Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias.
Ingeniería de Sistemas y Computación.
Trabajo de Grado.

La ciberseguridad de un canal de comunicación entre computadores cuánticos para la distribución de claves cuánticas simétricas.

Dilan Andres Correa Ortíz
Santiago Torres Rincón

Directora: MSc. Liliana Gutierrez Rancruel

21 de julio de 2024

Santiago de Cali, 21 de julio de 2024.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo Mauricio Sarria Montemiranda
Director Carrera de Ingeniería de Sistemas y Computación.
Cali.

Cordial Saludo.

Por medio de la presente me permito informarle que los estudiantes de Ingeniería de Sistemas y Computación Dilan Andres Correa Ortíz (cod: 8950069) y Santiago Torres Rincón (cod: 8940234) trabajan bajo mi dirección en el proyecto de grado titulado "La ciberseguridad de un canal de comunicación entre computadores cuánticos para la distribución de claves cuánticas simétricas.".

Atentamente,

MSc. Liliana Gutierrez Rancruel

Santiago de Cali, 21 de julio de 2024.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo Mauricio Sarria Montemiranda
Director Carrera de Ingeniería de Sistemas y Computación.
Cali.

Cordial Saludo.

Nos permitimos presentar a su consideración el trabajo de grado titulado "La ciberseguridad de un canal de comunicación entre computadores cuánticos para la distribución de claves cuánticas simétricas." con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el proyecto de grado y posteriormente optar al título de Ingeniero de Sistemas y Computación.

Al firmar aquí, damos fe que entendemos y conocemos las directrices para la presentación de trabajos de grado de la Facultad de Ingeniería aprobadas el 26 de Noviembre de 2009, donde se establecen los plazos y normas para el desarrollo del anteproyecto y del trabajo de grado.

Atentamente,

Dilan Andres Correa Ortíz
Código: 8950069

Santiago Torres Rincón
Código: 8940234

Resumen

La transmisión de información es vital en la actualidad donde tenemos un mundo que se encuentra cada vez más conectado a través de medios digitales. Se han realizado muchos métodos en pro de ejercer la ciberseguridad y brindar canales de comunicación seguros. Sin embargo, con el pasar del tiempo avanzamos a tecnologías cada vez más robustas y complejas que nos permiten alcanzar nuevos horizontes de conocimiento que están más cerca de nuestras manos. Un hecho del que poco impacto ha tenido para el público general, es a lo que se tendrá que enfrentar la ciberseguridad en unos cuantos años con la aparición de la computación cuántica, pues esta traerá consigo tiempos de cálculos computacionales abruptamente inferiores a los que puede ofrecer la computación tradicional. A causa de esto, el proyecto que se propone, como aporte a fomentar la investigación y la divulgación de información del problema descrito, es realizar en un lenguaje de programación la implementación del protocolo BB84 usando máquinas tradicionales donde se busca diseñar un algoritmo que simule el envío y la recepción de fotones por el canal cuántico creando la estructura básica de los componentes de un computador cuántico para posteriormente realizar pruebas experimentales que permitan evaluar la eficiencia y seguridad del protocolo considerando las limitaciones de los componentes creados. Para finalizar se presentarán las conclusiones encontradas con recomendaciones personales para trabajos futuros.
Palabras Clave: Ciberseguridad, canal de comunicación, computación cuántica, distribución de claves cuánticas, protocolo BB84.

Agradecimientos

Dilan Correa

Al concluir una etapa tan importante en mi vida, me gustaría dar agradecimiento, aunque quizá no lleguen a leer esto, a mis abuelos paternos, a mi mamá y a mi hermano, quienes han sido pilares en mi vida para poder llegar hasta este momento que veía tan lejos en un momento, pero que finalmente está más cerca de lo que me gustaría.

Por último, quiero recordar y agradecer especialmente a mi abuelo quien fue la persona que veló por mi desde mi infancia y que gracias a él soy la persona que soy y estoy a punto de conseguir este logro que seguramente estaría muy orgulloso de ver. Aunque no estuvo en toda mi etapa universitaria pues falleció meses antes de que me graduara como bachiller, quiero plasmar unas palabras que escribí hace un tiempo para despedirlo:
Te extrañaré y lloraré el resto de mis días. No le voy a contar a mis hijos o nietos sobre ti, porque no quiero tener, pero prometo que a todo ser merecedor le contaré sobre mi héroe, que sin ninguna capa salvó mi vida, y lo sigue haciendo a pesar de su ausencia.

Te amo mucho, viejo.

Santiago Torres

Me gustaría agradecer primero a mi compañero de tesis y mejor amigo por el excelente trabajo que hemos completado, por todas las horas que le dedicó a entender completamente los temas necesarios para sacar este trabajo adelante junto con las incontables horas dedicadas asegurándose de que todo quedara impecable. Agradezco mucho a mis padres y hermano por todo el apoyo que me brindaron en los momentos difíciles del desarrollo de este trabajo de grados, desde su concepción hasta su conclusión. A Beto, Majo, Nicolle, Guido y todos mis amigos por darnos motivación, darnos consejos de la forma en la que teníamos que desarrollar este trabajo y siempre creer en nosotros. A nuestra directora de tesis por tenernos paciencia y guiarnos de la mejor manera.

Por último me gustaría agradecer a todas las personas que ya no están con nosotros para disfrutar de estos momentos importantes y especiales, ya que sin su amor y enseñanzas no sería la persona que soy hoy. Los amo y extraño mucho.

Índice general

1. Descripción del Problema **11**

1.1. Planteamiento del Problema . . . 11

1.1.1. Formulación . . . 12

1.1.2. Sistematización . . . 12

1.2. Objetivos . . . 13

1.2.1. Objetivo General . . . 13

1.2.2. Objetivos Específicos . . . 13

1.3. Justificación . . . 13

1.4. Delimitaciones y Alcances . . . 14

1.4.1. Entregables . . . 14

2. Desarrollo del Proyecto **15**

2.1. Marco de Referencia . . . 15

2.1.1. Áreas Temáticas . . . 15

2.1.2. Marco Teórico . . . 16

2.1.3. Trabajos Relacionados . . . 19

2.2. Metodología . . . 20

2.2.1. Tipo de Estudio . . . 20

2.2.2. Actividades . . . 21

2.3. Resultados Esperados . . . 22

2.4. Cronograma . . . 22

2.5. Recursos . . . 22

2.5.1. Humanos . . . 22

2.5.2. Técnicos . . . 23

2.5.3. Presupuesto . . . 23

3. Investigación **25**

3.1. Conceptos adicionales relevantes . . . 25

3.1.1. Canales clásicos y cuánticos . . . 25

3.1.2. Mecanismos de verificación . . . 26

3.1.3. Métodos de reconciliación de información . . . 27

3.1.4. Privacidad amplificada [41] . . . 30

3.1.5. Propagación de creencias (Belief propagation) en códigos LDPC [42] . . . 30

4. Implementación y Resultados **32**
4.1. Metodología de desarrollo . . . 32
4.1.1. Diagrama de caso de uso . . . 34
4.1.2. Historias de Usuario . . . 35
4.1.3. Diagrama de clases . . . 37
4.1.4. Diagrama de flujo . . . 38
4.2. Implementación . . . 39
4.2.1. Librerías . . . 39
4.2.2. Implementación de la generación de bits pseudoaleatorios . . . 40
4.2.3. Cúbits, Preparación y Conciliación de Bases . . . 40
4.2.4. Creación de Bits Aleatorios . . . 41
4.2.5. Simulación envío y recepción de fotones polarizados . . . 41
4.2.6. Algoritmo LDPC . . . 41
4.3. Evaluación de Resultados . . . 45
4.3.1. Prueba 1 - Ejecución sin ruido y/o un espía . . . 45
4.3.2. Prueba 2 - Ejecución con ruido y/o un espía . . . 47
4.4. Validación de seguridad de la implementación del protocolo . . . 49
4.4.1. Parámetros evaluados . . . 49
4.4.2. Validación según ETSI GS QKD 005 V1.1.1 (2010-12) . . . 50
4.4.3. Validación según ETSI GS QKD 008 V1.1.1 (2010-12) . . . 50

5. Conclusiones y Trabajos Futuros **52**
5.1. Conclusiones . . . 52
5.2. Trabajos futuros . . . 52

6. Anexo A: Código del Protocolo BB84 **54**

Bibliografía **69**

Introducción

La comunicación es una de las actividades más importantes y fundamentales para el ser humano. Desde tiempos antiguos, el ser humano ha buscado diferentes formas de transmitir información, desde señales de humo, tambores, y telégrafos hasta el internet que usamos hoy en día. Sin embargo, para todas estas existe un problema común: la seguridad de la transmisión. En el pasado resultaba un poco más fácil garantizar la privacidad de la información que se enviaba, ya que los métodos que se utilizaban no eran tan susceptibles a ser interceptados por espías. Esto plantea algunas preguntas: ¿Cómo se puede garantizar que un espía no intercepte, modifique o copie los mensajes que se están enviando sin que nos demos cuenta? ¿Cómo se puede proteger nuestra privacidad en la transmisión de información en un mundo que está cada vez más conectado y amenazado por espías? [24] Para abordar estas preguntas, citamos a IBM: "La ciberseguridad es la práctica de proteger los sistemas más importantes y la información confidencial ante ataques" [2]. La ciberseguridad, entonces, juega un papel fundamental para la sociedad actual.

Actualmente, existen proyectos de ley que buscan sancionar a aquellas empresas que no implementen protocolos de ciberseguridad para la protección de información y su envío [25] [26] [27]. Algunos protocolos utilizados son los de criptografía simétrica o asimétrica, cuya función es establecer claves públicas y/o privadas que permiten una comunicación segura. Por ejemplo, el RSA [1], el primer algoritmo disponible al público y uno de los más utilizados para establecer claves públicas y privadas, es fundamental en la ciberseguridad para garantizar la seguridad en el envío y recepción de mensajes.

Sin embargo, mirando hacia el futuro, encontramos un gran problema: la computación cuántica, la cual podría comprometer la ciberseguridad debido a la capacidad de los computadores cuánticos para realizar cálculos gigantescos; incluso protocolos como el RSA podrían volverse obsoletos. Entonces, ¿cómo podríamos garantizar que la ciberseguridad siga cumpliendo su función? La respuesta podría encontrarse en la mecánica cuántica, que estudia el comportamiento de las partículas subatómicas. La mecánica cuántica nos muestra un mundo a escala microscópica muy diferente del que percibimos en escala macroscópica. Las partículas pueden existir en una superposición de estados, es decir, en una combinación de dos o más posibilidades hasta que se realiza una medición. Además, las partículas cuánticas pueden estar entrelazadas, compartiendo propiedades que se mantienen incluso cuando están separadas por grandes distancias.

Uno de los primeros y más famosos protocolos de comunicación cuántica es el BB84, propuesto por Charles Bennet y Gilles Brassard en 1984. Este protocolo permite que dos partes intercambien una clave privada usando un canal cuántico y un canal clásico. La clave secreta puede luego usarse para cifrar y descifrar mensajes usando el algoritmo de Vernam [16], que garantiza una buena seguridad dada su conformación. El protocolo BB84 se basa en el principio de incertidumbre de Heisenberg, que establece que no se puede medir con precisión dos propiedades complementarias de

una partícula cuántica, como su posición y su momento, o su polarización horizontal y vertical. Esto significa que si un espía intenta interceptar y medir los fotones que el emisor envía al receptor por medio del canal cuántico, alterará su estado y podrá ser detectado por estas dos partes, indicando que la comunicación no es segura en ese momento.

En este proyecto, se busca desarrollar la implementación del protocolo BB84 en un lenguaje de programación usando máquinas tradicionales. Aunque existen algunas propuestas y soluciones similares [17] [18] [19], se pretende diseñar un algoritmo que simule el envío y la recepción de fotones por el canal cuántico, creando la estructura básica de los componentes de un computador cuántico dentro de un lenguaje de programación. Además, se realizarán pruebas experimentales del algoritmo desarrollado para evaluar la eficacia y la seguridad del protocolo dentro de las limitaciones realizadas por los componentes creados para la simulación. Los resultados obtenidos se compararán con los esperados teóricamente, y se discutirán las conclusiones, además de formular recomendaciones para trabajos futuros.

