásico es,

$$H = \frac{p^2}{2m} + \frac{m\omega^2}{2}x^2. \tag{2.5}$$

Entonces considerando las ecuaciones de Hamilton,

$$\dot{x} = \frac{\partial H}{\partial p}; \quad \dot{p} = -\frac{\partial H}{\partial x} \tag{2.6}$$

el problema se resuelve de manera sencilla pues

$$m\ddot{x} + kx = 0. \tag{2.7}$$

con la soluciones:

$$x(t) = A\cos(\omega t) + B\sin(\omega t). \tag{2.8}$$

donde $\omega^2 = \frac{k}{m}$; k siendo la constante de restitución.

Antes de plantear el oscilador armónico cuántico, recordaremos un poco la notación de los corchetes de Poisson para coordenadas generalizadas en un sistema con mecánica Hamiltoniana que evoluciona en el tiempo,

$$[u, v] = \frac{\partial u}{\partial q_i}\frac{\partial v}{\partial p_i} - \frac{\partial u}{\partial p_i}\frac{\partial v}{\partial q_i}, \tag{2.9}$$

que tienen las siguientes propiedades de conmutación,

$$\left[q_j, q_k\right]_{q,p} = 0 = \left[p_j, p_k\right]_{q,p}, \tag{2.10}$$

y también,

$$\left[p_j, q_k\right]_{q,p} = \delta_{jk} = -\left[q_j, p_k\right]_{q,p}. \tag{2.11}$$

Entonces, se describe el oscilador armónico cuántico para 1 dimensión (1D), donde tenemos los operadores $\hat{x},\ \hat{p}$ que cumplen la relación de conmutación $[\hat{x}, \hat{p}] = i\hbar$. Los operadores en este espacio de Hilbert son de la forma,

$$\hat{f} = f(\hat{x}, \hat{p}), \tag{2.12}$$

donde el operador debe ser Hermitiano, cumpliendo $'\hat{f} = \hat{f}^{\dagger}$.

Se resuelve la ecuación de Schrödinger para el Hamiltoniano 2.5 que se plantea de la siguiente forma,

$$\hat{H} = \frac{\hat{p}^2}{2m} + \frac{m\omega^2}{2}\hat{x}^2, \tag{2.13}$$

es decir, queremos resolver,

$$\hat{H}\Psi(x,t) = i\hbar\frac{\partial\Psi(x,t)}{\partial t}, \tag{2.14}$$

donde $\Psi(x,t)$ es la función de onda, entonces como el Hamiltonieano no depende del tiempo la solución es,

$$\Psi(x,t) = \phi(x)e^{-\frac{i}{\hbar}Et}, \tag{2.15}$$

y $\phi(x)$ es solución a

$$-\frac{\hbar^2}{2m}\phi'' + \frac{1}{2}m\omega^2x^2\phi(x) = E\phi(x). \tag{2.16}$$

Para obtener $\phi(x)$ utilizaremos un método algebraico donde la ecuación de eigenvalores es la siguiente:

$$\hat{H}(\phi) = \frac{1}{2m}[\hat{p}^2 + (m\omega\hat{x})^2)]\phi = E(\phi), \tag{2.17}$$

si $\hat{x}$ y $\hat{y}$ fuesen números se podrían factorizar de la siguiente manera,

$$\hat{u}^2 + \hat{v}^2 = (\hat{v} + i\hat{u})(\hat{v} - i\hat{u}). \tag{2.18}$$

Sin embargo, $\hat{x}$ y $\hat{y}$ no conmutan, por lo que es necesario introducir los operadores de creación, aniquilación y el estado numero así podemos realizar una especie de factorización dentro de un espacio de Hilbert,

$$\hat{a} \equiv \frac{1}{\sqrt{2}}\left(\hat{u} + \frac{d}{du}\right) \quad \text{Operador Creación,} \tag{2.19}$$

$$\hat{a}^{\dagger} \equiv \frac{1}{\sqrt{2}}\left(\hat{u} - \frac{d}{du}\right) \quad \text{Operador Aniquilación,} \tag{2.20}$$

$$\hat{N} = \hat{a}\hat{a}^\dagger \quad \text{Operador Número.} \tag{2.21}$$

Podemos entonces designar los operadores para la ecuación de eigenvalores 2.17 de la siguiente manera,

$$\begin{aligned} \hat{a} &= \sqrt{\frac{1}{2\hbar m\omega}}(m\omega\hat{x} + i\hat{p}), \\ \hat{a}^\dagger &= \sqrt{\frac{1}{2\hbar m\omega}}(m\omega\hat{x} - i\hat{p}). \end{aligned} \tag{2.22}$$

Calculando los productos $\hat{a}\hat{a}^\dagger$ y $\hat{a}^\dagger\hat{a}$ se obtiene:

$$\hat{a}\hat{a}^\dagger = \frac{1}{2\hbar m\omega}\left[\hat{p}^2 - (m\omega\hat{x})^2\right] + \frac{i}{2\hbar}(\hat{x}\hat{p} - \hat{p}\hat{x}), \tag{2.23}$$

$$\hat{a}^\dagger\hat{a} = \frac{1}{2\hbar m\omega}\left[\hat{p}^2 + (m\omega\hat{x})^2\right] + \frac{i}{2\hbar}(\hat{x}\hat{p} - \hat{p}\hat{x}), \tag{2.24}$$

pero se tiene que $(\hat{x}\hat{p} - \hat{p}\hat{x}) = [\hat{x}, \hat{p}] = i\hbar$, por lo tanto,

$$\hat{a}\hat{a}^\dagger = \frac{1}{\hbar\omega}\hat{H} + \frac{1}{2}, \tag{2.25}$$

$$\hat{a}^\dagger\hat{a} = \frac{1}{\hbar\omega}\hat{H} - \frac{1}{2}. \tag{2.26}$$

A continuación se calcula el conmutador,

$$\left[\hat{a}, \hat{a}^\dagger\right] = \hat{a}\hat{a}^\dagger - \hat{a}^\dagger\hat{a} = 1. \tag{2.27}$$

Usando el conmutador con las ecuaciones 2.25 y 2.26 se escribe el Hamiltoniano como:

$$H = \hbar\omega\left(\hat{a}^\dagger\hat{a} + \frac{1}{2}\right). \tag{2.28}$$

2.2.2 Principio de incertidumbre

El principio de incertidumbre de Heissenberg dicta que es imposible determinar, con precisión absoluta y de forma simultánea, el valor la posición y momento de un sistema elemental [6, 7]. Entonces se entiende que a partir de este principio es imposible determinar de forma precisa la posición y la cantidad de movimiento de una partícula de manera simultanea.

Hay dos formas de representar esta incertidumbre, una de ellas es a la que Heissenberg llegó por su cuento y una mas actual es la forma moderna,

$$\Delta_\Psi \hat{A} \Delta_\Psi \hat{B} = \frac{1}{2} |\langle \Psi | \left[\hat{A}, \hat{B}\right] | \Psi \rangle |. \tag{2.29}$$

Entonces para la posición y el momento de una partícula podemos establecer las siguientes relaciones de conmutación. Para la deducción moderna, partimos de la desviación estándar denotada de la siguiente manera,

$$\begin{aligned} \left(\Delta_\Psi \hat{A}\right)^2 &= \langle \Psi | \left(\hat{A} - \left\langle \hat{A} \right\rangle\right)^2 | \Psi \rangle \\ = \langle \Psi | \hat{A}^2 | \Psi \rangle - \langle \Psi | \hat{A} | \Psi \rangle^2 &= \left\langle \hat{A}^2 \right\rangle_\Psi - \left\langle \hat{A} \right\rangle_\Psi^2, \end{aligned} \tag{2.30}$$

suponiendo la siguiente forma para el valor medio del producto de dos operadores hermiticos $\hat{C}$ y $\hat{D}$

$$\begin{aligned} \langle \Psi | \hat{C}\hat{D} | \Psi \rangle &= x + iy \\ &\text{entonces,} \\ \langle \Psi | \left[\hat{C}, \hat{D}\right] | \Psi \rangle &= 2iy, \\ \langle \Psi | \{\hat{C}, \hat{D}\} | \Psi \rangle &= 2x, \end{aligned} \tag{2.31}$$

$$\begin{aligned} |\langle \Psi | \left[\hat{C}, \hat{D}\right] | \Psi \rangle|^2 + |\langle \Psi | \{C, D\} | \Psi \rangle|^2 &= \\ = 4y^2 + 4x^2 = 4 |\langle \Psi | \hat{C}\hat{D} | \Psi \rangle|^2, & \\ \Rightarrow |\langle \Psi | \left[\hat{C}, \hat{D}\right] | \Psi \rangle|^2 \leq 4 |\langle \Psi | \hat{C}\hat{D} | \Psi \rangle|^2, & \end{aligned} \tag{2.32}$$

utilizando la desigualdad de Cauchy-Schwarz se obtiene,

$$|\langle \Psi | \left[\hat{C}, \hat{D}\right] | \Psi \rangle|^2 \leq 4 \langle \Psi | \hat{C}^2 | \Psi \rangle \langle \Psi | \hat{D}^2 | \Psi \rangle, \tag{2.33}$$

redefiniendo,

$$\left.\begin{aligned} \hat{C} &= \hat{A} - \left\langle \hat{A} \right\rangle_\psi \\ \hat{D} &= \hat{B} - \left\langle \hat{B} \right\rangle_\psi \end{aligned}\right\} \rightarrow \begin{cases} \left\langle \hat{C} \right\rangle^2 = \Delta_\psi^2 \hat{A} \\ \left\langle \hat{D} \right\rangle^2 = \Delta_\psi^2 \hat{B} \\ \left[\hat{A}, \hat{B}\right] = \left[\hat{C}, \hat{D}\right] \end{cases} \tag{2.34}$$

se sigue que,

$$\begin{aligned} |\langle\psi|[\hat{A},\hat{B}]|\psi\rangle|^2 &= 4\Delta_\psi^2\hat{A}\Delta_\psi^2\hat{B}, \\ |\langle\psi|[\hat{A},\hat{B}]|\psi\rangle| &= 2\Delta_\psi\hat{A}\Delta_\psi\hat{B}, \end{aligned} \tag{2.35}$$

aplicando para para $[\hat{X},\hat{P_x}] = i\hbar$ se obtiene,

$$|\langle\psi|[\hat{X},\hat{P_x}]|\psi\rangle| = \frac{\hbar}{2}, \tag{2.36}$$

entonces se tiene,

$$\Delta_\psi\hat{X}\Delta_\psi\hat{P_x} \geq \frac{\hbar}{2}. \tag{2.37}$$

2.2.3 Teorema de no clonación

Este teorema dicta que es imposible copiar el estado desconocido de un sistema cuántico, debido a que en el intento de obtener información acerca de este, la misma medición provoca su modificación [6, 7]. Esto hace que sea una ley de la naturaleza la que nos garantice la seguridad de los protocolos.

La demostración de este Teorema se plantea comenzando con un estado de un sistema que llamaremos $|\alpha\rangle$, tomaremos también un estado inicial de un segundo sistema que llamaremos $|\beta\rangle$. Nos interesa que el estado final del sistema en su conjunto sea $|\alpha\rangle \bigotimes |\beta\rangle$. Consideramos una base del estado $|\alpha_i\rangle$, que se descompone de la siguiente forma al copiarse,

$$|\alpha\rangle = \sum_i \alpha_i |\alpha_i\rangle . \tag{2.38}$$

Definimos un operador unitario y lineal $\hat{U}$ que representara el proceso de copiado, si el proceso de clonación funciona $\hat{U}$ debe de duplicar los vectores de la base,

$$\hat{U}|\alpha_i\rangle \otimes |\beta\rangle = |\alpha_i\rangle \otimes |\alpha_i\rangle . \tag{2.39}$$

Aplicamos el operador $\hat{U}$ al estado completo,

$$\begin{aligned} \hat{U}|\alpha\rangle \otimes |\beta\rangle &= \sum_i \alpha_i \hat{U}|\alpha_i\rangle \otimes |\beta\rangle \\ &= \sum \alpha_i |\alpha_i\rangle \otimes |\alpha_i\rangle . \end{aligned} \tag{2.40}$$

Pero este estado no es $|\alpha\rangle \otimes |\alpha\rangle$ que se descompone como:

$$|\alpha\rangle \otimes |\alpha\rangle = \sum \alpha_i \alpha_j |\alpha_i\rangle \otimes |\alpha_j\rangle \rightarrow \alpha_i \alpha_j = \alpha_i \delta_{ij}. \tag{2.41}$$

Por lo tanto, la ecuación 2.40 no es igual a la ecuación 2.41, mostrando así que no se puede copiar de manera idéntica un estado de un sistema a otro

2.2.4 Superposición

La superposición en mecánica cuántica se refiere a la propiedad de una partícula o sistema físico de existir en todos su posibles estados teóricos a la vez, es decir un sistema cuántico puede poseer simultáneamente dos o más valores de una cantidad observable, valores que pueden ser incluso diferentes. Para entender esto supongamos un pulso de luz, el cual contiene un solo fotón, si hacemos que dicho haz de luz pase por un divisor de haz tendremos dos pulsos con trayectorias distintas. Sin embargo, el pulso sigue constando de un solo fotón. Lo que obtenemos es un fotón que esta en dos puntos distintos del espacio al mismo tiempo. A esto se le llama superposición y tenemos una probabilidad de $\frac{1}{2}$ de encontrar al fotón en cualquiera de las posiciones en las que se encuentra.

2.2.5 Entrelazamiento

Este principio define una propiedad de un par de sistemas cuánticos que se encuentran ligados, de manera que ciertas modificaciones sobre alguno de ellos modificaran al otro. Este principio surge de la no separabilidad, la cual es una propiedad matemática que dicta que no es posible factorizar la distribución de probabilidad estadística de dos variables estocásticas como un producto de distribuciones independientes.

2.2.6 Bases Ortogonales

Una base ortogonal es un conjunto de vectores unitarios cuyo producto escalar es 0, es decir que no se puede escribir en términos del otro [8]. El algoritmo BB84 trabaja con este principio, de manera que

prepara al fotón usando un par de bases de polarización con estados ortogonales, que a su vez estos estados no son ortogonales entre bases diferentes, de manera convencional se eligen las siguientes bases,

- $$\hat{B}_{+} = \frac{1}{2} |{\rightarrow}\rangle \langle{\rightarrow}| + \frac{1}{2} |{\uparrow}\rangle \langle{\uparrow}|, \tag{2.42}$$

- $$\hat{B}_{\times} = \frac{1}{2} |{\nearrow}\rangle \langle{\nearrow}| + \frac{1}{2} |{\nwarrow}\rangle \langle{\nwarrow}|, \tag{2.43}$$

y podemos relacionar los estados con las siguientes ecuaciones de cambio de base (y su inversa),

- $$|\nearrow\rangle = \frac{1}{\sqrt{2}}(|\rightarrow\rangle + |\uparrow\rangle), \tag{2.44}$$

- $$|\nwarrow\rangle = \frac{1}{\sqrt{2}}(|\rightarrow\rangle - |\uparrow\rangle), \tag{2.45}$$

Ya que las dos bases se tienen elegidas, por convención en la base $\hat{B}_{+}$ se toma como bit 1 a la polarización vertical y como bit 0 a la polarización horizontal. De la misma forma, en la base $\hat{B}_{\times}$ para el bit 1 se elige la polarización a $\frac{\pi}{4}$ radianes y el bit 0 a $\frac{3\pi}{4}$ radianes.
Para la medición de estas polarizaciones, se necesita un polarizador de luz , dejando pasar solo las ondas de polarización horizontal y reflejando las de polarización vertical. Con la ayuda de un retardador de media onda este sistema puede distinguir entre polarización diagonal y polarización antidiagonal.

Capitulo 3

Algoritmos de computación cuántica

3.1 Protocolos de Encriptación

Existen diversos protocolos de encriptación basados en principios de mecánica cuántica. Como lo son los de variable discreta donde la codificación ocurre mediante una polarización, los de variable continua que utilizan estados gaussianos [9], protocolos de distribución de fase y otros mas extensos como lo son los protocolos de modos superiores.

El algoritmo mas sencillo de implementar es el algoritmo BB84 de dos bases no ortogonales, en el cual se utiliza la polarización de un paquete de información, en este caso un solo fotón, para poder codificar el mensaje que se esta enviando.

3.1.1 Protocolo BB84

El algoritmo mas sencillo de implementar es el algoritmo BB84 [10] de dos bases no ortogonales, en el cual se utiliza la polarización de un paquete de información. En este caso un solo fotón, para poder codificar el mensaje que se esta enviando.

Para poder ejemplificar el funcionamiento de este algoritmo, tomaremos como el emisor a una persona A que recibirá el nombre de Alice, y una persona B quien recibirá el nombre de Bob y este sera el receptor del mensaje que Alice envíe por el canal clásico y de la generación de la claves secretas a través del canal cuántico. También tendremos a un tercer elemento llamado Eve (proveniente del ingles Eavesdroper), quien tendrá acceso a los canales por donde Alice y Bob se comuniquen. Podrá tener acceso total al canal cuántico con la posibilidad de operar sobre el canal de cualquier manera, siempre y cuando no viole las reglas de la mecánica cuántica, pues no se podrá duplicar la información debido al principio de no clonación y cualquier lectura de la información modificara su estado.

Para comenzar a transmitir un mensaje, Alice envía los paquetes de información codificados conforme a cuatro estados de polarización. Estos cuatro estados se agrupan formando dos bases con estados de polarización ortogonales cada una. De esta forma, un estado de polarización queda perfectamente determinado al pasar por un polarizador correspondiente a su base, mientras que el resultado será totalmente aleatorio si pasa por un polarizador de otra base. El objetivo de esta primera fase del protocolo es que Alice prepare una secuencia aleatoria de n estados, obtenidos a partir de dos secuencias, también aleatorias, una de bases y otra de valores para la clave.

Una vez que Alice tiene listo cada estado de la clave, los envía por el canal cuántico y Bob intenta interpretar cada estado eligiendo al azar una de las dos bases existentes, ($\hat{B}_+$ o $\hat{B}_\times$). Debido a que las bases que se eligen son aleatorias, la probabilidad de que coincidan es de $\frac{1}{2}$, entonces solo la mitad de los estados que Alice envió, serán interpretados correctamente por Bob.
En cuanto al resto de los estados, aunque la base utilizada no haya sido correcta, la probabilidad de acierto también sera de $\frac{1}{2}$ por lo que la información total que Bob obtiene de la clave generada en Alice es de $\frac{3}{4}$.

Pasos del algoritmo

---

1. Alice genera una secuencia de valores aleatorios, esta será la clave que compartirá con Bob.
2. Alice genera otra secuencia aleatoria, ahora con las bases usadas para la codificación de la clave ya generada.
3. Alice codifica cada valor de la clave con la base correspondiente y envía la secuencia a Bob por un canal cuántico.
4. Bob genera una secuencia aleatoria con las bases que usará para decodificar la secuencia de estados que envíe Alice.
5. Bob mide cada estado recibido de acorde a la base correspondiente de la base que el generó.
6. Bob envía a Alice la secuencia de bases que utilizó por medio de un canal público autentificado.
7. Alice compara la secuencia de bases que ella utilizó para codificar la clave, con la secuencia que Bob le envió por el canal público.

Deshecha las mediciones con bases diferentes y conserva donde coinciden.

8. Ambos comparten una secuencia de valores formada por los valores que han coincidido.

9. A continuación se corrigen errores, se estima si hubo un intruso, y se amplifica la privacidad.

El ultimo paso puede verse como un subproceso adicional al protocolo BB84, pues este también tiene sus propios procesos y algoritmos para realiza ajustes de error, calibración y privacidad.

3.1.2 Protocolo SARG04

En este algoritmo los estados se generan de la misma forma que en el BB84, la diferencia se encuentra en la forma de cotejamiento entre estados [11], pues se tienen dos detectores y al momento de corregir errores y capturar los estados validos se hace de la siguiente forma.

Los estados generados en el SARG04 se denotan de la siguiente forma:

$$|\psi\rangle = \bigotimes_{i=1}^{n} \left|\psi_{e_i b_i}\right\rangle \tag{3.1}$$

cada estado son los i-esimos qubits con e_i, b_i de estado (e) y bit (b) respectivamente, entonces $e_i b_i$ nos da un índice para los siguientes cuatro estados.

$$\begin{aligned} |\psi_{01}\rangle &= |0\rangle, \\ |\psi_{11}\rangle &= |1\rangle \\ |\psi_{00}\rangle = |+\rangle &= \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \\ |\psi_{10}\rangle = |-\rangle &= \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \end{aligned} \tag{3.2}$$

Lo cual los estados $|\psi_{01}\rangle, |\psi_{11}\rangle$ forman el bit 1, los estados $|\psi_{00}\rangle, |\psi_{10}\rangle$ forman el bit 0.

Estados $\|\psi\rangle$		
0	$\|\psi_{01}\rangle$	$\|H_0\rangle$
1	$\|\psi_{11}\rangle$	$\|V_1\rangle$
2	$\|\psi_{00}\rangle$	$\|D_1\rangle$
3	$\|\psi_{10}\rangle$	$\|A_0\rangle$

Tabla 3.1. Estados de cotejamiento del protocolo SARG04

Ahora lo interpretaremos como estados horizontal (H), vertical (V), diagonal (D), y antidiagonal (A). de los cuales H y V hacen una base ortogonal al igual que D y A.Esto se muestra en la Tabla 3.1 La forma de detección del SARG04 no es por las bases ortogonales, si no por combinación de estados de ambas bases, por lo cual tendremos combinaciones como se muestra en la Tabla 3.2.

Estados de cotejamiento	
0	$\|H_0\rangle\langle D_1\|$
1	$\|H_0\rangle\langle A_0\|$
2	$\|V_0\rangle\langle D_1\|$
3	$\|V_0\rangle\langle A_0\|$

Tabla 3.2. Estados de cotejamiento del protocolo SARG04

Entonces tendremos detectores de Estado y Base que se cotejan como se muestra en la Tabla 3.3

Estados y Bits		
Estado	Bit(Base)	Cotejamientos
H-0	1	0,1
V-1	1	2,3
D-0	0	0,2
A-1	0	1,3

Tabla 3.3. Definición de Bits para el protocolo SARG04

El cotejamiento se refleja en la Tabla 3.4, donde 1 es un bit acertado, 0 es un errado y - es uno descartado

Estado Alice	Base Bit Alice	Base Bob	Detector 0 (H|D)	Detector 1 (V|A)	Estado de cotejamiento	Resultado	Bit de Bob
H -0	1	1	H		0(H,D)	X	-
					1(H,A)	X	-
				V	0(H,D)	E	0
					1(H,A)	E	0
			H	V	0|1	X	-
		0	D		0(H,D)	X	-
					1(H,A)	C	1
				A	0(H,D)	C	1
					1(H,A)	X	-
			D	A	0|1	X	-
V - 1	1	1	H		2(V,D)	E	0
					3(V,A)	E	0
				V	2(V,D)	X	-
					3(V,A)	X	-
			H	V	2|3	X	-
		0	D		2(V,D)	X	-
					3(V,A)	C	1
				A	2(V,D)	C	1
					3(V,A)	X	-
			D	A	2|3	X	-
D - 0	0	1	H		0(H,D)	C	1
					2(V,D)	X	-
				V	0(H,D)	X	-
					2(V,D)	C	1
			H	V	0|2	X	-
		0	D		0(H,D)	X	-
					2(V,D)	X	-
				A	0(H,D)	E	0
					2(V,D)	E	0
			D	A	0|2	X	-
A - 1	0	1	H		1(H,A)	C	1
					3(V,A)	X	-
				V	1(H,A)	X	-
					3(V,A)	C	1
			H	V	1|3	X	-
		0	D		1(H,A)	E	0
					3(V,A)	E	0
				A	1(H,A)	X	-
					3(V,A)	X	-
			D	A	1|3	X	-

Tabla 3.4. Algoritmo de decisión para la lectura de Bits del protocolo SARG04, en la columna Resultado X significa descartado, E significa error y C significa correcto

3.2 Métodos de reconciliación

Para realizar la parte de reconciliación de la llave cuántica, existen varios métodos, algoritmos y protocolos que pueden usarse una vez que se completa la parte de comunicación con el protocolo BB84 o con el SARG04.