Capítulo 1

Descripción del Problema

1.1. Planteamiento del Problema

Dentro del área de la ciberseguridad existe una rama que se encarga de la protección digital de los datos. La encriptación es la encargada de proteger los datos usando distintas técnicas y algoritmos con diversos propósitos. Una de estas técnicas es el algoritmo RSA (Rivest-Shamir-Adleman) [1], el cual usa el cálculo de números primos para convertir los datos en piezas de información que no se pueden descifrar si no se tienen los dos números primos que conforman la multiplicación de un número mayor.

Con la evolución de la tecnología, hemos alcanzado nuevos niveles tecnológicos como la computación cuántica, la cual tiene, entre muchas de sus capacidades, el cálculo rápido de números muy grandes. Esto presenta un problema significativo para los sistemas de encriptación basados en RSA, ya que, por medio de estas máquinas, se pueden encontrar los dos números primos que componen a este número mayor que protege los datos. Teniendo estos dos números primos, los datos encriptados se pueden visualizar, dando a la persona en posesión de estos la posibilidad de vender los datos, hacer daños, chantajear a las personas involucradas, malversación de fondos, paralizar su proceso de producción, entre otros. Millones de personas al año en todo el mundo son víctimas de estafadores que, a través de medios digitales, roban información o acceden a cuentas [23]. A este tipo de estafas o engaños se les conoce como fraude cibernético, existiendo varias modalidades y canales por los que se pueden presentar. Esto afecta a todo tipo de industrias, desde la financiera hasta la agrícola, ya que siempre existirán datos sensibles que no se pueden divulgar.

Llevar a cabo una comunicación segura mediante un sistema de clave privada resulta muy difícil si se quiere que la distribución de claves sea segura. Todo lo contrario ocurre con el sistema de clave pública, ya que renuncia a que ambas partes tengan que acordar y distribuir una clave privada. En la actualidad, no se ha encontrado un algoritmo que pueda factorizar un número entero en tiempo polinomial, lo que hace del RSA un algoritmo muy fuerte en la actualidad.

Con la construcción de un computador cuántico en el que se implemente el algoritmo de Shor [14], se podría factorizar un número entero en tiempo polinomial, lo que resultaría en un problema para el sistema de clave pública. La mecánica cuántica brinda una solución al sistema de clave privada en el que se usa un canal cuántico para transmitir la clave privada. El principio de polarización de fotones podría usarse para desarrollar un protocolo criptográfico que distribuya una clave aleatoria de un solo uso.

Con la aparición de la computación cuántica, el problema de la comunicación segura por un canal clásico fue debatido por personas y grupos involucrados en mantener los estándares de ciberseguridad. La ciberseguridad se genera para que las personas, las empresas y organizaciones puedan aceptar confiablemente el acceso a las diferentes plataformas y, a partir de esto, poder tener respaldos de lo que se realiza. Esto llevó a pensar en formas de aprovechar la gran capacidad de cómputo de estas máquinas para poder tener un protocolo que permitiera mantener una comunicación segura entre los dispositivos. Para esto, se requirió que nuevas bases matemáticas se exploraran y probaran, dando lugar a múltiples protocolos que usaban conceptos de mecánica cuántica para generar esta seguridad. Durante este proceso, se llegó a protocolos que permitían no sólo entablar una comunicación segura con una distribución de claves que no se pudiera descifrar, sino también detectar la presencia de un espía en la comunicación entre las partes. Esto abrió una nueva frontera para la ciberseguridad en la computación cuántica, permitiendo a las personas que se comunican saber casi inmediato si hay un intruso.

Dentro de las distintas propuestas para tener un canal de comunicación seguro, se propuso el protocolo BB84. Este protocolo utiliza distintos principios matemáticos y conceptos de la mecánica cuántica, que se explicarán más adelante, para poder entablar una comunicación sin interrupciones, siempre y cuando no exista un espía escuchando la conversación, el cuál haría que el protocolo deje de funcionar hasta que el espía abandone el canal de comunicación.

El proyecto se basará en la implementación del protocolo mediante un algoritmo que creará los componentes básicos de un computador cuántico, los cuales se usarán para generar la simulación del canal de comunicación que propone el protocolo BB84 y de esta manera, hacer distintas evaluaciones como la eficacia y seguridad, de las que se podrán generar conclusiones sobre si este protocolo tiene lo que se necesita para dar los primeros pasos a un mundo más seguro digitalmente, ahora que la computación cuántica se encuentra tan cerca de volverse comercial.

1.1.1. Formulación

¿Cómo generar un canal de comunicación soportado en la computación cuántica basado en el protocolo BB84 en comparación con métodos tradicionales de encriptación?

1.1.2. Sistematización

- ¿Qué conceptos matemáticos se utilizaron para proponer el protocolo BB84?
- ¿Por qué se puede seleccionar el protocolo BB84 para proponer una comunicación segura en un canal cuántico y uno clásico?
- ¿Qué parámetros se usarán para elaborar una implementación en Python del protocolo BB84?
- ¿Cómo implementar un canal de comunicación seguro utilizando el protocolo BB84?
- ¿Qué tan importante es la ciberseguridad en un canal de comunicación para nuestro futuro?

1.2. Objetivos

1.2.1. Objetivo General

El objetivo de esta tesis es desarrollar un canal de comunicación simulado para la distribución de claves simétricas soportado en la computación cuántica y la ciberseguridad.

1.2.2. Objetivos Específicos

1. Realizar un estudio de literatura sobre la computación cuántica, los principios de incertidumbre de Heisenberg, los protocolos de claves de distribución cuánticas y las implementaciones de estos protocolos.

2. Diseñar un algoritmo que simule un canal de comunicación de acuerdo con el protocolo BB84.

3. Realizar la implementación del algoritmo diseñado en el lenguaje de programación Python.

4. Validar los parámetros de seguridad que cumple la implementación del algoritmo del protocolo BB84 según ETSI GS QKD 005 V1.1.1 (2010-12).

1.3. Justificación

La computación cuántica es un tema que involucrará a todas las empresas e instituciones debido al avance tecnológico que traerá consigo. Esta tecnología permitirá crear sistemas más robustos de encriptación de datos, comunicación segura y protección a los usuarios. También podrá brindar mejoras en las empresas a través de la reducción de tiempo de ejecución y optimización de los algoritmos, como lo propone el banco BBVA con algún algoritmo que busque "agrupar los activos de una cartera de inversión" [20]. También, en un comunicado periodístico reciente se ha mostrado el interés que tienen múltiples países desarrollados en apostar por esta nueva tecnología. Los distintos esfuerzos mostrados por estos países buscan que esta tecnología tenga aplicaciones en distintos sectores, como: energía, finanzas y defensa. (Véase [21]).

Este trabajo permitirá concientizar al lector sobre el impacto que traerá la computación cuántica en la vida de las personas. Aunque esta tecnología pueda traer consigo un avance considerable, también puede significar una preocupación. Se volverá una necesidad para los países contar con algún computador cuántico para evitar la vulnerabilidad que tendrán a futuro los computadores tradicionales. Aunque los actuales computadores cuánticos no vulneran algún algoritmo criptográfico, realmente no se conoce el alcance que pueda tener esta tecnología y las implicaciones que puedan resultar.

La presente investigación se enfocará en el estudio del protocolo BB84 para la distribución de claves cuánticas, ya que se estima que en los próximos años los protocolos de distribución tradicionales pasarán a ser obsoletos. Se considera que con la mecánica cuántica se puedan lograr mejores

resultados de cómputo en un periodo de tiempo exponencialmente menor al actual. También permitirá mostrar cómo funciona un protocolo de distribución de claves cuánticas y la importancia de detectar un espía en un canal de comunicación. Esto podría asegurar a los extremos de cada canal en qué momento puede ser seguro comunicarse a través de este.

1.4. Delimitaciones y Alcances

El proyecto consistirá en una implementación del protocolo BB84 en el lenguaje de programación Python; se incluirán las librerías propias de Python. Sin embargo, se dejarán de lado las librerías que simulan el proceso del protocolo. Pese a que se implementará un protocolo cuántico, esta implementación será realizada y ejecutada en computadores tradicionales. Por lo anterior, la corrección del algoritmo desarrollado será a partir de lo conocido teóricamente de los resultados esperados. Se buscará a partir de los resultados obtenidos realizar conclusiones personales que dimensionen lo que traerá consigo la computación cuántica.

Para la validación de la implementación desarrollada se utilizará la especificación ETSI GS QKD 008 V1.1.1 (2010-12)[31], que describe propiamente los pasos a seguir a partir del protocolo BB84. Esta especificación se usará para determinar qué requisitos cumple la implementación según las consideraciones tomadas al momento de diseñar el algoritmo y probar su funcionamiento. Se propondrá, basado en las conclusiones encontradas, si lo implementado puede considerarse como una posible solución dentro de un computador tradicional para ser expandido posteriormente con nuevas técnicas y desarrollos tecnológicos.

1.4.1. Entregables

Como parte de la finalización de este trabajo de grado se entregará un algoritmo escrito en el lenguaje de programación Python, en el cual se verá reflejado el uso del protocolo BB84, utilizando distintos archivos para la creación de este protocolo que fue realizado por nosotros mismos, basándonos en la teoría matemática detrás de cada uno de sus componentes. Este programa también contará con un segmento donde se hará una simulación que permitirá probar la funcionalidad de nuestra implementación, dando paso a un análisis de eficiencia y acercamiento a las otras implementaciones existentes con el fin de dar una respuesta a la formulación del problema.

Capítulo 2

Desarrollo del Proyecto

2.1. Marco de Referencia

2.1.1. Áreas Temáticas

Las áreas temáticas que abarca el proyecto se verán reflejadas en la figura 2.1, de acuerdo con el sistema de Clasificación Computacional traducida de ACM [22].

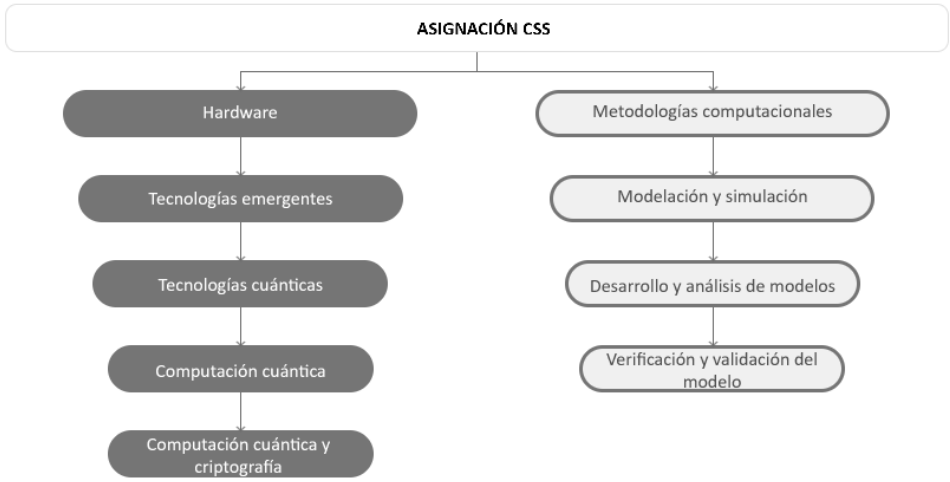

Figura 2.1: Clasificación Computacional ACM

2.1.2. Marco Teórico

2.1.2.1. Ciberseguridad

La ciberseguridad se refiere a la protección de sistemas y datos críticos de los ataques en línea. Involucra estrategias para defender sistemas y aplicaciones en red de amenazas internas y externas, cubriendo áreas como seguridad de infraestructura esencial, redes, aplicaciones, nube, información, usuario final, continuidad del negocio/recuperación tras desastres, almacenamiento y dispositivos móviles. [2]

2.1.2.2. Criptografía

La criptografía es la ciencia que estudia los métodos para asegurar la comunicación, garantizando la privacidad y la autenticidad de los mensajes Ha evolucionado con el tiempo debido a los avances tecnológicos que presentan nuevos desafíos.
El objetivo de esta es mantener en secreto un mensaje. Actualmente no se busca sólo la privacidad o confidencialidad de la información que se envía, sino también garantizar la autenticación del emisor y receptor. Inicialmente, se utilizaban sistemas de clave privada (o encriptación simétrica), donde la misma clave se usa tanto para cifrar como para descifrar. Sin embargo, con la aparición de computadoras más rápidas, estos sistemas fueron superados por la encriptación asimétrica que utiliza una clave pública para cifrar y una clave privada para descifrar, mejorando significativamente la seguridad. [3]