3.2.1 Cascade

Propuesto en 1993 por Bennet Brassard [12, 13], tiene como propósito detectar y corregir algún error remanente de errores en los bits de la llave con ruido que Bob recibe. Supongamos que Alice y Bob ya han compartido mensaje. Alice tiene la distribución de llave correcta. Mientras que Bob posee una llave similar, pero con ruido generado de errores de bit. Alice y Bob utilizan el protocolo Cascade para limpiar esos errores.

El método es bastante simple pues comienza a comparar bloques de tamaño idéntico, extraídos de cada una de las secuencias de bits.

A continuación, se calcula la paridad uno a uno y utilizando una búsqueda binaria, encuentra la cantidad de errores pero corrige solo uno. Para completar una secuencia completa tiene que trabajar en paralelo o de forma iterada hasta que no se detecten mas errores, por esto el método es muy costoso y tardado.

3.2.2 Winnow

El método Winnow, propuesto por Buttler et al. [14] sigue el mismo principio de Cascade, solamente que la búsqueda de errores se hace mediante códigos Hamming, los cuales consisten en una matriz generadora G y una matriz de paridad H definida por

$$H_{i,j}^{m} = \left\lfloor \frac{j}{2i-1} \right\rfloor \text{mod} 2 \tag{3.3}$$

Donde m representa la cantidad de paridades posibles, i, j son los índices de columna y renglón respectivamente

La iteración de bloques en cada secuencia es similar a Cascade, sin embargo si aquí se encuentra mas de un error por bloque, este bloque se divide en sub-bloques para aislar el error y no meter mas ruido al momento de corregir. Sin embargo una cantidad mas pequeña de tamaño en los bloques significa una cantidad mas grande de bloques, lo que sugiere un tiempo de computo mas largo.

3.2.3 Low Density Parity Check(LDPC)

El Low Density Parity Check es un codigo de correción lineal de errores, un método para transmitir un mensaje a través de un canal ruidoso [15]. El método es construido usando una matriz de chequeo de paridad H, que suele ser generada de manera aleatoria, y una matriz generatriz G. La matriz de chequeo de paridad contiene una cantidad arbitraria de 1s por columna y fila, sujetos al limitante de $k = \frac{n \cdot j}{m}$, donde j es el numero de 1s por columna y k es el numero de 1s por fila para una matriz $m \times n$

Para corregir una distribución de llave cuántica, el proceso de un código LPDC es similar a un Winnow de un solo bloque. Para esto, Alice calcula un peso de errores usando códigos Hamming y es lo único que le otorga a Bob, no se subdivide la llave ni existe un intercambio de paridad. Con esto Bob calcula su versión de la llave. Para conocer

la ubicación del error, Bob utiliza un codigo de decodificación, uno de ellos es el algoritmo de suma de producto o propagación de creencia.

Este algoritmo es un algoritmo iterativo y de transferencia de mensaje donde las creencias son pasadas entre los nodos para satisfacer los chequeos de paridad recibidos por el peso inicial. Para entender esto observemos en la Figura 3.1 que la matriz H puede ser representada por un grafo de Tanner, los cuales son representaciones de código con nodos bipartidos de variables y limitaciones.

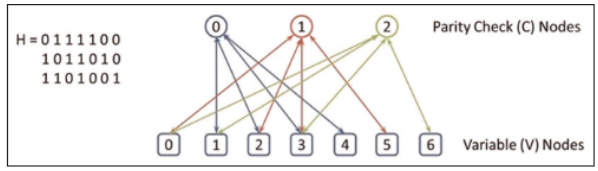

Figura 3.1. Grafo de Tanner de una matriz de chequeo de paridad

Los nodos de chequeo de paridad representan las filas de la matriz y los nodos de variables representan los bits recibidos de la llave. En la primera iteración del algoritmo de propagación de creencias, cada nodo de variable (V) enviara un mensaje a cada uno de los nodos de chequeo de paridad (C) al que esta conectado, representando la creencia de que el bit que se leyó es correcto. Así, los nodos C pueden comparar la información que reciben con la de otros nodos, a continuación Bob calcula el peso de su llave, usando códigos Hamming, y la misma matriz de paridad que Alice usó. Si el peso concuerda con el de Alice, se concluye que todos los errores se corrigieron, de lo contrario se vuelve a iterar la matriz de chequeo de paridad.

3.3 Esfera de Bloch

Para poder representar los qubits de manera geométrica, se usa la esfera de Bloch (Figura 3.2), la cual es una representación en el espacio de estados puros de un sistema cuántico de dos niveles. En sus polos tiene los dos niveles o estados posibles dentro del sistema, que para el caso de un qubit es $|0\rangle$ o $|1\rangle$.

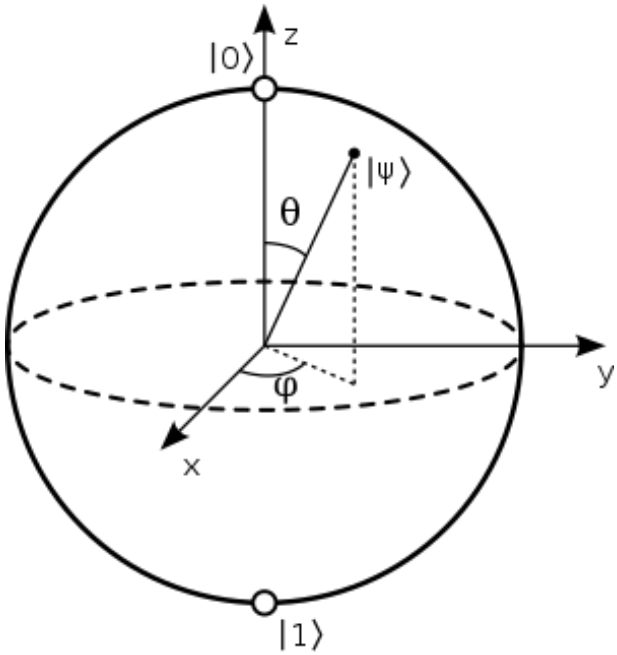

Figura 3.2. Esfera de Boch que representa una distribución de los qubits

Cualquier punto de la esfera de Bloch es un estado cuántico o qubit que se puede expresar como,

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle, \tag{3.4}$$

con $0 \leq \theta \leq \pi$ y $\leq \theta \leq 2\pi$.

3.4 Algoritmos cuánticos

Los qubits se aprovechan en los algoritmos cuánticos que resuelven tareas y problemas clásicos a una velocidad y eficiencia mayor. En estos algoritmos suele hablarse de una maquina llamada oracle u oráculo, que propuso Alang Turing como una caja negra que realiza operaciones de decisión. Esta maquina hipotética y abstracta es un buen referente para los algoritmos cuánticos pues muchas veces no se sabe con certeza que ocurre en los cálculos de una computadora cuántica debido a la naturaleza de la misma.

3.4.1 Algoritmo de Deutsch-Jozsa

El algoritmo de Deutsch-Jozsa [16] es un algoritmo cuántico determinista, que muestra lo exponencialmente rápida que es la computación cuántica con la clásica. En este algoritmo se nos presenta un problema a resolver, en el cual se tiene una computadora cuántica que funciona a manera de caja negra, como una maquina oráculo, e implementa una función que realiza la operación de $f : \{0,1\}^n \rightarrow \{0,1\}$. La función toma n valores binarios como entrada y produce ya sea un 0 o un 1 como salida. La distribución de las salidas puede ser constante (es decir todos 0 o todos 1), o uniforme (la mitad 1 y la mitad 0). El objetivo del algoritmo es verificar si la función f es constante o uniforme usando el oráculo.

Para un algoritmo clásico convencional, donde n es el numero de bits, se necesitarían $2^{n-1}+1$ evaluaciones en el peor de los casos, lo cual es bastante demandante para una computadora clásica. En el algoritmo de Deutsch-Jozsa se necesitaría solo una evaluación de f para verificar la distribución de las salidas. Para que el algoritmo de Deutsch-Jozsa funcione, la maquina oráculo que procesa $f(x)$ debe ser cuántico y coherente. También debe dejar una copia de x al final del proceso.

El algoritmo comienza con el estado $n+1$ de los bits $|0\rangle^{\otimes n}|1\rangle$. Entonces los primeros n bits estarán en el estado $|0\rangle$ y el ultimo estará en el estado $|1\rangle$. Entonces se aplica una transformada de Hadamard (La compuerta de Hadamard es un caso 2×2 de esta transformada, así que el funcionamiento generalizado es análogo), de esta forma obtenemos el estado,

$$\frac{1}{\sqrt{2^{2n+1}}}\sum_{x=0}^{2^n-1}|x\rangle\left(|0\rangle-|1\rangle\right). \tag{3.5}$$

Tenemos entonces la función f, que es implementada como una máquina oráculo cuántica, mapea el estado $|x\rangle|y\rangle \rightarrow |x\rangle|y\oplus f(x)\rangle$ con $\oplus$ como suma en base binaria. Aplicando esto obtenemos,

$$\frac{1}{\sqrt{2^{2n+1}}}\sum_{x=0}^{2^n-1}|x\rangle\left(|f(x)\rangle-|1\oplus f(x)\rangle\right). \tag{3.6}$$

Para cada $x, f(x)$ es 0 o 1, cuando se prueba para cada una de las posibilidades obtenemos el estado,

$$\frac{1}{\sqrt{2^{2n+1}}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle (|0\rangle - |1\rangle). \tag{3.7}$$

Podemos ignorar el ultimo qubit $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ por lo que obtenemos,

$$\frac{1}{\sqrt{2^{2n+1}}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle . \tag{3.8}$$

Nuevamente se aplica una transformada de Hadamard a cada qubit y se obtiene,

$$\frac{1}{\sqrt{2^{2n+1}}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle \left[\frac{1}{\sqrt{2^{2n+1}}} \sum_{y=0}^{2^n-1} (-1)^{x\cdot y} |y\rangle \right] = \frac{1}{2^n} \sum_{y=0}^{2^n-1} \left[\sum_{x=0}^{2^n-1} (-1)^{f(x)} (-1)^{x\cdot y} \right], \tag{3.9}$$

donde $x = x_0y_0 \oplus x_1y_1 \oplus \cdots \oplus x_{n-1}y_{n-1}$ es la suma binaria del producto bit a bit entre la copia y y la entrada transformada x.

Finalmente se examina la probabilidad de medir $|\otimes n\rangle$,

$$\left| \frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(x)} \right|^2 . \tag{3.10}$$

Si al evaluarse se obtiene un 1, entonces $f(x)$ es constante, y si se obtiene 0, $f(x)$ es uniforme. La Figura 3.3 es un ejemplo gráfico del funcionamiento de este algoritmo.

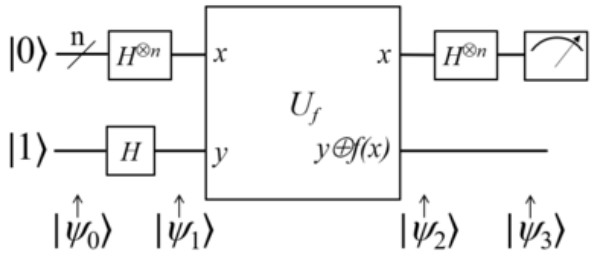

Figura 3.3. Circuito del algoritmo de Deutsch-Jozsa usando compuertas lógicas cuánticas.

3.4.2 Algoritmo de Shor

El algoritmo de Shor es un algoritmo cuántico de tiempo polinómico usado para la factorización de enteros [17], el cual resuelve el problema

de la factorización de números primos de un entero N. El algoritmo de Shor consiste en dos partes:

1. Una reducción de un problema de factorización a uno de ordenado que puede hacerse con una computadora clásica.
2. Un algoritmo cuántico para resolver el problema de ordenado.

En su parte clásica se tiene:

1. Seleccionar un numero aleatorio $\alpha \in 1 < \alpha < N$.
2. Calcular el mayor común divisor K de α y N.
3. si $K \neq 1$ entonces K es un factor no trivial de N. En tal caso se termina el algoritmo.
4. De otra forma se utiliza una subrutina cuántica para encontrar el periodo r de la siguiente función.

$$f(x) = a^x \bmod (N), \tag{3.11}$$

es decir, el numero entero mas pequeño r para el cual

$$f(x + r) = f(x). \tag{3.12}$$

5. Si r es impar ir de nuevo al paso 1.
6. Si $a^{\frac{r}{2}} \equiv -1 \bmod (N)$ ir de nuevo al paso 1.
7. Los factores de N no son el $\bmod (a^{\frac{r}{2}} \pm 1, N)$ se termina.

Subrutina Cuántica para encontrar el periodo:

1. Comenzar con un par de registros de qubits de entrada y salida en base 2, iniciados en

$$N^{\frac{-1}{2}} \sum_{x=0}^{N} -1 \left|x\right\rangle \left|0\right\rangle . \tag{3.13}$$

2. Construir $f(x)$ como función cuántica para aplicarla al estado anterior

$$N^{\frac{-1}{2}} \sum_{x=0}^{N} (-1) \left|x\right\rangle \left|f(x)\right\rangle . \tag{3.14}$$

3. Se aplica la transformada cuántica de Fourier a la entrada, esta transformada se define como

$$U_{QFT}\,|x\rangle = N^{\frac{-1}{2}} \sum_{y} e^{\frac{2\pi i x y}{N}}\,|y\rangle\,, \tag{3.15}$$

de donde obtenemos el estado siguiente

$$N^{-1} \sum_{x} \sum_{y} e^{\frac{2\pi i x y}{N}}\,|y\rangle\,|f(x)\rangle\,. \tag{3.16}$$

4. Convertir $\frac{y}{N}$ en una fracción irreducible y obtener el denominador r'

5. Se comprueba si $f(x) = f(x + r')$, si es asi terminamos.

6. Si no, Obtener mas candidatos a r usando valores cercanos a y o múltiplos de r'.

7. Si aun no se cumple, volver al paso 1 de la subrutina.

Capitulo 4

Soft Computing y Machine Learning

4.1 Soft Computing y Machine Learning

El soft computing es una rama de la ciencia de la computación cuyas ideas básicas se remontan a varios artículos de Lotfi Zadeh [18], entre los que destaca el famoso artículo sobre conjuntos difusos que publicó en 1965. Soft Computing difiere de la computación convencional en que es tolerante a la imprecisión, la incertidumbre y la verdad parcial, a fin de conseguir soluciones robustas y de bajo costo a diferentes problemas de la ciencia y la tecnología.

Entre sus elementos constitutivos tenemos a la lógica difusa, las redes neuronales, los algoritmos genéticos, las redes de creencia, la computación evolutiva, teorías del caos y algunas partes de la teoría del aprendizaje como el Deep Learning.

La importancia del surgimiento del Soft Computing ha sido tal que en 2001 se crea la revista especializada Applied Soft Computing, que tiene como editor en jefe a R. Roy (Cranfield University, UK), y a Lotfi Zadeh (University of California, Berkeley, USA) como editor honorario. Además, existe ya la World Federation on Soft Computing [19].

4.2 Redes Neuronales

Una red neuronal o neural network es una serie de algoritmos que ayudan a reconocer patrones y relaciones en un conjunto de datos a través de procesos que imitan la forma que un cerebro humano funciona [20]. En este sentido una red neuronal se puede referir a un sistema de neuronas artificiales o de materia orgánica ya presente en la naturaleza. Las redes neuronales se puede adaptar a una entrada cambiante, así la red genera el mejor resultado posible sin rediseñar el algoritmo.

Las redes neuronales dependen del entrenamiento de datos para aprender y mejorar su exactitud con el tiempo. Sin embargo, una vez

que sus algoritmos de aprendizaje están bien afinados y son certeros. Las redes neuronales se convierten en herramientas poderosas, permitiendo realizar diferentes tareas que parecen complejas a una gran velocidad.

Las redes neuronales están constituidas por diferentes capas de neuronas. Podemos seccionar las redes en tres partes: La capa de entrada, la capa oculta y la capa de salida (Figura 4.1). La capa oculta es donde suelen ocurrir todos los procesos, algoritmos y procesamiento de datos.

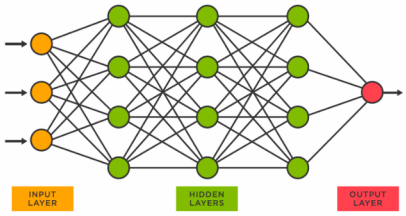

Figura 4.1. Ejemplo de una red neuronal, en amarillo la capa de entrada, en rojo la capa de salida y la capa oculta en verde. Imagen recuperada de https://www.asimovinstitute.org/neural-network-zoo/

4.2.1 Funciones de Activación

Para que las redes neuronales funcionen correctamente, se necesita una función de activación que procesa los datos de entrada y los transforma según la función definida, normalmente a los datos procesados que se mueven de neurona a neurona se denominan pesos, y todos estos pesos convergen en la neurona final de salida dando paso al resultado de la red neuronal.

4.2.1.1. Función Escalón

Esta función depende de un valor de umbral que decide si una neurona debería ser activada o no. La entrada que es alimentada a la función de activación se compara con este valor umbral (Δ), si el valor es mayor a este valor la neurona se activa, de otra manera no se activa.

$$f(x) = \begin{cases} 0 & \text{para} \quad x < \Delta, \\ 1 & \text{para} \quad x \geq \Delta. \end{cases} \tag{4.1}$$

4.2.1.2. Sigmoide

Esta función toma cualquier valor real como entrada y como resultado se obtiene un valor entre 0 y 1, mientras mas grande el valor la salida sera mas cercana a 1, y mientras mas pequeño o negativo sera mas cercano a 0,

$$f(x) = \frac{1}{1 + e^{-}x}. \tag{4.2}$$

4.2.1.3. Tangente Hiperbólica

La función tanh (tangente hiperbólica) es similar a la función sigmoide con la diferencia de obtener un resultado en un rango entre -1 y 1. Mientras mas grande la entrada, el valor de salida será mas cercano a 1 y mientras mas pequeña o negativa, el valor de salida será mas cercano a -1,

$$\tanh = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{4.3}$$

4.2.1.4. Signum

Esta función obtiene el signo de cualquier número real que se tome por entrada. Al combinarse con una función escalón podemos activar o no una neurona dependiendo el signo de la entrada,

$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{para} \quad x < 0, \\ 0 & \text{para} \quad x = 0, \\ 1 & \text{para} \quad x > 0. \end{cases} \tag{4.4}$$

4.3 Tree Parity Machine

Una máquina de árbol de paridad o Tree Parity Machine (TPM), es una red neuronal multicapa que no genera ciclos ni bucles [21, 22, 23, 24]. Las TPM consisten de una neurona de salida, K neuronas ocultas y $K \times N$ neuronas de entrada, la Figura 4.2 muestra el funcionamiento de esta red neuronal de manera gráfica. La máquina utiliza la función signo o signum de la ecuación 4.4 como función de activación.

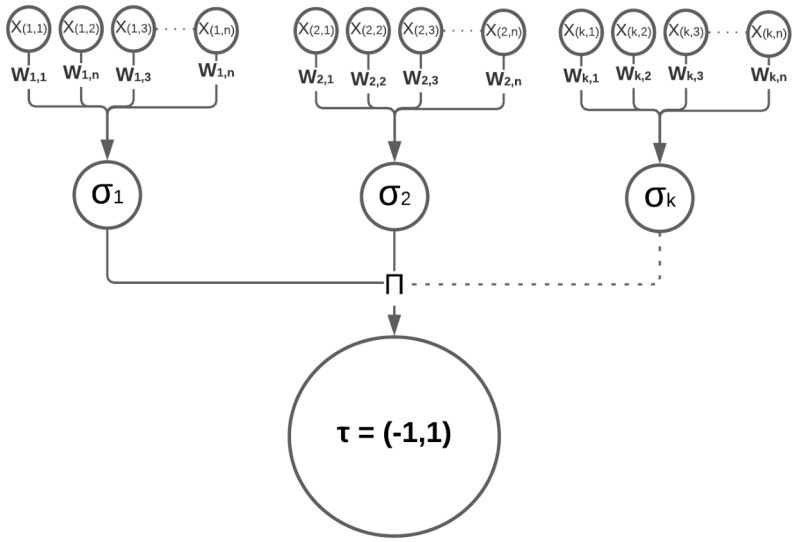

Figura 4.2. Ejemplo de como funciona una TPM, KN nueronas de entradas X con KN pesos W, K funciones sigmoide que despues se procesan para obtener el valor de τ.

Las entradas a la red neuronal solo pueden tomar tres valores que se filtran con la función signo,

$$x_{ij} \in [-1, 0, +1]. \tag{4.5}$$

Después de la entrada se generan pesos en las neuronas ocultas que solo pueden tomar valores de $w_{i,j} \in [-L, L]$, en este caso L es un valor de entrada que se define al construir la máquina. La salida de cada neurona oculta es calculada como una suma de la multiplicación de las entradas con estos pesos, denotada de la siguiente forma,

$$\sigma_i = \text{sgn}\left(\sum_{j=1}^{N} w_{i,j} x_{i,j}\right). \tag{4.6}$$

Si el producto escalar es 0, la salida de esta neurona se mapea a -1 para asegurarse que hay una salida binaria (-1 o 1). La salida de la red neuronal es calculada como la multiplicación de todos los valores obtenidos por las neuronas ocultas,

$$\tau = \prod_{i=1}^{K} \sigma_i. \tag{4.7}$$

El algoritmo para esta red neuronal es el siguiente:

1. Alice y Bob preparan su propia TPM.
2. Se inicializan pesos aleatorios.
3. Los siguientes pasos se ejecutan hasta que haya una sincronización.
   a) Se introduce el vector de entrada X.
   b) Se calculan los valores de las neuronas ocultas.
   c) Se calcula el valor de la neurona de salida.
   d) Se comparan ambos valores de salida.
      - si los valores son iguales, los pesos se pueden usar como llaves.
      - si no son iguales, repita desde el paso 3.

Una vez que ambos valores de salida τ son iguales, los pesos que fueron generados de manera aleatoria pueden usarse como llave entre ambas partes para poder tener un mensaje cifrado. Para que los pesos puedan sincronizarse es necesario usar diferentes reglas de aprendizaje, es decir como se conectan y comunican las neuronas entre si, tres de las mas comunes son la regla Hebbiana, la Anti-Hebbiana y los Random Walkers.

4.3.1 Aprendizaje Hebbiano

La teoría del aprendizaje Hebbiano (Hebbian Learning) introducida en 1949 por Donald Hebb, es una teoría de neurociencia que explica como las neuronas se activan y los caminos que estas eligen suelen ser los mismos para ciertas funciones [25], es decir, las rutas entre neuronas se repiten para cada función o resultado en especifico.

En su libro, The Organization of Behavior, Hebb postula lo que se conocería después como la regla Hebbiana, donde se asume que la repetición de los caminos entre neuronas tiende a inducir cambios celulares prolongados que se suman a una estabilidad en la ruta entre neuronas. Cuando el estimulo de una neurona A es bastante cercano para excitar a una neurona B repetidamente se disparan los estímulos hasta provocar el asentamiento de la ruta entre las neuronas A y B para este estimulo especifico. A partir de este postulado, se basan las reglas de aprendizaje en redes de neuronas artificiales

4.3.1.1. Regla Hebbiana

La regla Hebbiana, especifica que tanto el peso de una conexión entre dos neuronas debe de aumentar o disminuir, la regla funciona bien siempre y cuando las entradas sean ortogonales o no tengan correlación una de otra. La regla Hebbiana aplicada a una TPM seria denotada de la siguiente forma,

$$w_{kn}^{*} = v_L(w_{kn} + \sigma_{kn}x_{kn}\Theta(\sigma_k,\ \tau)\Theta(\tau^A,\ \tau^B)), \tag{4.8}$$

donde se relacionan los pesos de la ruta entre una neurona A y una neurona B, y se aplica una función signo σ_i para encontrar el signo de la entrada x_i, y finalmente se tiene una función Θ y una función v_L la cual limita los valores de la conexión a un rango $[-L,\ L]$

Función Θ

$$\Theta(,\tau) = \begin{cases} 0 & \text{para} \quad \sigma_k \neq \tau \\ 1 & \text{para} \quad \sigma_k = \tau \end{cases} \tag{4.9}$$

Función v_L

$$v_L(z) = \begin{cases} -L & \text{si} \quad z \leq -L \\ z & \text{si} \quad -L < z < L \\ L & \text{si} \quad z \geq L \end{cases} \tag{4.10}$$

4.3.1.2. Regla Anti-Hebbiana

La regla Anti-Hebbiana es una regla que también esta basada en el postulado de Donald Hebb, pero aplicada de manera inversa, es decir, aquí la activación entre neuronas es controlada y el efecto de activación de una neurona a otra es para cerrar un camino y encontrar otro, la regla se escribe de la siguiente forma,

$$w_{kn}^{*} = v_L(w_{kn} - \sigma_{kn}x_{kn}\Theta(\sigma_k,\ \tau)\Theta(\tau^A,\ \tau^B)). \tag{4.11}$$

Es similar a la regla Hebbiana 4.8, solo que con una resta entre el peso w_{kn} y la operación que genera el peso de la siguiente neurona.