- **Encriptación Simétrica:** Este tipo de encriptación usa la misma clave tanto para cifrar como para descifrar. En este sistema, el emisor y el receptor deben llegar a un acuerdo con la elección de la clave para poder establecer comunicación. Generalmente esta clave es un número que se encuentra en el formato binario y se sugiere que el tamaño de esta debe ser significativamente grande en número de bits, ya que, si existiese un espía que trate de realizar un ataque descifrando la clave por fuerza bruta, deba realizar un mayor número de combinaciones y, por consecuencia, le tome más tiempo. [9]

- **Encriptación Asimétrica:** Este tipo de encriptación usa una clave para cifrar y otra para descifrar. Aunque estas claves están relacionadas, no se puede hallar una a partir de la otra sin conocer la relación exacta que manejan estas. La clave pública se comparte abiertamente para que cualquiera pueda cifrar un mensaje, pero solo el poseedor de la clave privada puede descifrarlo, lo que aumenta la seguridad. [9]

2.1.2.3. Computación Cuántica

La computación cuántica es el campo que utiliza la mecánica cuántica como base para resolver problemas que, en términos computacionales no son posibles de resolver en computadores tradicionales. Explota la superposición de estados y el entrelazamiento cuántico para realizar cálculos complejos de manera exponencialmente más rápida. [4] [5]

- **Cúbits:** La unidad más simple dentro de una computadora cuántica es el cúbit. Esta unidad se representa normalmente como un vector en una esfera unitaria, que puede tomar valores entre 0 y 1 de manera simultánea debido a los principios de la mecánica cuántica. Esto permite alcanzar una combinación lineal de estos dos estados, aprovechando las probabilidades asociadas a cada uno para computar los resultados de ecuaciones matemáticas complejas de manera más rápida.[6]

- **Superposición de Estados:** Es el estado en el que se encuentra un cúbit al representar simultáneamente los valores 0 y 1, mediante probabilidades asociadas a cada estado. Esta propiedad permite realizar múltiples cálculos de forma paralela con un solo cúbit, lo que conlleva un crecimiento exponencial en la velocidad de cómputo a medida que aumenta la cantidad de cúbits utilizados. Sin embargo, para obtener los resultados de estos cálculos, es necesario medir el estado del cubit, lo que lo hace colapsar en uno de los dos estados (0 o 1), perdiendo la información de la superposición. [7]

2.1.2.4. Criptografía Cuántica

La criptografía cuántica busca permitir que dos usuarios se comuniquen de manera más segura que con la computación tradicional. Los métodos tradicionales dependen de la seguridad matemática para realizar cálculos complejos que, en teoría, tomarían millones de años en completar, lo que hace que los datos interceptados pierdan relevancia. Aunque actualmente no existen modelos que puedan realizar estos cálculos de manera eficiente en computadores tradicionales, se espera que en el futuro esto no sea posible. Este avance ha impulsado a la creación de nuevas medidas de seguridad basadas en conceptos nuevos como la mecánica cuántica.[12]

- **Entrelazamiento Cuántico:** Este fenómeno se presenta cuando dos o más cúbits se encuentran entrelazados de tal manera que sus estados son los mismos hasta que sean medidos por una entidad externa que pueda sacar la información. Esta información que sale del entrelazamiento es solo uno de los posibles resultados que se encuentran en una superposición de estados de dos o más cúbits, lo que permite que los cálculos que se requieran se hagan de manera más rápida que usando un solo cúbit, generando un crecimiento exponencial al momento de conseguir resultados. [8]

- **Principio de Incertidumbre de Heisenberg:** Este principio consta de que no es posible medir un estado cuántico en un sistema sin afectar el sistema en cuestión. Esto permite saber cuándo un sistema está siendo espiado, ya que al afectar el sistema de transmisión de los datos por medio de los estados cuánticos el receptor de estos datos nunca obtendrá el mensaje, permitiendo determinar que este canal no es seguro en ese momento, por lo que deberá cambiar de canal o esperar a que la o las personas que se encuentran tratando de descifrar los mensajes que se están pasando se desconecten. Este principio también dice que ciertos pares de propiedades físicas están relacionadas, de tal manera que evita que un espía que mida una propiedad no pueda medir el resto de propiedades de manera simultánea [10][11].

- **Principio de Polarización de Fotones:** Este principio se basa en que un espía no podrá copiar cúbits desconocidos, es decir, estados cuánticos desconocidos. Esto se debe a que los fotones de luz se pueden polarizar en una dirección específica haciendo que, en una instancia de medición donde no se sepa la forma en la que se encuentra polarizado este fotón, todas las mediciones siguientes se verán afectadas por la primera decisión. [10]

2.1.2.5. Protocolo Cuántico de Distribución de Claves

Un canal de comunicación es el medio físico a través del cual dos o más dispositivos transportan información. Un canal de comunicación seguro es aquel donde sólo los dispositivos involucrados pueden emitir y recibir información; ningún dispositivo o agente externo debería poder tener acceso a la información transmitida. Para mejorar la seguridad en la comunicación entre dos dispositivos, se implementan diversos protocolos que aseguran que la información transmitida se mantenga privada e integra. Estos protocolos de distribución de claves se dividen entre simétrico y asimétrico.

Un ejemplo de un protocolo asimétrico es el RSA, que utiliza una clave privada para cada usuario que sirve para desencriptar el mensaje y una clave pública que se usa para encriptar el mensaje. Se ha comprobado que esta es una manera efectiva de establecer una comunicación segura, ya que no se necesita la misma clave para encriptar los datos.

Por otro lado, los protocolos simétricos emplean una única clave para encriptar y desencriptar la información compartida, pero presentan el riesgo de que la clave debe ser compartida previamente con el usuario receptor. Esto introduce una vulnerabilidad, ya que al compartir la clave por un canal inseguro, existe el riesgo de que un espía intercepte la comunicación y acceda a la información compartida. Debido a la falta de un método seguro para distribuir las claves simétricas, este método no se aplica en los protocolos de comunicación estándar, prevaleciendo el uso de los protocolos asimétricos.

En un protocolo de distribución de claves cuánticas, no existe distinción entre simétrico o asimétrico. Un ejemplo común para explicar el protocolo de distribución de claves cuánticas a tres actores: El emisor (Alice), el receptor (Bob) y un espía (Eva), y dos canales de comunicación: uno clásico para el acuerdo de claves y la depuración de información, y otro cuántico por el cúal se envían fotones. El espía (Eva) puede acceder a los dos canales y usar todos los medios que las leyes de la mecánica cuántica para intentar descifrar la clave. El emisor (Alice) y el receptor (Bob) envían un solo fragmento de su clave para detectar al espía; si no se detecta o si el espía obtiene muy poca información, se llega a un acuerdo sobre la clave. [15]

2.1.2.6. Protocolo BB84

El funcionamiento del protocolo se basa en distintos conceptos mencionados previamente, ya que es una distribución de claves cuánticas de forma simétrica que utiliza como base los principios de la mecánica cuántica, los principios de incertidumbre de Heisenberg y la polarización de fotones. Este protocolo considera la existencia de Alice (emisor) y Bob (receptor), quienes desean entablar

una comunicación segura por medio de un canal cuántico. Este canal cuántico se puede considerar como un cable de fibra óptica o un espacio abierto.

Ambos partes necesitan saber cuál es la polarización de fotones que escogió Alice para poder comunicarse, por lo que usan un canal de comunicación tradicional que debe ser previamente autenticado para pasarse estos datos. Una vez que se conoce la polarización de los fotones, Bob podrá recibir la información transmitida a través del canal cuántico.

Este protocolo también considera la posibilidad de que Eve (espía) tenga acceso al canal de comunicación cuántico. Como se mencionó previamente, Eve puede recibir los fotones que se están transmitiendo, pero no podrá leer ni retransmitir los fotones de manera correcta debido a que no conoce la polarización de los mismos. Esto hará que el mensaje llegue a Bob de manera incorrecta, generando una alerta de que existe alguien interceptando la comunicación. Como parte de este proceso, Alice y Bob deberán decidir si debe continuar la conversación o esperar a que Eve se retire del canal cuántico para seguir con la transmisión de los fotones.[13]

2.1.3. Trabajos Relacionados

2.1.3.1. Implementation of the protocol BB84 for encryption of a message [17]

Este documento trata sobre la implementación del protocolo BB84 para el cifrado de un mensaje que se lleva a cabo mediante un montaje óptico en el laboratorio de óptica cuántica de la Universidad de Los Andes. La configuración implica el uso de los elementos ópticos lineales y no lineales para trabajar con la polarización de los fotones.

2.1.3.2. Implementación del Protocolo de Distribución Cuántica de Claves, Protocolo BB84 [18]

Este documento trata sobre la implementación del protocolo BB84 para la distribución cuántica de claves y se discute la relevancia de la investigación en nuevas formas de cifrado. Se contemplan dos propuestas de implementación: la primera es con uso de LibQuantum en el lenguaje de programación C para simular el comportamiento de una computadora cuántica, y la segunda es sin el uso de las librerías cuánticas en el lenguaje de programación $C\#$.

2.1.3.3. Implementation of QKD BB84 Protocol in Qiskit [19]

Este documento trata sobre la implementación del protocolo de distribución de claves cuánticas BB84 en Qiskit, que es una plataforma de computación cuántica desarrollada por IBM. Este proyecto presenta una simulación del protocolo BB84. Se discuten los conceptos básicos de criptografía, las propiedades cuánticas necesarias para entender el protocolo BB84, y las herramientas para la simulación cuántica. Esta implementación del protocolo es hecha en Python usando Qiskit y se ejecuta en simuladores locales y en el servicio de computación cuántica en la nube de IBM.

2.1.3.4. Quantum Cryptography on IBM QX [28]

Este documento trata sobre los resultados obtenidos por una implementación realizada en el software IBM QX que es un servicio de computación cuántica en la nube de IBM para hacer una comparación entre los resultados obtenidos y los resultados teóricos que se tienen para el protocolo, junto con un análisis estadístico de las pruebas usando la desviación estándar de los resultados. Por último hacen cálculos de probabilidad de la detección de un espía en el canal de comunicación para ser comparados con los resultados teóricos esperados. Los cálculos realizados muestran que a mayor número de cúbits, el porcentaje de detección de un espía es mayor.

2.1.3.5. Experimental Side Channel Analysis of BB84 QKD Source [29]

Este documento trata de como una implementación del protocolo BB84 utiliza diodos para transmitir pulsos débiles coherentes de información los cuales pueden no tener las mismas características. Esto permite a los autores calcular posibles filtraciones de información al espía por medio de ataques a un canal secundario por medio de una evaluación mutua de información entre la fuente y el espía. Para finalizar los autores proponen métodos para reducir la filtración de la información al espía.

2.1.3.6. Software Development for B92 Quantum Key Distribution Communication Protocol [30]

Este documento trata sobre las implicaciones de software que llegan con las primeras dos fases de la distribución de claves cuánticas, la extracción de claves sin procesar y la estimación de error. Los autores hablan de propuestas para futuros trabajos con el fin de mejorar estas fases. Para finalizar proponen una implementación para la comunicación entre dos usuarios en el lenguaje C++ por medio de una conexión Cliente-Servidor basado en TCP/IP.

2.2. Metodología

2.2.1. Tipo de Estudio

Este trabajo de grado es un estudio exploratorio puesto que pretende familiarizar a los lectores sobre nuevas tecnologías y cómo estas afectan nuestro entorno, junto con la forma en la que un prototipo puede ser de utilidad para demostrar que estos conceptos novedosos tienen una base clara y moderadamente fácil de replicar para ser usada en contextos más complejos.

Para el desarrollo del prototipo se seguirá la metodología SCRUM, en donde a través de Sprints, se permita ir generando valor al proyecto, con la finalidad de lograr los objetivos propuestos, partiendo desde la investigación, hasta la elaboración del prototipo.

2.2.2. Actividades

2.2.2.1. Objetivo 1: Realizar un estudio de literatura sobre la computación cuántica, los principios de incertidumbre de Heisenberg, los protocolos de claves de distribución cuánticas y las implementaciones de estos protocolos.

Para este objetivo se realizan las siguientes actividades:

- Realizar un estudio que permita comprender el concepto y el comportamiento del principio de incertidumbre de Heisenberg.
- Realizar un estudio sobre los protocolos existentes de claves de distribución cuánticas.
- Realizar un estudio sobre las posibles implementaciones existentes de cada uno de los protocolos investigados.