4.3.1.3. Random Walker

En esta regla de aprendizaje, la ruta que se toma desde la entrada hasta la salida es aleatoria, por lo que se elimina le función signo y se elige la neurona siguiente de forma aleatoria. La regla se describe de la forma siguiente,

$$w_{kn}^{*} = v_L(w_{kn} + x_{kn}\Theta(\sigma_k,\ \tau)\Theta(\tau^A,\ \tau^B)). \tag{4.12}$$

4.3.2 Peso de Hamming

Cuando se escribe una TPM se piensa en que los pesos serán aleatorios y estos se usaran como llave. Sin embargo, cuando se tiene ya una secuencia de datos, caracteres, bits o cualquier información que se pueda usar como llave, para poder usar estas llaves se tienen que obtener los pesos que serán introducidos a la máquina.

Una forma de calcular estos pesos para una secuencia de bits es obtener los pesos de Hamming, lo cual simplemente es elegir una sección de la llave, contar cuantos bits 1 hay en esa sección y sumarlos, así el numero resultante es el peso de esa sección, si se realiza este proceso $k \times n$ veces se obtienen los suficientes datos para alimentar los pesos de una TPM.

Para dejar mas claro este concepto, realicemos un ejemplo:

1. Se tiene una cadena de bits N = '0110101110111000010'.
2. Recorremos la cadena posición por posición sumando todos los 1.
3. El resultado de esta cadena seria W(N) = $\sum n_i$ = = 0+1+1+0+1+0+1+1+1+0+1+1+1+0+0+0+0+1+0.
4. El peso de esta cadena es W(N) = 10.

Un punto importante es entender que los pesos deben ser aleatorios, por lo que las llaves propuestas a corregir usando la TPM deben ser aleatorias, tal es el caso de una distribución de llave cuántica (QKD). Por lo que en este caso es sencillo calcular los pesos pues la aleatoriedad de la llave reside en la incidencia de fotones de un criptógrafo cuántico, lo cual ya es los suficientemente aleatorio.

Parte II

Montaje, Métodos y Resultados

Capitulo 5

Métodos y Arreglos Experimentales

5.1 Arreglo Experimental

El montaje experimental que funciona con un láser pulsado que se divide en dos haces de luz, cada uno pulsando a 20 MHz, estos haces inciden en un fotodiodos de avalancha, donde se realiza la cuenta de fotones por instantes de 81 ps. Una vez que se procesan los datos de la cuenta de fotones se genera una secuencias de unos y ceros, el uno indicando cuando hubo incidencia de fotones y el cero indicando cuando no la hubo. Este resultado es la distribución de llave cuántica generada por el criptógrafo. La Figura 5.1 muestra un diagrama del arreglo experimental.

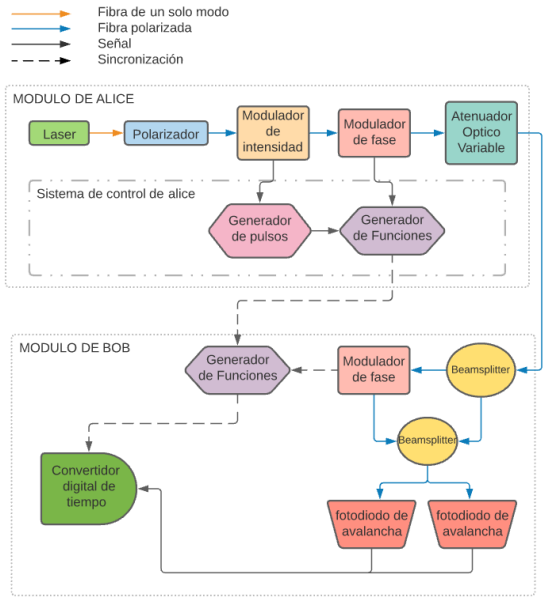

Figura 5.1. Diagrama de montaje del criptógrafo real, los fotodiodos de avalancha son idQuantique: id230-FR-SMF, el convertidor digital de tiempo es idQuantique: id801.

5.2 Códigos de corrección

El programa de corrección y reconciliación consta de 4 módulos principales: `weights, machine, update_rules, run.` Cada uno con un propósito diferente que usa los conocimientos que se vieron en los capítulos anteriores.

El programa esta orientado a sincronizar y cotejar las llaves de Bob y Alice, basándose en la llave que arroja el criptógrafo. El programa consta de 4 apartados, los cuales están destinados a diferentes tareas. De manera general el programa gira en torno a las ecuaciones de la regla Hebbiana (ecuación 4.8), Anti-Hebbiana (ecuación 4.11) y el Random Walker (ecuación 4.12). La función σ (ecuación 4.6) y la función τ (ecuación 4.7), para calcular los valores τ de cada máquina. Una vez que se tienen los valores τ de Alice y Bob se comparan los vectores hasta que sean iguales, se dice entonces que las máquinas están sincronizadas y la llave de ambas es la misma.

5.2.1 Modulo 'weights'

Necesitamos una serie de valores arbitrarios como peso para alimentar a las funciones, aprovecharemos la llave que se obtiene del montaje del criptógrafo cuántico, creando así unos pesos realmente aleatorios pues fueron generados usando principios de mecánica cuántica.

Este proceso de generación de pesos a partir de la llave obtenida del criptógrafo consta de una función que lee el archivo y lo almacena en un vector del tamaño total de dígitos que tenga la llave, después se usa una función que separa una secuencia de tamaño S de estos datos para poder sumar la cantidad de unos que hay en ella, la suma de los unos en esta secuencia es nuestro peso, este algoritmo se llama peso de Hamming o Hamming Weight, Estas secuencias se almacenan en una matriz de k $\times$ n, donde k es la cantidad de neuronas ocultas y n nuestras neuronas de entrada a cada neurona oculta. El diagrama de flujo de este procedimiento se puede ver en la Figura 5.2

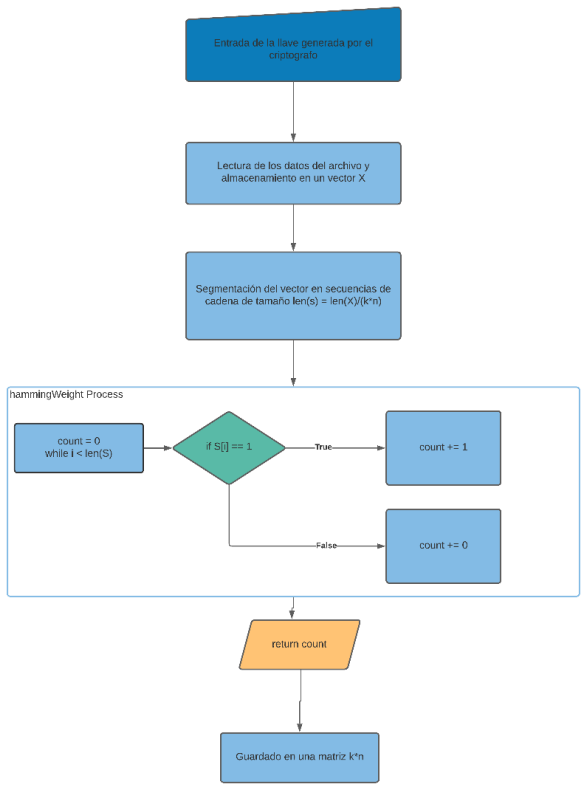

Figura 5.2. Diagrama de flujo del modulo weights donde se calculan los pesos para las neuronas de entrada a partir de la llave obtenida del criptógrafo.

5.2.2 Modulo 'machine'

En este modulo se crean las máquinas de árbol de paridad (TPM), la máquina se define como una clase con los valores principales: k, n, L y un vector W que es el vector de pesos, arbitrariamente se puede usar un vector de valores aleatorias delimitados entre $-L$ y L sin embargo, usar los pesos creados de la llave del criptógrafo hace que obtengamos una distribución de llave mas exacta y exclusiva para cada corrida en el criptógrafo.

En una función llamada '`get_output`' obtenemos todo el proceso de la función signum σ de la ecuación 4.4 para obtener el valor de τ de la ecuación 4.7, con los valores k, n y L que se definen al principio de la clase, al igual que el vector W. Finalmente se introduce un vector aleatorio X que nos ayudara a calcular el valor de τ. El diagrama de flujo de este procedimiento se puede ver en la Figura 5.3

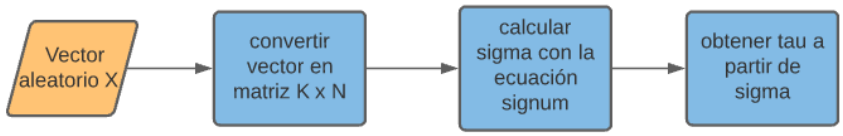

Figura 5.3. Procesos de la función `get_output`.

La siguiente función de esta clase es `update` que actualiza los pesos de acuerdo a la regla escogida, para esta función se necesita el vector aleatorio X, el valor de τ propio de la máquina, el vector de pesos W, y el parámetro L, de manera externa se introduce el valor tau2 que es el valor de la máquina con la que se va a comparar y la regla de aprendizaje que se va a usar. El diagrama de flujo de este procedimiento se puede ver en la Figura 5.4

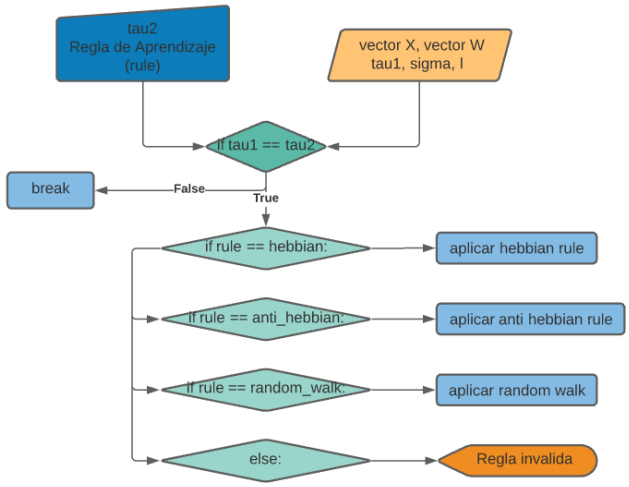

Figura 5.4. Diagrama de la función update.

La última función es `get_key` que da como resultado un archivo de texto dando como resultado una llave que consta de 1 y 0, basada en el vector de pesos W. Lo que hace esta función es convertir un peso a 0 si es menor a 0, y 1 si es mayor o igual a 1. En la Figura 5.5.

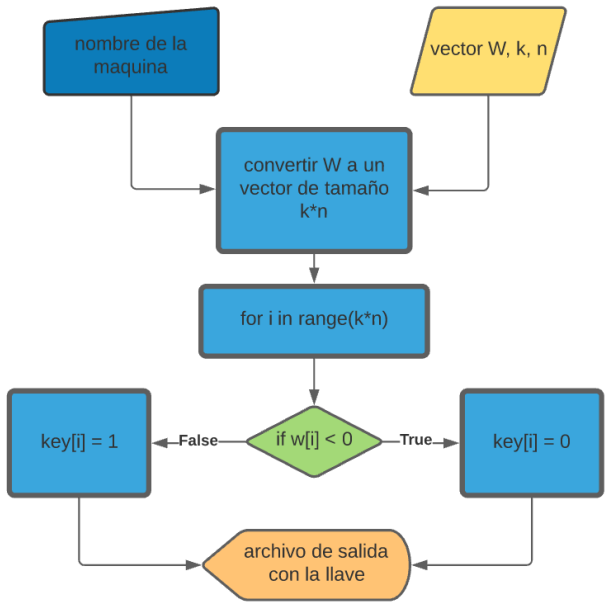

Figura 5.5. Diagrama de flujo de la función get_Key.

5.2.3 Modulo 'update_rules'

En este apartado, se definen las funciones de las reglas de aprendizaje que se mencionan en el capítulo 4, las reglas son: Hebbiana, Anti-Hebbiana y Random Walker. A continuación se muestran los diagramas de flujo de cada uno de estas funciones.

Para simular las reglas de aprendizaje debemos basarnos en las ecuaciones 4.8, 4.11, y 4.12, por lo que necesitaremos una función Θ que nos que nos regresa un valor de 1 cuando nuestros valores de ambas máquinas son iguales, dado que nuestra función Θ se define de la siguiente forma en python:

```
def theta(t1, t2):
        return 1 if t1 == t2 else 0
```

Con esta función definida podemos escribir las demás, en cada función se ingresa un vector W que son nuestros pesos generados a partir de la llave cuántica del criptógrafo, X que es el vector de entradas aleatorias, el resultado de la función signum σ y el valor de τ que se calculan en la función `get_output` del modulo `machine`

El flujo de cada regla es similar, la única diferencia es la forma en la que se operan los valores ingresados, y eso dependerá de la definición de cada regla.

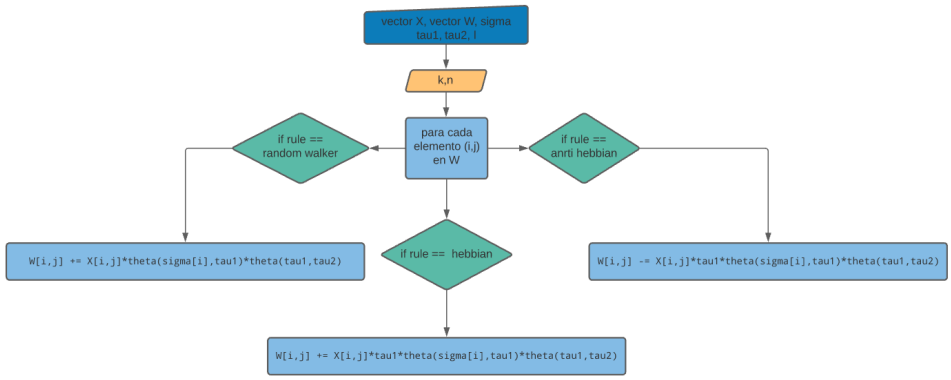

Figura 5.6. Diagrama de las funciones de aprendizaje.

5.2.4 Modulo principal: 'Run'

Todos los módulos se integran en una función principal llamada `run`, esta función toma todos los datos que se ingresan desde el módulo weights hasta el módulo de update_rules, se muestra mejor el proceso en los diagramas de flujo Figura 5.7, Figura 5.8, Figura 5.9.

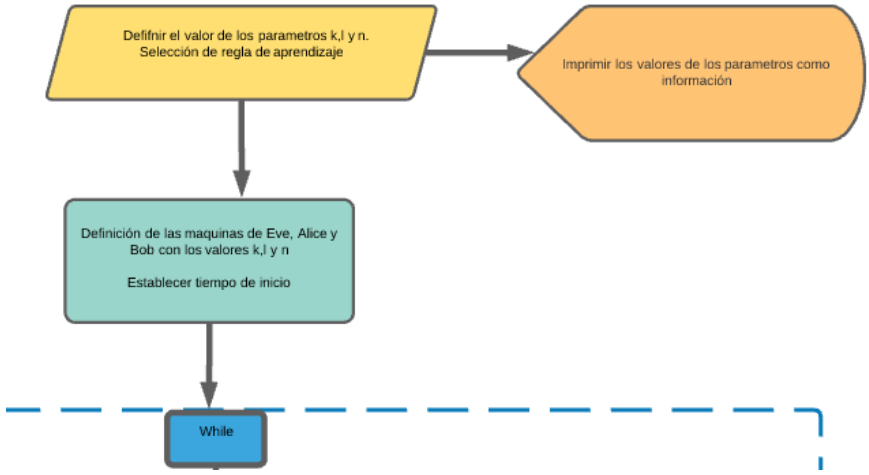

Figura 5.7. Entradas y preparación de la función principal, el diagrama continua en la Figura 5.8

En esta primera parte se preparan todos los datos y valores para comenzar el proceso de verificación de paridad, se imprime la información para tener una noción de que es lo que va a ocurrir. Se definen las máquinas de Eve, Alice y Bob para generar el valor τ que es el que se usa para calcular la sincronización.

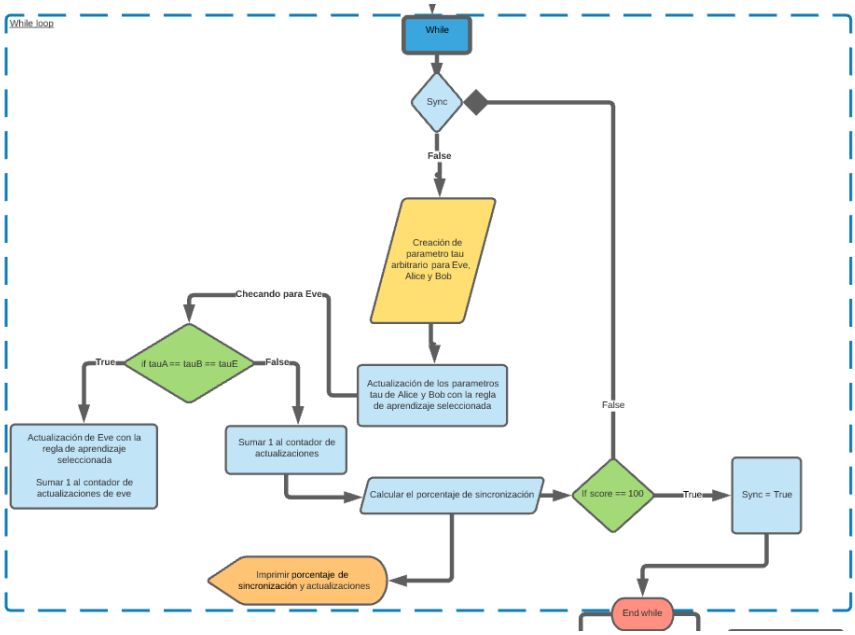

Figura 5.8. Ciclo while que verifica la sincronización de la máquina de Alice con la de Bob, el diagrama continua en la Figura 5.9.

Esta parte es el núcleo de todo el programa, donde se busca la paridad a partir de toda la información que ya se tenia, se comienza creando un valor τ arbitrario para todas las máquinas, se actualizan las máquinas de Alice y Bob cada vez que inicia un ciclo, se suma 1 al contador cada que se actualiza, cuando el valor τ de Eve es igual al de Alice y Bob se actualiza la máquina de Eve y se suma 1 al contador de Eve.

Al final de cada corrida se calcula el porcentaje sincronización con:

```
score = 100 * sync_score(Alice, Bob).
```

Una vez que el porcentaje de sincronización llegue al 100 % la sincronización estará completa y el ciclo termina.

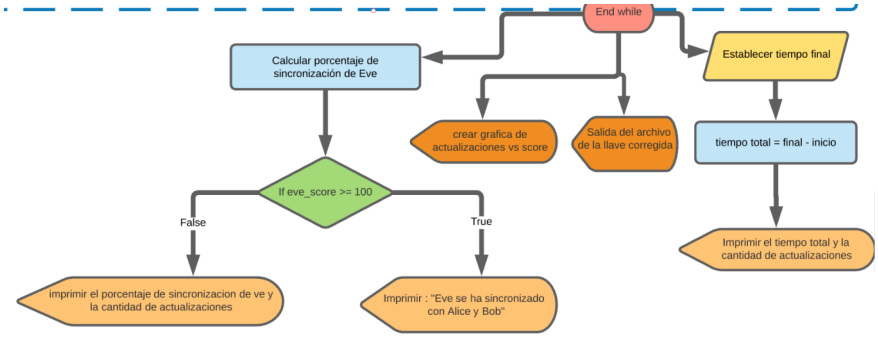

Figura 5.9. Salidas de la función principal.

Una vez que el ciclo ha terminado, se calcula el porcentaje de sincronización de Eve, se imprime una gráfica de la cantidad de actualizaciones de Alice y Bob contra el porcentaje de sincronización en cada una de las actualizaciones. También de manera informativa se imprime cuanto tiempo hubo desde que inicio hasta que finalizo el proceso, y finalmente obtenemos un archivo de texto con la llave generada por Alice y Bob que deben ser idénticas.

Capitulo 6

Resultados

En esta sección mostraremos los resultados obtenidos después de realizar las pruebas correspondientes, las pruebas se realizaron en su mayoría con simulación de generador de llave cuántica, que consta de un generador de números pseudo aleatorios con una probabilidad $\frac{1}{2}$ para obtener un 1 o un 0.

6.1 Preparación para las pruebas

Para cada regla se realizó una iteración de 50 pruebas, con los parámetros de $k = 100$, $n = 10$, $L = 10$, para llaves de 1 millón de bits para las pruebas. Para tener una idea mas clara, 1 MB es lo que suele pesar una imagen de alta resolución en PNG o JPEG, un PDF de mas de 200 páginas, por lo que probar el código con vectores que pesan 1 MB hará que el tiempo de prueba sea menor.

6.2 Procesamiento para obtención de la llave

Una vez que se decide el tamaño de la llave, podemos proceder a realizar todo el proceso descrito en el capitulo anterior, iniciamos con el modulo de 'Weights' que se explican en forma de diagrama de flujo en la Figura 5.2, donde ahí generamos nuestro vector de prueba y también existe la función para insertar un archivo .txt como la llave obtenida del criptógrafo, una vez que obtenemos nuestra matriz W de pesos continuamos al modulo 'Machine' que prepara los valores de σ y τ en la función `get_output` (Figura 5.3), estos valores se usan en el modulo 'Update_Rules' (Figura 5.6), que se aplica dentro del modulo 'Machine' en la función `update` (Figura 5.4), y finalmente cuando en el ciclo `while` del modulo principal 'Run' se obtiene un porcentaje de sincronización del 100 %, la función `get_key` (Figura 5.5) del modulo 'Machine', que nos imprime un archivo .txt con el

nombre de la maquina, la regla de aprendizaje, y la fecha y hora de creación.

Como ejemplo obtenemos:

```
QKDAlice2022_01_18-10:39:44_pm.txt
QKDBob2022_01_18-10:39:44_pm.txt
```

Que son la misma llave, entonces Alice y Bob ya pueden entablar comunicación a través de un canal seguro.

6.3 Comparación de reglas de aprendizaje

Para la comparación de las reglas, se realizan 50 pruebas de cada uno, por lo que se itera el programa una y otra vez por 50 veces y al final tenemos como salida una gráfica para cada regla con el historial de actualizaciones y tres archivos que registran el tiempo y la cantidad de actualizaciones para la regla Hebbiana Figura 6.1, para la regla Anti-Hebbiana Figura 6.3 y para la regla Random Walker Figura 6.2.

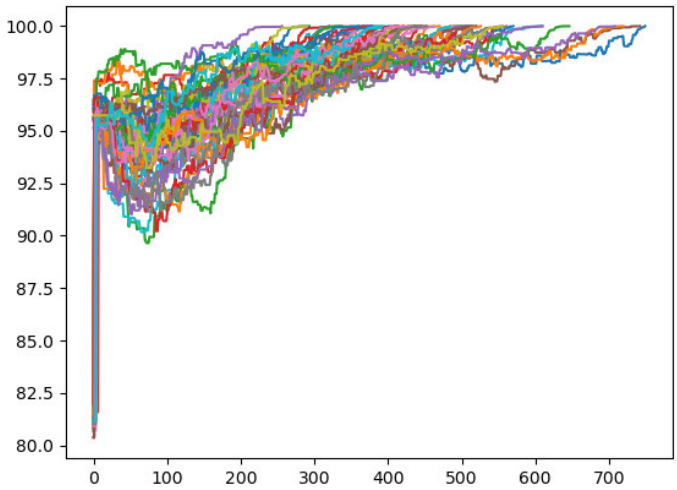

Figura 6.1. Historial de las actualizaciones de las 50 pruebas con la regla Hebbiana.