2.2.2.2. Objetivo 2: Diseñar un algoritmo que simule un canal de comunicación de acuerdo con el protocolo BB84

Para este objetivo se realizan las siguientes actividades:

- A partir de las investigaciones realizadas, seleccionar el protocolo de preferencia para el cual se desea realizar una implementación de este.
- Identificar las ventajas y desventajas que se pueden presentar al desarrollar la implementación en el lenguaje de programación Python en comparación con otros lenguajes de programación.

2.2.2.3. Objetivo 3: Realizar la implementación del algoritmo diseñado en el lenguaje de programación Python

Para este objetivo se realizan las siguientes actividades:

- A partir del diseño del algoritmo, codificar la implementación en el lenguaje de programación Python.
- Validar la correctitud del algoritmo.
- Dado el caso, corregir errores del algoritmo para volver a la evaluación el ítem anterior.

2.2.2.4. Objetivo 4: Validar los parámetros de seguridad que cumple la implementación del algoritmo del protocolo BB84 según ETSI GS QKD 005 V1.1.1 (2010-12)[43]

Para este objetivo se realizan las siguientes actividades:

- Realizar un estudio de los distintos estándares establecidos por ETSI GS QKD 005 V1.1.1 (2010-12) para la distribución de claves cuánticas.

- Validar los estándares que cumple el protocolo BB84 según los estándares establecidos por ETSI GS QKD 005 V1.1.1 (2010-12) para la distribución de claves cuánticas.
- Validar las consideraciones de la implementación desarrollada frente a los estándares establecidos por ETSI GS QKD 005 V1.1.1 (2010-12) para la distribución de claves cuánticas.

2.3. Resultados Esperados

- Diseñar un algoritmo que simule un canal de comunicación de acuerdo al protocolo BB84.
- Implementar un prototipo funcional del protocolo BB84 en el lenguaje de programación Python para mostrar cómo funcionan los distintos componentes en un contexto distinto al propuesto para su uso.
- Desarrollar conclusiones personales a partir de las pruebas realizadas y los resultados obtenidos de la implementación.
- Desarrollar conclusiones personales sobre la situación que enfrentará la humanidad con la llegada de la computación cuántica en términos de ciberseguridad a partir de los estándares ETSI GS QKD 005 V1.1.1 (2010-12) para la distribución de claves cuánticas.

2.4. Cronograma

Cronograma de actividades			
Descripción de actividades	**Inicio**	**Duración (semanas)**	**Fin**
Reuniones semanales grabadas con la directora Liliana para mostrar avances, hacer ajustes y dar retroalimentación del proceso.	25/08/2023	47	22/07/2024
Investigar el concepto y el comportamiento del principio de incertidumbre.	02/10/2023	3	16/10/2023
Investigar los protocolos existentes para la distribución de claves cuánticas.	17/10/2023	2	30/10/2023
Investigar implementaciones existentes de los protocolos seleccionados.	31/10/2023	1	06/11/2023
Desarrollar la implementación del protocolo BB84 en el lenguaje de programación seleccionado.	07/11/2023	29	30/05/2024
Realizar pruebas y corregir posibles errores de la implementación.	31/05/2024	5	07/07/2024
Finalizar el documento de proyecto de grado con los resultados obtenidos.	08/07/2024	2	22/07/2024

2.5. Recursos

2.5.1. Humanos

Las personas encargadas de realizar el trabajo de grados son: Dilan Andres Correa Ortíz y Santiago Torres Rincón. La directora del trabajo de grado es Liliana Gutiérrez Rancruel, Ingeniera de Sistemas y Telecomunicaciones de la Universidad Libre Seccional Cali, Magíster en Informática Educativa de la Universidad Libre Seccional Cali y actualmente estudiante del Doctorado en Gestión Tecnológica e Innovación de la Universidad Autónoma de Querétaro en México. Tiene experiencia y conocimiento en Acreditación y Aseguramiento de la Calidad para los Programas de Ingeniería, así como Formulación, Diseño, Implementación de Planes Educativos y de Tecnologías de Información y Comunicación aplicadas en el Sector Educativo. Además, posee conocimiento en

la Formulación de Proyectos, Diseño, Gestión y Mantenimiento de Redes, IPV6, Ciberseguridad, Realidad Aumentada y Auditoría. Es docente e investigadora, e instructora de la Academia de Cisco Universidad Javeriana. Sus habilidades y aptitudes incluyen comunicación, liderazgo, desarrollo de proyectos y comunicación asertiva con alumnos y docentes. Genera conocimiento con los estudiantes empleando metodologías de innovación estratégica como design thinking, y didácticas que permitan generar experiencias que conecten y detonen la creatividad, la indagación, el trabajo en equipo, la adaptabilidad, la comunicación, el aprender a aprender y el pensamiento disruptivo.

2.5.2. Técnicos

Para el desarrollo de este trabajo de grado se requiere el uso de computadores que cuenten con una buena capacidad de procesamiento computacional a nivel de CPU y GPU para obtener una buena aproximación a lo que se tiene con una computadora cuántica.

Computador 1: Escritorio Personal

- **Procesador:** Intel(R) Core(TM) i5-9400 @ 2.90GHz (6 CPUs), 2.90GHz
- **RAM:** 16.0 GB
- **Tarjeta Gráfica:** NVIDIA GeForce RTX 2060 de 14 GB

Computador 2: Laptop Personal 1

- **Procesador:** 11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz
- **RAM:** 16.0 GB
- **Tarjeta Gráfica:** Intel iRIS Xe

Computador 3: Escritorio Pontificia Universidad Javeriana Cali

- **Procesador:** 12th Gen Intel(R) Core(TM) i7
- **RAM:** 32.0 GB
- **Tarjeta Gráfica:** NVIDIA GeForce RTX 3060 de 12,0 GB

2.5.3. Presupuesto

Este presupuesto es un estimado teniendo en consideración el costo de los dispositivos a nuestra disposición y el costo del personal científico.

RECURSOS HUMANOS				
Personas	**Total Sin Adiciones**	**% Adicional**	**Precio Hora**	**Total**
Dilan Correa	$1.500.000	$1.000.000	$9.375	$2.500.000
Santiago Torres	$1.500.000	$1.000.000	$9.375	$2.500.000
Liliana Gutierrez	$12.000.000	$3.000.000	$75.000	$15.000.000

RECURSOS GENERALES			
Rubros	**Fuentes**		**Total**
	Propios	**PUJ**	
Equipos	$15.000.000	$4.924.796	$19.924.796
Bibliografía	$0	$4.000.000	$4.000.000
Personal Científico	$5.000.000	$2.500.000	$7.500.000
Materiales e Insumos	$0	$0	$0
Servicios Técnicos	$0	$0	$0
Viajes	$0	$0	$0
Salidas de Campo	$0	$0	$0
Eventos Académicos	$0	$0	$0
Publicaciones y Patentes	$0	$0	$0
Software	$0	$0	$0
Administración	$0	$0	$0
Seguimiento y Evaluación	$0	$0	$0
Total	$20.000.000	$11.424.796	$31.424.796

Capítulo 3

Investigación

3.1. Conceptos adicionales relevantes

Durante el desarrollo e investigación del proyecto, se identificaron conceptos adicionales que son cruciales para una comprensión completa y detallada del tema. Se identificaron distintos mecanismos que buscan reforzar la seguridad que provee el protocolo BB84 para la transmisión de la información dentro del canal tradicional y así mismo poder completar la comunicación entre dos agentes (Alice y Bob, como se conoce en el marco teórico). Estos conceptos no fueron abordados en la fase inicial del marco teórico, pero su inclusión es fundamental para asegurar la integridad y seguridad de la implementación del protocolo BB84.

3.1.1. Canales clásicos y cuánticos

La implementación de protocolos cuánticos de distribución de claves requiere el uso de dos tipos de canales: cuánticos y clásicos. El canal cuántico se usa para la transmisión de estados cuánticos (fotones) que contienen la información de la clave, mientras el canal clásico se utiliza para la comunicación de información adicional necesaria para la corrección de errores y la privacidad amplificada.

3.1.1.1. Canal clásico [31]

El canal clásico es un canal de comunicación convencional que permite a un agente A (Alice) y un agente B (Bob) intercambiar la información sobre las bases de medición y otros datos necesarios para la reconciliación de información y la privacidad amplificada. La seguridad de este canal puede garantizarse en el proceso de la reconciliación de información a partir de los porcentajes de error aceptado por cada etapa del proceso de comunicación.

3.1.1.2. Canal cuántico [31]

El canal cuántico es el medio a través del cual se envían los fotones polarizados. Este canal eventualmente se compone de fibra óptica. La seguridad de este canal se basa en los principios de la mecánica cuántica, como la imposibilidad de clonar los estados cuánticos y la detección de la presencia de un espía debido al principio de incertidumbre de Heisenberg.

3.1.2. Mecanismos de verificación

En el contexto de la implementación del protocolo BB84, es fundamental que las cadenas de bits utilizadas sean realmente pseudoaleatorias, sobre todo para probar la correctitud del código con diferentes casos de prueba. La aleatoriedad en la generación de bits es crucial para la seguridad y eficiencia del protocolo. Una cadena de bits pseudoaleatorios garantiza que no haya patrones predecibles que puedan ser explotados por un atacante para interceptar o deducir la clave cuántica compartida entre dos agentes (Alice y Bob). Si los bits utilizados para determinar la polarización de los fotones no son verdaderamente aleatorios, un espía (Eva) podría detectar patrones y predecir la clave, comprometiendo la seguridad del sistema. La utilización de cadenas de bits pseudoaleatorios asegura que cada bit es independiente y tiene una probabilidad igual de ser 0 o 1, dificultando así cualquier intento de interceptación o predicción por parte de un posible espía.

Para evaluar si una cadena de bits generada es realmente pseudoaleatoria, se utilizan diversas pruebas estadísticas basadas en el documento "A Statistical Test Suite for Random and Pseudorandom Number Generators for Criptographic Applications" del NIST [32]. Este documento proporciona un conjunto de pruebas estadísticas diseñadas para evaluar la calidad de los generadores de números aleatorios y pseudoaleatorios. A continuación, se describen los métodos específicos implementados en la función '*validar_aleatoriedad*' para evaluar, como su nombre lo indica, la aleatoriedad de la cadena de bits:

3.1.2.1. Prueba de frecuencia de Monobit

Esta prueba verifica si el número de 1s y 0s en la cadena de bits es aproximadamente igual. Una diferencia significativa indicaría una falta de aleatoriedad. La función calcula la diferencia absoluta entre el número de 1s y 0s, normalizada por la longitud de la cadena.

```
# Prueba de frecuencia monobit
num_1s = cadena_bits.count('1')
num_0s = cadena_bits.count('0')
prueba_monobit =  abs(num_1s - num_0s) / len(cadena_bits)
```

Figura 3.1: Prueba de monobit
Fuente: Autores

3.1.2.2. Transformada Discreta de Fourier (DFT)

La DFT se usa para analizar la frecuencia de la secuencia de bits. Al calcular la Transformada Discreta de Fourier de la secuencia de bits, se obtienen las magnitudes de las frecuencias presentes. Luego, se aplica una prueba de Chi-cuadrado para verificar si estas magnitudes siguen una distribución uniforme.

```python
#DFT
secuencia_numeros = np.array([int(bit) for bit in cadena_bits])
frecuencia_secuencia = np.fft.fft(secuencia_numeros)
magnitudes = np.abs(frecuencia_secuencia)

# Usamos el test de Chi-cuadrado para verificar si las magnitudes siguen una distribución
bins = len(magnitudes) // 2  # Solo analizamos la mitad positiva del espectro
hist, bin_edges = np.histogram(magnitudes[:bins], bins=bins, density=True)
chi_squared = np.sum((hist - np.mean(hist))**2 / np.mean(hist))
p_value = chi2.sf(chi_squared, bins - 1)
```

Figura 3.2: Prueba Transformada de Fourier
Fuente: Autores

Un p_value alto en esta prueba podría indicar que las frecuencias están distribuidas de manera uniforme, lo que podría significar una aleatoriedad.