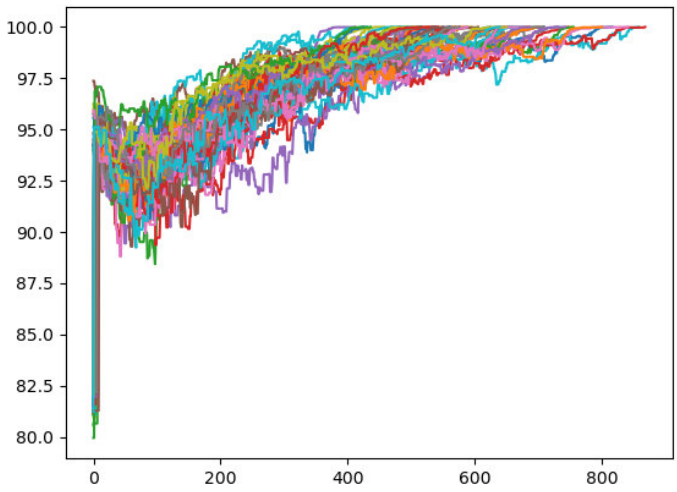

Figura 6.2. Historial de las actualizaciones de las 50 pruebas con la regla Anti-Hebbiana.

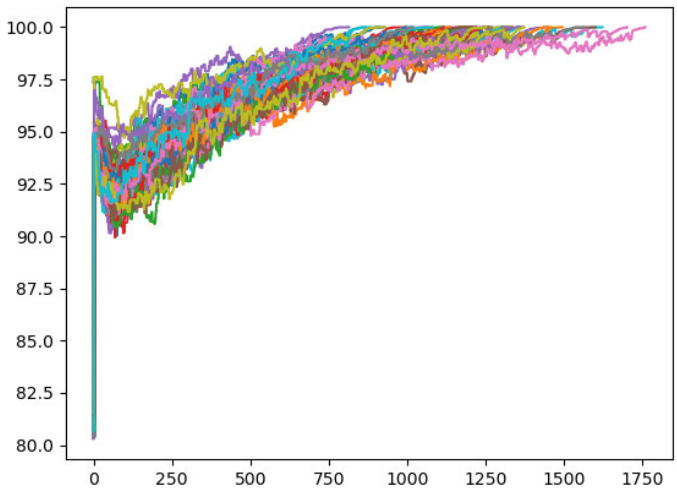

Figura 6.3. Historial de las actualizaciones de las 50 pruebas con la regla Random Walker.

A primera instancia se puede notar que la regla Hebbiana tiene menos cantidad de actualizaciones antes de llegar al 100 % de porcentaje de sincronización, para poder comprobar esto se gráfica el tiempo

transcurrido de todas las pruebas de cada regla, como se muestra en la Figura 6.4.

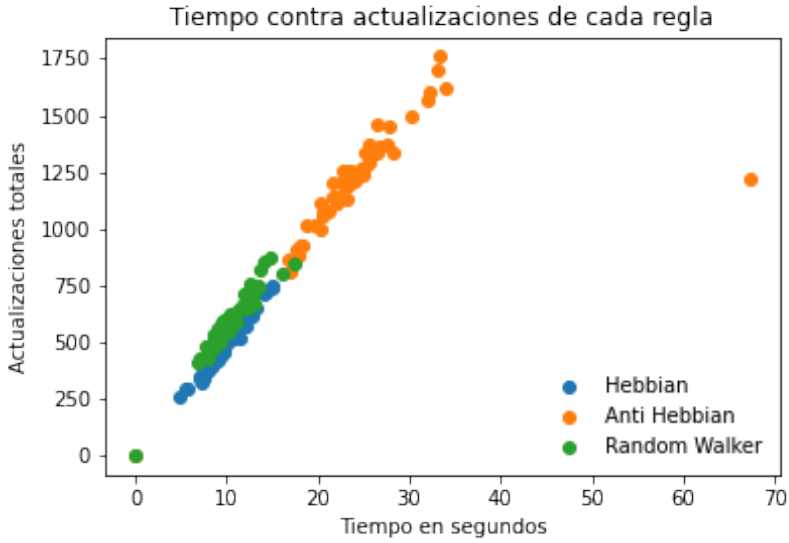

Figura 6.4. Correlación del tiempo tomado contra las actualizaciones de cada regla.

Al promediar y obtener la desviación estándar se obtiene la Tabla 6.1:

Regla	Tiempo promedio (s)	Desviación estándar Tiempo (s)	Actualizaciones promedio	Desviación estándar Actualizaciones
Hebbian	9.43	2.30	467	115
Anti Hebbian	24.80	7.51	1230	213
Random Walker	12.42	1.66	728	125

Tabla 6.1. Comparación de tiempos y actualizaciones de las diferentes reglas de aprendizaje.

Donde se comienza a apreciar que efectivamente la regla Hebbiana es la mas rápida para el aprendizaje y el proceso de corrección. Para una mejor visualización se realizan gráficas de caja para visualizar el promedio y la desviación estándar de los tiempos en la Figura 6.5 y de actualizaciones en la Figura 6.6.

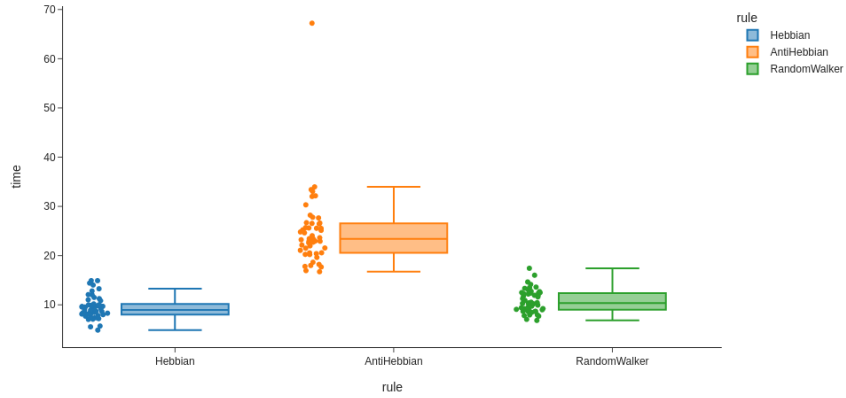

Figura 6.5. Gráfica estadística de promedio y desviación estándar del tiempo tomado en las simulaciones de las tres clases de reglas.

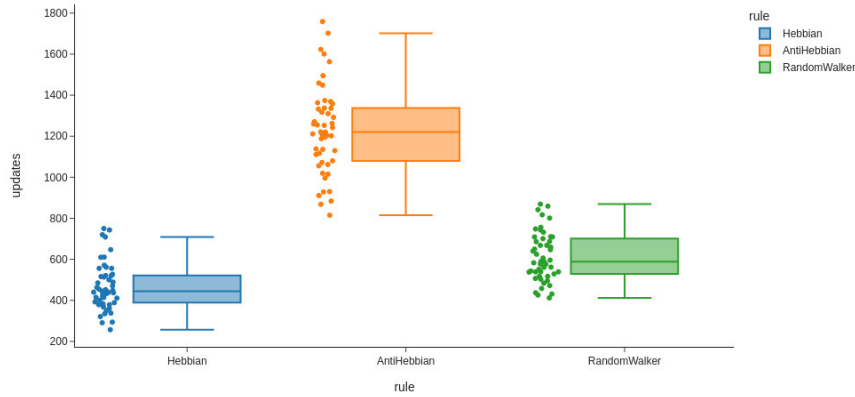

Figura 6.6. Gráfica estadística de promedio y desviación estándar de las actualizaciones totales hechas en las simulaciones de las tres clases de reglas.

Finalmente, se puede concluir que la regla de aprendizaje Hebbiana es la más adecuada y eficaz para corregir los errores en la llave cuántica del criptógrafo y generar una llave que coincida para ambas partes.

6.4 La llave Cuántica

En esta sección se mostrara la llave de salida que genera el programa y se comprobara que sean iguales, este análisis se hizo en una libreta

de Jupyter. Primero se carga una llave para Alice y otra para Bob que fueron generadas con el programa, la llave sera convertida a un vector y a cadena para verificar ambos objetos como se muestra en la Figura 6.7.

```
Alice = pd.read_csv('/home/jonathan/Documents/QKD/TPM_QKD/keys/hebbian/QKDAlice2022_01_18-11:01:28_pm.txt')
A = Alice.to_numpy()
A = A.T
A_string = np.array2string(A)
##########
Bob = pd.read_csv('/home/jonathan/Documents/QKD/TPM_QKD/keys/hebbian/QKDBob2022_01_18-11:01:28_pm.txt')
B = Bob.to_numpy()
B = B.T
B_string = np.array2string(B)
```

Figura 6.7. Preparación de las llaves para convertirse en objetos mejor manipulables.

Se verifica la cadena de Alice que se ha nombrado `A_string` y la cadena de Bob que se ha nombrado `B_string` como se muestra en la Figura 6.8.

```
A_string == B_string

True
```

Figura 6.8. Verificación de la cadena de la llave de Alice y Bob, resultando en TRUE pues ambas son idénticas.

Las cadenas resultan ser iguales, por lo que la llave de Alice es idéntica a la de Bob. A continuación se muestran ambas llaves una contra otra en la Figura 6.9 y la Figura 6.10.

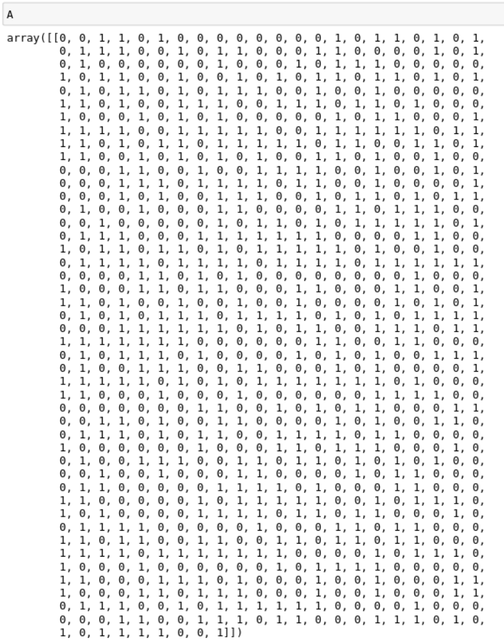

Figura 6.9. Arreglo de la llave de Alice.

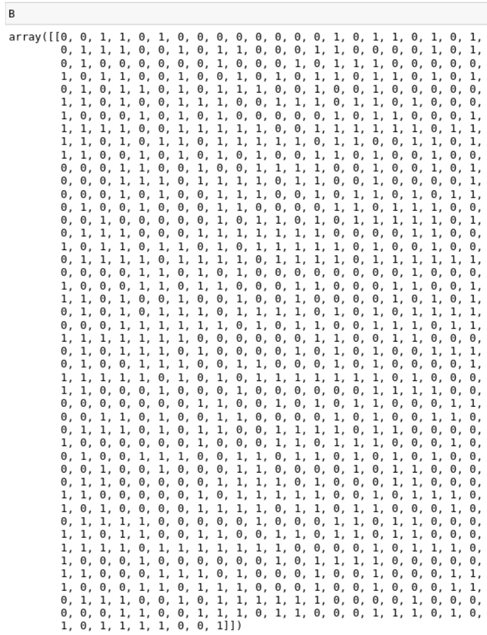

Figura 6.10. Arreglo de la llave de Bob.