3.1.2.3. Entropía

La entropía mide el grado de incertidumbre o aleatoriedad en la cadena de bits. Una alta entropía podría significar que la cadena de bits no contiene patrones repetitivos y que cada bit es impredecible.

```python
# Entropía
frecuencias = Counter(cadena_bits)
entropia = -sum((frecuencia / len(cadena_bits)) * math.log2(frecuencia / len(cadena_bits))
          for frecuencia in frecuencias.values())
```

Figura 3.3: Prueba de entropía
Fuente: Autores

3.1.3. Métodos de reconciliación de información

La conciliación de información es un paso importante en los protocolos de distribución de claves cuánticas (QKD). Este proceso lo que busca es garantizar que tanto el emisor (Alice) como el receptor (Bob) obtengan cadenas de bits idénticas después de la transmisión a través del canal cuántico, a pesar de la presencia de errores causados por la naturaleza del canal o por la presencia de un espía (Eva) [34]. Existen varios métodos para realizar una conciliación de información, entre los más destacados se encuentran el algoritmo CASCADE [38] y los códigos de baja densidad de paridad [35].

3.1.3.1. Algoritmo Binario [37]

El algoritmo Binario es un método utilizado que tiene como enfoque la idea de dividir cadenas de bits en segmentos más pequeños y realizar comparaciones binarias para identificar y corregir errores.

Para su implementación, el algoritmo divide la cadena de bits en segmentos y compara estos segmentos entre Alice y Bob. Si se detecta una discrepancia, el segmento se subdivide en parte más pequeñas y se repite la comparación hasta que se identifique el bit erróneo. Este proceso se repite para toda la cadena de bits hasta que todas las discrepancias sean corregidas.

Este algoritmo es relativamente simple de implementar y comprender, lo que lo hace accesible para una amplia gama de aplicaciones. En particular, es eficiente cuando la tasa de error en la transmisión es baja, ya que requiere menos subdivisiones y comparaciones. En comparación con otros métodos como CASCADE, este algoritmo puede requerir menos comunicación entre Alice y Bob. Sin embargo, al requerir la cadena de ambos para poder resolver el conflicto podría resultar más inseguro en comparación con los códigos LDPC.

3.1.3.2. Algoritmo CASCADE [38]

El algoritmo CASCADE es uno de los métodos pioneros y más reconocidos para la reconciliación de información de la distribución de claves cuánticas (QKD). En este algoritmo lo que se realiza es un procedimiento iterativo que busca corregir errores mediante el intercambio de información adicional entre las partes involucradas, Alice y Bob.

El funcionamiento del algoritmo se basa en dividir las cadenas de bits en bloques en cada iteración. Posteriormente, se comparan las paridades de dichos bloques para identificar y corregir errores. Esta técnica se repite hasta que se logre la reconciliación completa de la información.

El algoritmo CASCADE se caracteriza por su simplicidad en cuanto a implementación y compresión, lo que lo convierte en una opción atractiva para su uso práctico. Además, demuestra un buen desempeño cuando la tasa de error en el canal cuántico es baja. Sin embargo, es importante tener en cuenta que requiere múltiples rondas de comunicación entre Alice y Bob, lo que puede ser ineficiente en términos de tiempo y recursos, además de que esto podría ser más vulnerable para ataques de ciberseguridad conocidos como 'man in the middle' o hombre en el medio. A su vez, su eficiencia se ve afectada negativamente al trabajar con grandes cadenas de bits o altas tasas de error, volviéndose un método poco práctico ante estos escenarios.

3.1.3.3. Código de baja densidad de paridad (LDPC) [35]

Los códigos LDPC (LDPC del inglés low density parity check) son un tipo de código de corrección de errores que se destaca por su uso de una matriz de paridad de baja densidad para la detección y corrección de errores. Estas características los convierten en una opción atractiva para

la reconciliación de información en QKD, ya que son capaces de manejar grandes volúmenes de datos de manera eficiente y ofrecen robustez frente a las altas tasas de error.

En la implementación del LDPC para la reconciliación de información, Alice y Bob comparte una matriz de paridad [36] y emplean procedimientos de decodificación iterativa para corregir los errores en sus cadenas de bits. Este proceso se basa en los principios de la teoría de códigos y en algoritmos de decodificación como el algoritmo de propagación de creencias (belief propagation).

Estos códigos presentan ventajas y desventajas que deben ser consideradas en su aplicación para QKD como las mencionadas en el primer párrafo de esta explicación. Además, estos códigos tienen menor interactividad en comparación con otros método como el CASCADE, también los códigos LDPC pueden adaptarse a QBER (Tasa de error cuántico en bits, traducido de su significado original en inglés "Quantum Bit Error Rate") altas, teniendo en cuenta que el QBER puede aumentar entre distancias más largas. Sin embargo, su implementación y decodificación pueden ser computacionalmente intensivas, y su eficiencia depende en gran medida del diseño de la matriz de paridad utilizada.

3.1.3.4. Algoritmo Winnow [39]

El algoritmo Winnow tiene un enfoque más sofisticado y eficiente en la corrección de errores, especialmente en entornos con tasas de error más altas. Este algoritmo es un método iterativo que divide la cadena de bits en bloques y utiliza la información de paridad para detectar y corregir errores. El proceso consiste en las siguientes etapas:

1. **División en bloques:** La cadena de bits se divide en bloques de igual tamaño.
2. **Cálculo de paridad:** Se calcula la paridad de cada bloque y se compara entre Alice y Bob.
3. **Identificación de errores:** Si se detecta una discrepancia en la paridad, el bloque se subdivide y se repite el cálculo de paridad para identificar los bits erróneos.
4. **Corrección de errores:** Una vez identificados, los bits erróneos se corrigen. Este proceso se repite iterativamente hasta que no se detecten más errores en los bloques.

Este algoritmo es muy eficiente para la corrección de errores, especialmente en entornos con tasas de error moderadas a altas, es escalable y puede manejar cadenas de bits de mayor longitud de manera efectiva y, sobre todo, lo más importante para el desarrollo de este trabajo, requiere menos interactividad y comunicación entre Alice y Bob en comparación con otros métodos como el CASCADE.

Cabe destacar que no es un algoritmo perfecto, pues es más complejo de implementar en comparación con algoritmos más simples como el algoritmo binario. La elección del tamaño de los bloques puede afectar significativamente la eficiencia del algoritmo ya que tamaños de bloque subóptimos

pueden reducir su efectividad, a su vez, puede requerir más recursos computacionales para el cálculo de paridad y la identificación de errores en comparación con métodos más simples.

3.1.3.5. Matrices de paridad [40]

Una matriz de paridad es una herramienta fundamental en los códigos de corrección de errores, como los códigos LDPC y el algoritmo Winnow. Una matriz de paridad es una estructura matemática que se utiliza para verificar y corregir errores en la transmisión de datos. Se compone de filas y columnas de bits (0s y 1s), donde cada fila representa una ecuación de paridad. La densidad baja en los códigos LDPC significa que la mayoría de los elementos de la matriz son ceros, lo que permite una mayor eficiencia en el procesamiento y la corrección de errores.

3.1.4. Privacidad amplificada [41]

La privacidad amplificada es una técnica utilizada para mejorar la seguridad de las claves compartidas. Después de la reconciliación de información, Alice y Bob poseen una clave común que puede haber sido parcialmente conocida por un espía (Eva). La privacidad amplificada se emplea para reducir la cantidad de información que Eva podría conocer, generando una clave final más corta, pero altamente segura. Este proceso implica el uso de funciones hash universales y técnicas de comprensión para minimizar la información accesible al espía.

3.1.5. Propagación de creencias (Belief propagation) en códigos LDPC [42]

La propagación de creencias (Belief propagation, BP) es un algoritmo iterativo utilizado para la decodificación de códigos de corrección de errores, especialmente en códigos de paridad de baja densidad (LDPC). Este algoritmo, también es conocido como el algoritmo de suma-producto en el que se utilizan gráficos bipartitos, conocidos también como gráficos de Tanner, que representan las relaciones de paridad de los códigos LDPC.

El gráfico de Tanner está compuesto por dos tipos de nodos: nodos de variable (que representan los bits de datos) y nodos de verificación (que representan las restricciones de paridad). Este algoritmo iterativo se compone de 3 procedimientos:

1. Inicialización: Se parte de unos mensajes iniciales basados en la probabilidad logarítmica (LLR) de que el bit correspondiente sea 0 o 1.

2. Actualización de mensajes: Se realiza una actualización del mensaje donde los nodos de verificación actualizan los mensajes enviados a los nodos de variable basándose en los mensajes recibidos de otros nodos de variable conectados.

3. Decisión final: Después de un numero definido de iteraciones, se calcula el LLR total para cada nodo variable.

Como tarea para el lector, se recomienda leer el documento [39], sección 2.4 "Sum-product Decoding for Information Reconciliation", pág. 1721-1722, donde se hace una explicación matemática más detallada de estos procesos.

Capítulo 4

Implementación y Resultados

4.1. Metodología de desarrollo

La elección de la metodología SCRUM para el desarrollo de nuestra simulación fue una decisión fundamentada en varios factores clave que se alinean con los objetivos y necesidades de nuestros alcances y delimitaciones. Al ser una estructura iterativa e incremental nos ofrece una flexibilidad y adaptabilidad esencial en el desarrollo de simulaciones, donde los requisitos pueden evolucionar y ajustarse conforme avanza el desarrollo del producto final para de esta manera cumplir con los objetivos pactados inicialmente.

Una vez definida la metodología, se definieron la cantidad de Sprints (entregas en rango de tiempo), de dos semanas cada una, junto con las tareas propuestas para estas etapas. Dentro de estos Sprints se definieron una serie de objetivos derivados de las actividades inicialmente planteadas en el capítulo 2, las cuales ofrecen un mejor acercamiento de los distintos conceptos e implementaciones requeridos.

- **Sprint 1:** Este sprint estuvo compuesto por las siguientes actividades.
  - Analizar la normativa ETSI GS QKD 005 V1.1.1 (2010-12) para definir los distintos componentes necesarios para realizar una simulación segura y cercana a la propuesta para un entorno real.
  - Descomponer los componentes principales necesarios en secciones menores para diseñar y posteriormente implementar.
- **Sprint 2:** Este sprint estuvo compuesto por las siguientes actividades.
  - Investigar en artículos los conceptos relacionados con la generación de bits aleatorios y cómo validar que estos son aptos para su uso en entornos criptográficos.
  - Diseñar e implementar la validación y generación de bits aleatorios usando las distintas técnicas investigadas.
- **Sprint 3:** Este sprint estuvo compuesto por las siguientes actividades.
  - Investigar en artículos la forma apropiada para simular la transmisión de información en un canal cuántico usando bits.

- Diseñar la estructura de la clase que contendrá la información simulada de los cubits para cada uno de los agentes.

- Diseñar e implementar la conciliación de bases transmitidas en el canal cuántico simulado.

- **Sprint 4:** Este sprint estuvo compuesto por las siguientes actividades.

  - Investigar en artículos los mecanismos de validación dentro del canal tradicional para la reconciliación de información.

  - Analizar los distintos mecanismos propuestos para diseñar e implementar aquel que prevea la mayor seguridad entre los agentes.

- **Sprint 5:** Este sprint estuvo compuesto por las siguientes actividades.

  - Investigar en artículos el funcionamiento del algoritmo LDPC para la reconciliación de información.

  - Diseñar e implementar el algoritmo LDPC bajo las restricciones definidas en las delimitaciones y alcances de no usar librerías que no fueran las estándar de Python.

- **Sprint 6:** Este sprint estuvo compuesto por las siguientes actividades.

  - Corrección de errores y validación de la normativa frente a la implementación.

  - Probar y validar el funcionamiento de los distintos componentes implementados por separado y cómo conjunto para dar una respuesta coherente.

4.1.1. Diagrama de caso de uso

Como parte de la metodología de desarrollo ágil de software, a continuación se mostrará y se explicará acerca del diagrama de casos de uso propuesto para la simulación desarrollado con la Figura 4.1, pues este cumple con todas las acciones e interacciones que se proponen en el.