Para verificar que las dos cadenas sean iguales a primera vista, se decodifican la llave binaria a código ASCII con el alfabeto latino, lo que resultara en una cadena de caracteres que la maquina podrá leer como binario y los humanos como texto plano, el resultado se muestra en la Figura 6.11.

```
print(bytearray(np.packbits(A)).decode("latin-1").strip("\x00"))

Â_\¶¾ï{ðÔ#cÒHU]êñúÎÿ]{!å#
9«ú1<Ó
gYì FâNmPF0_.¡íà¶=ü.8ÜHÜØêò

print(bytearray(np.packbits(B)).decode("latin-1").strip("\x00"))

Â_\¶¾ï{ðÔ#cÒHU]êñúÎÿ]{!å#
9«ú1<Ó
gYì FâNmPF0_.¡íà¶=ü.8ÜHÜØêò
```

Figura 6.11. Decodificación de binario a código ASCII para interpretar la llave como texto plano.

Así entonces se obtiene la llave depurada entre Alice y Bob. Aplicando a una llave cuántica mas grande implicara mas tiempo de computo, por lo que seccionar la llave en llaves mas cortas puede ser una solución a este problema.

Capitulo 7

Conclusión y Perspectiva

El resultado de este trabajo ha sido satisfactorio, pues se logro crear una reconciliación de las distribuciones de las llaves con corrección de errores, las maquinas de árbol de paridad (TPM) muestran ser una herramienta adecuada para la reconciliación pues realizan un trabajo eficiente. Además de agregar una capa de seguridad a la llave de encriptación pues el proceso de redes neuronales suele asemejarse al de una caja negra, se sabe como funciona; sin embargo, no se sabe como funciona en tiempo real, como dato notable el proceso de las TPM es Turing completo, pues dentro de una maquina de paridad podemos recrear otra maquina de paridad. Se concluye que de las tres reglas de aprendizaje, la mas eficiente es la regla Hebbiana.

Las TPM son bastante sencillas de aplicar pero sus fundamentos son complejos, tomar reglas de aprendizaje basado en el comportamiento de las neuronas de un cerebro. El hecho de preparar la entrada para que tenga mejor rendimiento indica que se puede optimizar el proceso si se considera la entropía de Shannon para reducir el tamaño de las llaves cuánticas que se van a procesar. Se sugiere encontrar la manera de realizar este proceso en tiempo real con el criptógrafo y así tener un mejor rendimiento y eficiencia del proceso de cifrado y corrección

Este proceso de reconciliación y corrección de errores, en conjunto con la generación de llaves cuánticas a partir del criptógrafo, es una integración bastante potente, pues combinar tecnologías de estado del arte genera resultados que propician una suma de conocimiento a los campos de ciencia y tecnología correspondientes.

Capitulo 8

Bibliografía

[1] S. Wiesner, "Conjugate coding," SIGACT News, vol. 15, p. 78–88, jan 1983. Citado en pág/(s) 4.

[2] C. H. Bennett, G. Brassard, and J.-M. R., "Privacy amplification by public discussion," SIAM J. Comput., vol. 17, pp. 210–229, 1988. Citado en pág/(s) 5.

[3] N. P. Smart, Cryptography Made Simple. Springer, 2016. Citado en pág/(s) 7.

[4] M. Kubat, An Introduction To Machine Learning. Springer, 2017. Citado en pág/(s) 10.

[5] P. R. Berman, Introductory Quantum Mechanics. Springer, 2018. Citado en pág/(s) 10.

[6] R. Shankar, Principles of Quantum Mechanics. Springer, 1994. Citado en pág/(s) 12, 14.

[7] T. Norsen, Foundations Of Quantum Mechanics. Springer, 2017. Citado en pág/(s) 12, 14.

[8] P. L. K. C. C. Gerry, Introductory Quantum Optics. Cambridge, 2005. Citado en pág/(s) 15.

[9] F. Grosshans, G. Van Assche, J. Wenger, R. Brouri, N. Cerf, and P. Grangier, "Quantum key distribution using gaussian-modulated coherent states," Nature, vol. 421, p. 238–241, Jan 2003. Citado en pág/(s) 17.

[10] "Quantum cryptography: Public key distribution and coin tossing," vol. 560. Citado en pág/(s) 17.

[11] C. F. Fung, K. T., and H.-K. L., "Performance of two quantum-key-distribution protocols," Physical Review A, vol. 73, Jan 2006. Citado en pág/(s) 19.

[12] G. Brassard and L. Salvail, "Secret-key reconciliation by public discussion," in Advances in Cryptology — EUROCRYPT '93 (T. Helleseth, ed.), (Berlin, Heidelberg), pp. 410–423, Springer Berlin Heidelberg, 1994. Citado en pág/(s) 21.

[13] C. Bennett, G. Brassard, C. Crepeau, and U. Maurer, "Generalized privacy amplification," IEEE Transactions on Information Theory, vol. 41, no. 6, pp. 1915–1923, 1995. Citado en pág/(s) 21.

[14] W. T. Buttler, S. K. Lamoreaux, J. R. Torgerson, G. H. Nickel, C. H. Donahue, and C. G. Peterson, "Fast, efficient error reconciliation for quantum cryptography," Physical Review A, vol. 67, May 2003. Citado en pág/(s) 22.

[15] T. Richardson and R. U., "Multi-edge type ldpc codes," ISIT talk, 01 2002. Citado en pág/(s) 22.

[16] B. Perez-Garcia, J. F., M. McLaren, R. I. Hernandez-Aranda, A. Forbes, and T. Konrad, "Quantum computation with classical light: The deutsch algorithm," Physics Letters A, vol. 379, p. 1675–1680, Aug 2015. Citado en pág/(s) 25.

[17] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM Journal on Computing, vol. 26, p. 1484–1509, Oct 1997. Citado en pág/(s) 26.

[18] L. A. Zadeh, "Fuzzy logic, neural networks, and soft computing," Communications of the ACM, vol. 37, pp. 77–84, Mar. 1994. Citado en pág/(s) 29.

[19] World Conference on Soft Computing. http://softcomputing.org/. Citado en pág/(s) 29.

[20] I. J. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016. http://www.deeplearningbook.org. Citado en pág/(s) 29.

[21] M. Volkmer and S. Wallner, "Tree parity machine rekeying architectures," IEEE Transactions on Computers, vol. 54, no. 4, pp. 421–427, 2005. Citado en pág/(s) 31.

[22] M. Niemiec, "Error correction in quantum cryptography based on artificial neural networks," CoRR, vol. abs/1810.00957, 2018. Citado en pág/(s) 31.

[23] W. Wang and H. Lo, "Machine learning for optimal parameter prediction in quantum key distribution," Physical Review A, vol. 100, Dec 2019. Citado en pág/(s) 31.

[24] Lizama-Pérez, J. M. López R., and E. H. Samperio, "Beyond the limits of shannon's information in quantum key distribution," Entropy, vol. 23, no. 2, 2021. Citado en pág/(s) 31.

[25] A. Goltsev and V. Gritsenko, "Modular neural networks with hebbian learning rule," Neurocomputing, vol. 72, no. 10, pp. 2477–2482, 2009. Lattice Computing and Natural Computing (JCIS 2007) / Neural Networks in Intelligent Systems Designn (ISDA 2007). Citado en pág/(s) 33.

UNIVERSIDAD DE GUANAJUATO

León, GTO, 22 de febrero de 2022

Dr. Delepine, David Yves Ghislain

Director de la División de Ciencias e Ingenierías

Universidad de Guanajuato campus León

Por este medio, le comunico que me fue entregada para revisión en calidad de sinodal, la tesis con título ***Corrección de Errores con Machine Learning en un Sistema de Distribución de Claves Cuánticas* (QKD)**, del estudiante de licenciatura en Ingeniería Física, **Jonathan Alejandro González Pérez**.

Cabe señalar que el **C. González Pérez** mostró secuencia lógica en su escrito y amplio dominio del tema, atendió además mis sugerencias, por lo que sugiero se le permita llevar a cabo su defensa para la obtención del título de Ingeniero en Físico.

Permítame enviar un cordial saludo y agradece su fina atención a la presente

ATENTAMENTE
"LA VERDAD OS HARÁ LIBRES"

Dr. Teodoro CORDOVA FRAGA
Profesor Titular B
SNI nivel II

División de Ciencias e Ingenierías, Campus León
Loma del Bosque 103, Lomas del Campestre, 37150 León, GTO, Mx
Email: tcordova@ugto.mx, Tel 477 2723151 ex. 8454

UNIVERSIDAD DE GUANAJUATO

León, Gto, a 22 de febrero de 2022

Dr. David Yves Ghislain Delepine
Director de la División de Ciencias e Ingenierías
Campus León, Universidad de Guanajuato

Estimado Dr. Delepine:

Por medio de la presente informo a usted que he leído el manuscrito de tesis del estudiante de Ingeniería Física de la DCI, C. Jonathan Alejandro González Pérez, la cual lleva por título "**Corrección de Errores con Machine Learning en un sistema de Distribución de Claves Cuánticas (QKD)**".

Después de haber realizado mis comentarios, el estudiante realizó las correcciones pertinentes. Por lo anterior doy mi consentimiento para que el **C. Jonathan Alejandro González Pérez** defienda de manera oral su tesis de licenciatura en la fecha que sus directores de tesis juzguen conveniente.

Sin más por el momento, y agradeciendo la atención a la presente, me despido enviando un cordial saludo

Atentamente

Dr. Luis Carlos Padierna García
Profesor-Investigador

DEPARTAMENTO DE INGENIERÍAS QUÍMICA, ELECTRÓNICA Y BIOMÉDICA
DIVISIÓN DE CIENCIAS E INGENIERÍAS
Loma del Bosque #103, Col. Lomas del Campestre. León, Gto. C.P. 37150
Tel. +52 (477) 7 88 51 00 ext. 8522
lc.padierna@ugto.mx
www.dci.ugto.mx/depingquim

UNIVERSIDAD DE GUANAJUATO

León, Gto., a 21 de febrero de 2022
Asunto: **Revisión de Tesis**

DR. DAVID YVES GHISLAIN DELEPINE
DIRECTOR
DIVISIÓN DE CIENCIAS E INGENIERIAS
CL -UNIVERSIDAD DE GUANAJUATO

A través de la presente constato que he revisado la tesis del C. **Jonathan Alejandro Gozález Pérez** con el fin de obtener el grado de Licenciatura en Ingeniería Física. El trabajo de tesis se titula **"Corrección de errores con Machine Learning en un sistema de Distribución de Claves Cuánticas (QKD)"**. En este estudio realizado por Jonathan desarrolla tres algoritmos de corrección de erres cuánticos basados en máquinas de soft computing para usarse como la capa de corrección de errores en un sistema de distribución de claves cuánticas. Evalúa los algoritmos en una clave simulada usando el protocolo SARG04, obtiene la comparación de tiempos y eficiencia de los algoritmos. El trabajo de titulación satisface con la completez y solidez de un proyecto de titulación a nivel licenciatura. Jonathan ha realizado las correcciones pertinentes al documento de la tesis. Además, he cuestionado a Jonathan en los temas relacionados a su trabajo de tesis, demostrando su amplio dominio en los temas abordados en su trabajo de tesis. Por lo que considero que ya se puede proceder con la disertación de tesis.

Sin más por el momento le envío saludos cordiales.

Atentamente

Dr. Carlos Herman Wiechers Medina
Profesor-Investigador

Tel. +52 (477) 7885100 Ext. 3867
Cel. +52 (477) 1080605
e-mail 1: carherwm@fisica.ugto.mx
e-mail 2: ch.wiechers@ugto.mx

UNIVERSIDAD DE GUANAJUATO

León, Gto., 22 de febrero del 2022

DR. DAVID DELEPINE
DIVISIÓN DE CIENCIAS E INGENIERÍAS, CAMPUS LEÓN
DIRECTOR

P R E S E N T E

Por medio de la presente, como miembro del jurado calificador designado para revisar el trabajo de grado con título ***"Corrección de Errores con Machine Learning en un Sistema de Distribución de Claves Cuánticas (QKD)"*** que sustenta el ***C. Jonathan Alejandro González Pérez*** con el fin de obtener el grado de ***Licenciado en Ingeniería Física***, hago constar que he leído el trabajo y que avalo el contenido y calidad del mismo como un trabajo de Tesis de Licenciatura.

Sin más por el momento le envío saludos cordiales, quedando de usted para cualquier aclaración.

Atentamente,

Lorena Velázquez I.

Dra. Lorena Velázquez Ibarra

Profesora Asociada C