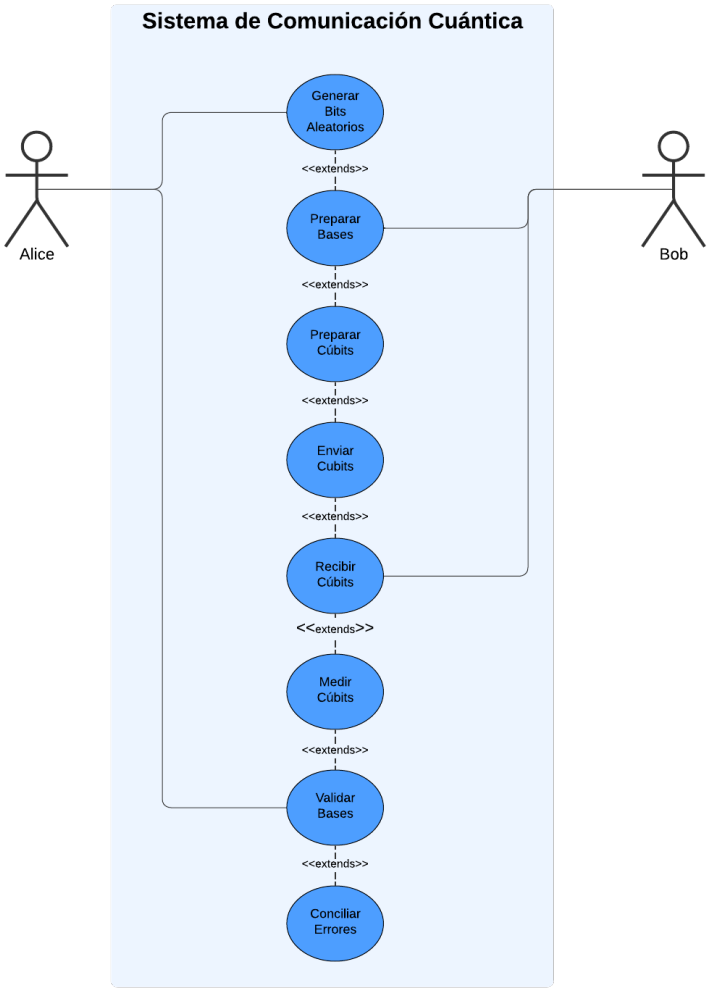

Figura 4.1: Diagrama de Caso de Uso Fuente: Autores

Los agentes Alice y Bob interactúan con el sistema en etapas diferentes pero hacen que las acciones después de su interacción puedan ejecutarse permitiendo tener un flujo directo de principio a fin.

4.1.2. Historias de Usuario

4.1.2.1. Historia 1: Generar Bits Aleatorios

- **Como:** Usuario (Alice)
- **Quiero:** Generar una cadena de bits aleatorios
- **Para:** Poder utilizarlos en la preparación de cubits.
- **Criterios de Aceptación:**
  - Los bits generados deben ser verdaderamente aleatorios.
  - La longitud de la cadena debe ser configurable.

4.1.2.2. Historia 2: Preparar Cúbits

- **Como:** Usuario (Alice)
- **Quiero:** Preparar cúbits utilizando los bits aleatorios generados y bases seleccionadas
- **Para:** Codificar la información que enviaré a Bob.
- **Criterios de Aceptación:**
  - Los cúbits deben ser preparados utilizando las bases (rectilíneas o diagonales) seleccionadas aleatoriamente.
  - Debe haber un mecanismo para verificar la correcta preparación de los cúbits.

4.1.2.3. Historia 3: Enviar Cúbits

- **Como:** Usuario (Alice)
- **Quiero:** Enviar los cúbits preparados a Bob
- **Para:** Que él pueda recibir la información codificada.
- **Criterios de Aceptación:**
  - Los cúbits deben ser enviados a través de un canal cuántico.
  - Debe garantizarse la mínima pérdida de cúbits durante la transmisión.

4.1.2.4. Historia 4: Recibir Cúbits

- **Como:** Usuario (Bob)
- **Quiero:** Recibir los cúbits enviados por Alice
- **Para:** Decodificar la información usando mis propias bases seleccionadas.
- **Criterios de Aceptación:**
  - Bob debe recibir los cúbits con la mínima interferencia posible.
  - Bob debe ser notificado de cualquier pérdida de cúbits durante la recepción.

4.1.2.5. Historia 5: Medir Cúbits

- **Como:** Usuario (Bob)
- **Quiero:** Medir los cúbits utilizando mis propias bases seleccionadas
- **Para:** Decodificar la información enviada por Alice.
- **Criterios de Aceptación:**
  - Las bases deben ser seleccionadas aleatoriamente por Bob.
  - Las mediciones deben ser registradas para la comparación posterior con Alice.

4.1.2.6. Historia 6: Validar Bases

- **Como:** Sistema de Comunicación Cuántica
- **Quiero:** Comparar las bases utilizadas por Alice y Bob
- **Para:** Identificar y descartar los bits en los que las bases no coinciden.
- **Criterios de Aceptación:**
  - Solo los bits medidos con bases coincidentes deben ser considerados válidos.
  - La comparación debe ser segura y no revelar información a terceros.

4.1.2.7. Historia 7: Conciliar Errores

- **Como:** Sistema de Comunicación Cuántica
- **Quiero:** Utilizar algoritmos de reconciliación de errores (como LDPC)
- **Para:** Corregir posibles errores introducidos por el ruido del canal o un espía.
- **Criterios de Aceptación:**
  - Los errores deben ser corregidos con alta precisión.
  - El algoritmo debe ser eficiente y rápido.

4.1.3. Diagrama de clases

Cómo se puede apreciar en la Figura 4.2, el diagrama de clases muestra que la mayor parte de los objetos están funcionando de forma independiente a las funciones ya que cuentan como un tipo de dato abstracto para almacenar los distintos componentes requeridos por el sistema para entablar una comunicación entre los agentes.

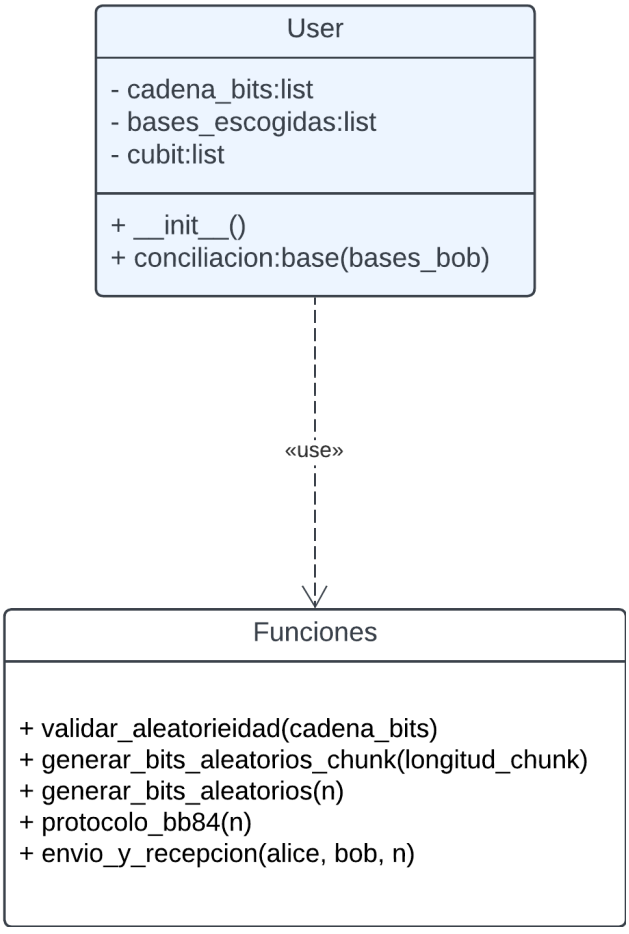

Figura 4.2: Diagrama de Clases Simulación Fuente: Autores

4.1.4. Diagrama de flujo

Para continuar con la ilustración y representación de este proyecto, es importante entender la lógica detrás del funcionamiento del sistema. En otras palabras, las posibles rutas o acciones que se pueden tomar en el prototipo. Para esto se tiene el diagrama de flujo de la Figura 4.3, donde se muestran las distintas etapas que deben cumplirse para poder llegar al fin, considerando las distinta verificaciones de los porcentajes de error y los umbrales máximos de iteración que puede tener el sistema, y en donde si alguna de estas condiciones no es posible cumplirla, se procederá a reiniciar todo el proceso desde cero.

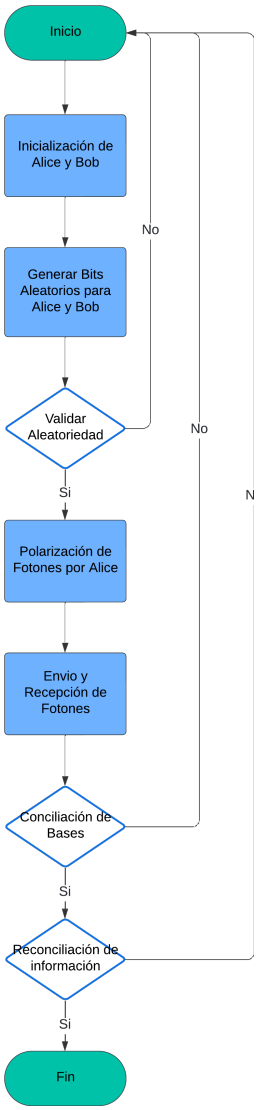

Figura 4.3: Diagrama de flujo Simulación Fuente: Autores

4.2. Implementación

En esta sección, se describen los pasos llevados a cabo para implementar el protocolo BB84 en un entorno de simulación utilizando Python. La implementación se realizó siguiendo una metodología incremental y centrada en la validación constante de cada componente desarrollado, asegurando que todos cumplan con los estándares establecidos por la normativa ETSI GS QKD 005 V1.1.1 (2010-12)[31].

El proceso de implementación comenzó con el diseño y codificación del algoritmo principal que simula el canal de comunicación cuántico según el protocolo BB84. Posteriormente, se llevaron a cabo múltiples Sprints, cada uno enfocado en diferentes aspectos del proyecto, tales como la investigación de mecanismos de validación, el diseño e implementación de algoritmos de conciliación de información, y la validación y corrección de errores en la simulación. Cada sprint permitió iterar y mejorar sobre el trabajo realizado previamente, asegurando una progresión continua y la incorporación de mejoras basadas en los resultados obtenidos y en los análisis realizados.

Cómo ayuda para el lector se elaboro la Figura 4.4, donde se podrán ver las distintas áreas en las que se desarrolló el proyecto en forma de un mapa conceptual, el cual permite aclarar la dirección en la que se elaboró el proyecto.

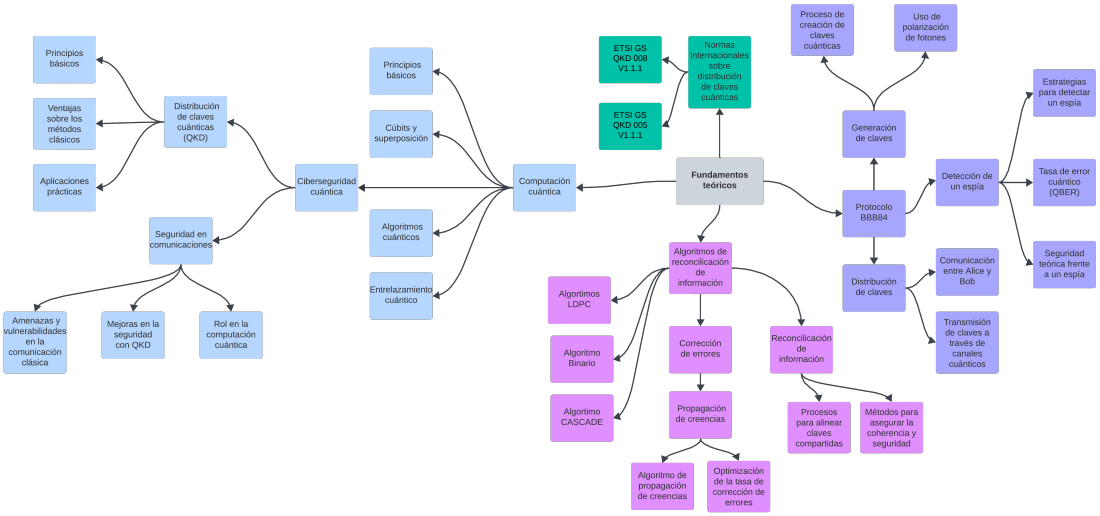

Figura 4.4: Mapa Conceptual del Proyecto
Fuente: Autores

4.2.1. Librerías

Dentro de la implementación se utilizaron las siguientes librerías:

- Random

- Hashlib
- NumPy
- SciPy
- Math
- Queue
- Time
- Threading

Las únicas que no pertenecen a las librerías estándar de Python son NumPy ya que se requirió para el manejo de las matrices, los cálculos logarítmicos. Las pruebas estadísticas usan la librería SciPy.

4.2.2. Implementación de la generación de bits pseudoaleatorios

En el anexo A, la función 'generar_bits_aleatorios_chunk' genera una cadena de bits aleatorios de una longitud específica utilizando un generador de números aleatorios criptográficamente seguro y el algoritmo de hashing SHA-256 para asegurar la calidad de los bits generados.

Finalmente, en el anexo A, la función 'generar_bits_aleatorios' gestiona la generación de una cadena de bits de la longitud que se quiere, dividiéndola en chunks para asegurar la aleatoriedad en cada segmento.

La razón por la que no se utilizó la librería 'random' de Python se debe a que dentro de la documentación de esta misma, especifica que los números generados son pseudoaleatorios y no deben ser utilizados para propósitos de seguridad [33]. Aunque dentro de esta misma documentación se recomienda utilizar la librería 'secrets' para propósitos criptográficos, por motivos de investigación y con el objetivo de realizar las funciones propias, no se tuvo en cuenta esta.

4.2.3. Cúbits, Preparación y Conciliación de Bases

Cómo parte del análisis de los componentes que se requieren para el desarrollo de la simulación y la guía sobre la implementación de este protocolo propuesto por la normativa [31] se decidió organizar la estructura de los cúbits de la siguiente manera.

Como se aprecia en el anexo A, la clase 'User' cuenta con 3 arreglos vacíos, donde se almacenan los bits generados por los agentes Alice y Bob, permitiendo también que la información pueda ser accedida y manipulada de manera eficiente. La función de conciliación de bases permite saber cuántos errores se tienen entre las bases de los agentes y dar una primera evaluación si existe un espía entre los agentes o si hay demasiado ruido en el canal, evitando una comunicación buena que

puede generar un mal paso de información más adelante.

Los agentes son creados usando la clase 'User' que se definió previamente para posteriormente usar la generación de bits aleatorios de la función 'generar_bits_aleatorios' del anexo A. Estos bits se utilizan para crear la simulación de los cúbits usando la operación XOR ya que fundamentalmente son la combinación de la cadena de bits y las bases seleccionadas para cada uno de los bits cómo se aprecia en la función 'preparar_bases' del anexo A.

En este punto se revisa si las bases que tienen ambos agentes son aceptables bajo el porcentaje de error permitido para poder entablar la comunicación entre las partes como se puede apreciar en la función 'conciliacion_bases'. Este porcentaje de error es definido por el articulo: QKD iterative information reconciliation based on LDPC codes [39].

4.2.4. Creación de Bits Aleatorios

Como se muestra en el anexo A, en la función 'generar_bits_aleatorios_chunk, la generación de bits aleatorios consiste en la creación de bytes aleatorios usando la librería 'random' de Python, de la que se le pasara un bytearray para ser encriptado y devuelto a la función que haga el llamado la cual necesitara verificar si es una cadena aleatoria o no.

4.2.5. Simulación envío y recepción de fotones polarizados

Para la simulación del envió y recepción de los cubits se tomo la decisión de hacer uso de hilos ya que la información tiene que ser enviada un cúbit a la vez, cómo lo estipula la normativa [31]. Esto permite que Bob pueda recibir el cúbit, aplicar sus bases y obtener un resultado sin que se pierda información enviada por Alice. En el anexo A, en la función 'envio_y_recepcion' se puede apreciar cómo se le aplica la operación XOR al cúbit y la base de Bob para obtener la cadena de bits.

4.2.6. Algoritmo LDPC

Basándonos en el documento [39], se explicará la reconciliación de información utilizando código de baja densidad de paridad (LDPC). Para esto, será necesario considerar un diagrama de flujo en el que se explica el proceso iterativo para la reconciliación de información. Para una comprensión más extensa de cada uno de los componentes del diagrama, se sugiere al lector tener en consideración el documento utilizado para esta sección. Sin embargo, se explicará brevemente el diagrama de flujo.

Sección de Alice

1. Grabación de datos(Data reordering): Alice registra los datos iniciales obtenidos durante el proceso de transmisión cuántica. Después de esto, usa un vector G para ordenar aleatoriamente los bits.

2. Agrupación de datos y llenado de bits (Data grouping and bit-filling): Los datos son agrupados en varias secciones de tamaño fijo y, si es necesario, se llenan con una cadena V de bits adicionales para completar los bloques.

3. Cálculo de síndrome(S): Alice utiliza la matriz de comprobación de paridad H para calcular los síndromes de cada grupo de datos X_i según la fórmula $S_i = H \cdot X_i^T$. Aquí, los síndromes ayudan a identificar posibles errores.

4. Verificación y reconciliación de Grupos de Errores: Si se detectan errores, Alice verifica si hay grupos de error que necesitan reconciliación. Si es así, el proceso se repite.

5. Descartar bits de relleno y marcados: Una vez que la reconciliación es exitosa, se descartan los bits de relleno y los bits marcados como errores.

Sección de Bob

1. Grabación de datos (Data recording): Bob, al igual que Alice, graba los datos recibidos usando el mismo vector G.

2. Agrupación de datos y llenado de bits (Data grouping and bit-filling): Bob también agrupa los datos en bloques similares.

3. Cálculo de Síndromes y Propagación de creencias: Bob calcula los síndromes $S_i' = H \cdot Y_i^T$ y aplica un algoritmo de propagación de creencias para actualizar su conocimiento sobre los errores presentes.

4. Verificación de Errores y Reconciliación: Bob verifica si la información obtenida es consistente. Si no se llega a un acuerdo tras un número máximo de iteraciones, Bob marca los errores y los informa a Alice para la reconciliación.

5. Proceso iterativo: Si se detectan errores, se informa a Alice y se repite el proceso de reconciliación hasta que todos los errores sean corregidos o se alcance un número máximo de iteraciones.

A continuación, se explicará la implementación realizada a partir del diagrama de flujo, teniendo en cuenta que cada una de las funciones mencionadas se encuentran en el anexo A.

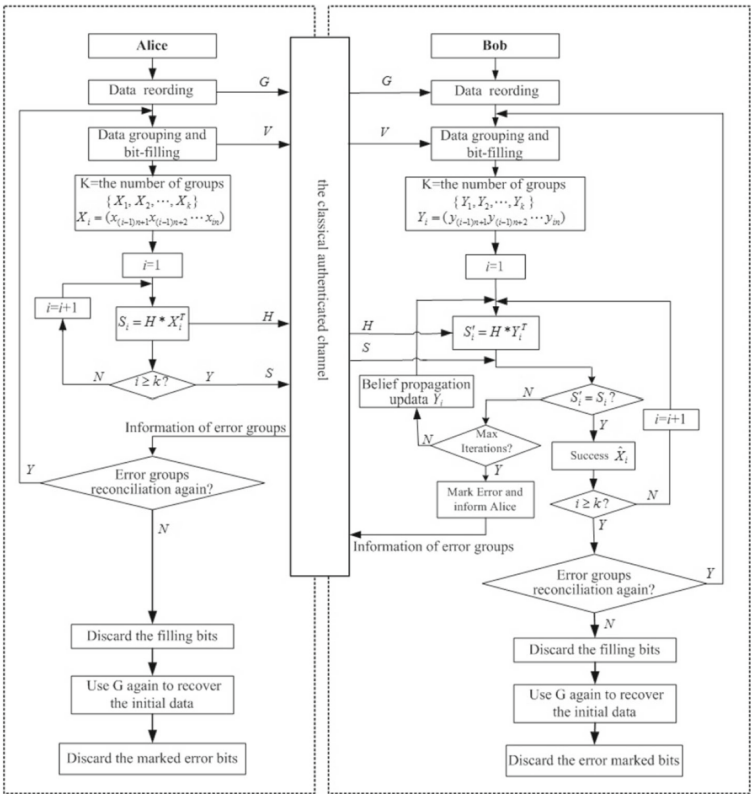

Figura 4.5: Diagrama de bloques de la conciliación iterativa de información basada en códigos LDPC. Imagen extraída de [39] pág. 1725.

1. La función 'generate_parity_check_matriz' crea una matriz de comprobación de paridad H de tamaño $(n-k)\cdot n$.

2. La función 'data_grouping_and_bit_filling' agrupa los datos y los rellena con bits adicionales para garantizar la uniformidad del tamaño de los grupos. Por efectos de simulación, se utiliza la misma cadena V aleatoria con misma probabilidad de 1s y 0s para cada uno de los grupos. También, se guardan las posiciones que corresponden a bits de relleno en cada uno de los grupos, con el objetivo de poder obtener la secuencia original al final el proceso de reconciliación.

3. La función 'reorder_data' reordena los datos de Alice y Bob según una secuencia compartida aleatoria.

4. La función 'calculate_syndrome' calcula el síndrome de una cadena de bits X utilizando una matriz H. Esta formula sale del documento [39], pág. 1722.

5. La función 'initialize_ri' inicializa los mensajes de creencia para la corrección de errores. Esta función corresponde al paso 1 de la decodificación suma-producto para la conciliación

de información del documento [39], pág. 1721. Donde tras calcular la probabilidad de que la variable aleatoria tome el valor 1 o 0 se inicializa los coeficientes de log-verosimilitud p a valores iniciales basados en lo conocido $M_{j,i} = ri$ donde $j = 1, 2, ..., n-k$ y $i = 1, 2, ..., n$.

6. La función 'calculate_extrinsic_messages' calcula los mensajes extrínsecos utilizados en la corrección de errores. Para este caso se lleva un resultado de la productoria en la variable *product*. Finalmente, para cada mensaje extrínseco para cada bit i en la verificación j se calcula utilizando el producto anterior y el síndrome s_j, como se especifica en el documento. Este proceso garantiza que la información intercambiada entre los nodos se actualice de manera consistente, mejorando la estimación de los bits a medida que se realizan más iteraciones. En esta función, sólo se tienen en cuenta los bits cuyo valor es 1 para evitar realizar operaciones innecesarias.

7. La función 'calculate_total_llr' calcula el LLR total para cada bit. Los LLR proporcionan una medida de probabilidad logarítmica de que un bit específico sea 0 o 1.

8. La función 'iterative_reconciliation' realiza el proceso iterativo de la corrección de errores entre la cadena de Alice y Bob a partir del cálculo de los síndromes de cada uno.

De acuerdo con la implementación, podemos observar que se sigue correctamente el diagrama de flujo (Ver figura 4.5) para su correcto funcionamiento. Este proceso iterativo de reconciliación se centra en la corrección de errores mediante varias iteraciones, ajustando los bits de Bob hasta que coincidan con los de Alice o hasta alcanzar un número máximo de iteraciones.

Aunque lo que se busca es corregir la mayor cantidad de errores en este proceso, no se puede asegurar que al final de la ejecución se corrijan todos los errores, sin embargo, el porcentaje de errores que se tienen al finalizar el algoritmo si es cercano a cero comparado con su inicio. Para esto, es necesario considerar que el QBER inicial no debe superar el 11 % para el protocolo BB84, ya que, podría considerarse un canal poco seguro por la posible intervención de un espía [39].

Finalmente, se revisa el orden en el que se llevan a cabo las funciones en el proceso iterativo de reconciliación de información:

1. Para cada grupo de datos, se genera una nueva matriz H de comprobación de paridad.

2. Se calculan los síndromes para las cadenas de Alice y Bob utilizando H. Si los síndromes difieren, significa que hay errores y deben corregirse.

3. En caso de existencia de errores, se realiza la iteración de corrección de errores (belief propagation):

   - Mensaje de creencia iniciales: Se inicializan los mensajes de creencia M y ri para los bits de Bob.
   - Mensajes extrínsecos: Se calculan los mensajes extrínsecos E usando M y el síndrome.

- Log-Likehood Ratios (LLR): Se calculan los LLR totales L, y se ajustan los bits de Bob basándose en estos valores.

4. Si después de las iteraciones los síndromes no coinciden, los datos se reagrupan y se rellenan nuevamente, y el proceso se repite hasta que los síndromes coincidan. El documento [39] sugiere que cuando los síndromes no son semejantes tras superar las iteraciones, ambos deben dividir sus grupos en 2 y repetir el proceso del diagrama de flujo desde el llenado de bits.

5. Cuando todos los síndromes coinciden, los grupos corregiros se almacenan tanto para Alice y Bob con el fin de eliminar los bits aleatorios de relleno utilizados y poder organizar nuevamente la cadena de bits para que ambos tengan la misma cadena de bits y poder finalmente hacer una amplificación de privacidad.

4.3. Evaluación de Resultados

Durante la ejecución de la implementación se planearon 2 pruebas para la demostración del funcionamiento de la simulación. Estas pruebas mostrarán el funcionamiento de la simulación donde ambos agentes logran entablar una comunicación sin ruido y/o un espía, y la otra con ruido y/o un espía. Estos resultados cuentan con unas limitaciones computacionales, cómo la cantidad de bits que se pueden generar de manera aleatoria para cada una de las iteraciones, la cantidad de grupos en los que se separan en la reconciliación de información y por consecuente el tamaño de la matriz de paridad que tiene dimensiones acorde al tamaño de la cadena de bits aleatorios que se generaron. Estas limitaciones mencionadas se deben al largo tiempo de ejecución de estas operaciones, la cual para tamaños de mas de 16 bits puede tomar tiempos en las horas o días, evitando el poder hacer una ejecución completa del proceso realizado. Por esta razón, los resultados se generaron con una cantidad de 16 bits aleatorios.

4.3.1. Prueba 1 - Ejecución sin ruido y/o un espía

Para iniciar esta prueba se ejecuta la clase 'User' y la función 'generar_bits_aleatorios', de la cual se obtiene el resultado presente en la figura 4.6.

```
Bits aleatorios certificados (16 bits): 0111000010011110
Resultado de las pruebas estadísticias: True
```

Figura 4.6: Resultado Generación de Bits Aleatorios y Pruebas Estadísticas
Fuente: Autores

Las pruebas estadísticas para esta cadena de bits dieron verdadero, comprobando que la cadena generada es aleatoria y su uso en entornos criptográficos es válido.
Una vez demostrado el funcionamiento de la generación de bits y las pruebas estadísticas exitosas se ejecuta la función 'preparar_bases', permitiendo a los agentes generar la información requerida

para entablar una comunicación inicial. Esto se ve reflejado en la figura 4.7.

```
print(Bob.bases_escogidas)
print(Alice.bases_escogidas)
print(Alice.cadena_bits)
print(Alice.cubit)

0111000010011110
0111000010011110
0110111000110100
[0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]
```

Figura 4.7: Resultado Preparación de Bases Alice y Bob
Fuente: Autores

Como se puede apreciar en la figura 4.7, ambos agentes resultan con la misma base, reflejado en las primeras dos cadenas de bits generados. Este caso se puede presentar en la realidad por la cantidad de opciones que pueden escoger ambos al momento de seleccionarlas. Al tener estas bases se procede a verificar si el QBER no se ha superado y dar continuación a la ejecución.

La siguiente etapa es transmitir los cúbits simulados de Alice hacia Bob por medio de las funciones 'envio_y_recepcion' y 'conciliacion_bases'. De las cuales se generan los resultados de las figuras 4.8 y 4.9 respectivamente.

```
Base Alice: 0111000010011110
Base Bob: 0111000010011110
Cadena Alice: 0110111000110100
Cadena Bob: [0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0]
```

Figura 4.8: Resultado Simulación envío y recepción de fotones polarizados
Fuente: Autores

```
cnt_errores, bases_correctas_bob = Alice.conciliacion_bases(Bob.bases_escogidas)
print(cnt_errores > 0.11*len(Alice.bases_escogidas))
if cnt_errores > 0.11*len(Alice.bases_escogidas):
  print("Demasiado Ruido")
else:
  print("Buenas bases")

False
Buenas bases
```

Figura 4.9: Resultado Validación Conciliación de Bases Alice y Bob
Fuente: Autores

Aquí se pueden apreciar claramente las cadenas de bits que representan las bases de los agentes junto con la cadena de Alice y la cadena resultante de Bob tras la operación de la función 'conciliacion_bases' de la clase 'User'.

Es necesario aclarar que para este caso de prueba no se aplica ruido sobre la cadena final después del proceso de conciliación de bases. Finalmente, se aplica el algoritmo LDPC explicado en la subsección 4.2.6. Tras este resultado podemos obtener los siguientes resultados:

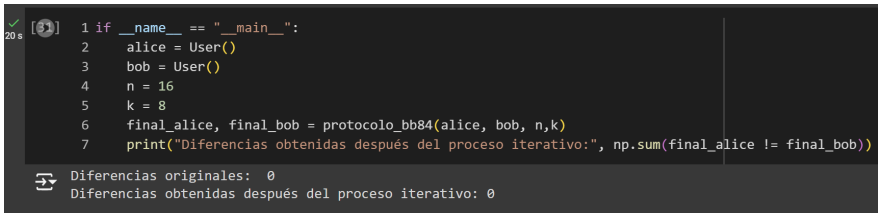

Figura 4.10: Resultado obtenido tras la conciliación de bases
Fuente: Autores

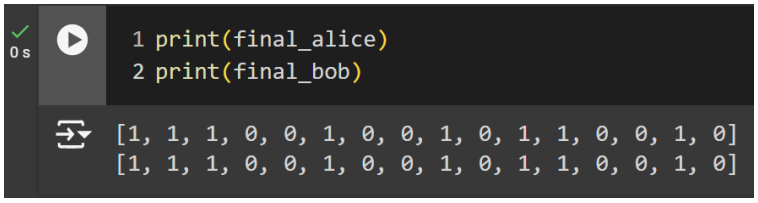

Figura 4.11: Resultado obtenido tras el proceso iterativo del algoritmo LDPC
Fuente: Autores

Tras estas pruebas, se puede observar en la imagen 4.10 que tras la conciliación de bases, la cadena de Alice y Bob son idénticas. Por esto, se puede apreciar en la imagen 4.11 que, tras el proceso iterativo del algoritmo LDPC, la cadena final de Alice y la cadena final de Bob son idénticas también.

Es evidente que la cadena final no corresponde a la inicial, esto se debe a que hace falta reordenar los bits en relación del vector G.

4.3.2. Prueba 2 - Ejecución con ruido y/o un espía

Para iniciar esta prueba se definió que se utilizaría un porcentaje de ruido del 60 % en la cadena de Bob para el proceso del algoritmo LDPC donde se hace la reconciliación de información, cómo se podrá apreciar a continuación.

Una vez demostrado el funcionamiento de la generación de bits y las pruebas estadísticas exitosas demostrado en la figura 4.6 se ejecuta la función 'preparar_bases', permitiendo a los agentes

generar la información requerida para entablar una comunicación inicial. Esto se ve reflejado en las figura 4.12 y 4.13 respectivamente.

```
print(Bob.bases_escogidas)
print(Alice.bases_escogidas)
print(Alice.cadena_bits)
print(Alice.cubit)

0111000010011110
0111000010011110
1110001110110000
[1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0]
```

Figura 4.12: Resultado Preparación de Bases Alice y Bob
Fuente: Autores

```
cnt_errores, bases_correctas_bob = Alice.conciliacion_bases(Bob.bases_escogidas)
print(cnt_errores > 0.11*len(Alice.bases_escogidas))
if cnt_errores > 0.11*len(Alice.bases_escogidas):
  print("Demasiado Ruido")
else:
  print("Buenas bases")

False
Buenas bases
```

Figura 4.13: Resultado Validación Conciliación de Bases Alice y Bob
Fuente: Autores

Cómo se puede apreciar en la figura 4.12, ambos agentes resultan con la misma base, reflejado en las primeras dos cadenas de bits generados. Este caso se puede presentar en la realidad por la cantidad de opciones que pueden escoger ambos al momento de seleccionarlas. Al tener estas bases se procede a verificar si el QBER no se ha superado y dar continuación a la ejecución.

La siguiente etapa es transmitir los cúbits simulados de Alice hacia Bob por medio de la función 'envio_y_recepcion'. De la cual se genera el resultado de la figura 4.14.

```
Base Alice: 0111000010011110
Base Bob: 0111000010011110
Cadena Alice: 1110001110110000
Cadena Bob: [1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0]
```

Figura 4.14: Resultado Simulación envío y recepción de fotones polarizados
Fuente: Autores

Tras estas pruebas, se puede observar en la imagen 4.15 que tras la conciliación de bases, la cadena de Alice y Bob no son idénticas. Por esto, se puede apreciar en la imagen 4.16, que tras el proceso iterativo del algoritmo LDPC, la cadena final de Alice y la cadena final de Bob son idénticas, lo que significa que se corrigieron la totalidad de sus errores a pesar de que la cantidad de errores iniciales supera el 11 %.

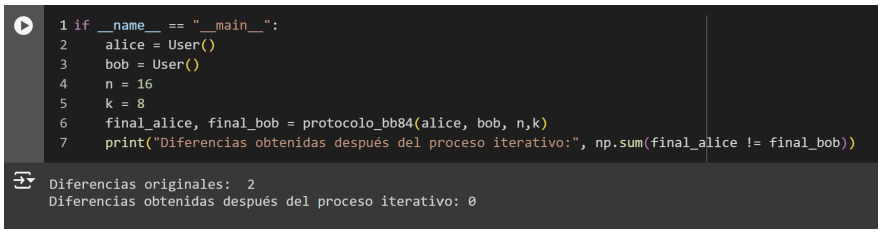

Figura 4.15: Resultado obtenido tras la conciliación de bases
Fuente: Autores

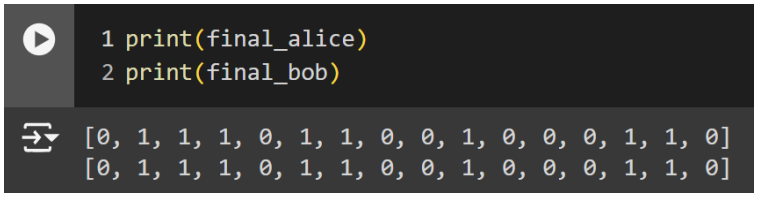

Figura 4.16: Resultado obtenido tras el proceso iterativo del algoritmo LDPC
Fuente: Autores

Es evidente que la cadena final no corresponde a la inicial, esto se debe a que hace falta reordenar los bits en relación del vector G.

4.4. Validación de seguridad de la implementación del protocolo

Para validar de los parámetros de seguridad de la implementación del protocolo BB84, siguiendo las especificaciones establecidas en el estándar ETSI GS QKD 005 V1.1.1.

4.4.1. Parámetros evaluados

- Tasa de error cuántico (QBER): Para un sistema seguro, el QBER debe mantenerse por debajo de un umbral específico, generalmente alrededor del 11 %. En la implementación, se ha calculado el QBER para diferentes conjuntos de datos simulados, asegurando que los errores introducidos por Eve se mantengan dentro de los límites aceptables (véase 4.9).

- Porcentaje de errores tras el LDPC: Para que un protocolo de distribución de claves pueda considerarse óptimo, entre tantos aspectos a considerar, uno de los más importantes es que la clave final entre Alice y Bob debe tener la menor cantidad de errores posibles. Las pruebas realizadas mostraron que para una cadena tamaño $n = 16$ y una matriz de paridad $M_{(n-k)\cdot n}$ con $k = 8$, donde la cadena de Bob presentaba 2 errores en relación con la cadena de Alice (Véase la figura 4.15), tras el proceso iterativo del LDPC se obtuvo que se corrigieron 2 errores, quedando con cadenas idénticas finalmente.

4.4.2. Validación según ETSI GS QKD 005 V1.1.1 (2010-12)

El estándar establece una serie de requisitos y recomendaciones para la implementación de sistemas de distribución de claves cuánticas que luego se extiende al documento [31]. A continuación, se detallarán los puntos con los que cumple la implementación:

- Generación de claves cuánticas: La implementación sigue el protocolo BB84 para la generación de claves cuánticas simuladas, asegurando que los bits sean aleatorios y no predecibles.
- Detección de interferencias y espionaje: El QBER son indicadores de la capacidad del sistema para detectar intentos de espionaje lo que se alinea con las directrices del estándar. Si el QBER supera el umbral, no se debería continuar con la comunicación, pues se podría suponer la presencia de un espía o que el canal posee mucho ruido.
- Elección del algoritmo de reconciliación: Se ha investigado cada uno de los algoritmos existentes para la reconciliación de información y se ha escogido un algoritmo LDPC, ya que este tiene un mayor índice de seguridad en comparación con los otros.
- Reconciliación de la información: Se ha implementado un algoritmo LDPC para la reconciliación de información, garantizando que los errores se corrigen eficazmente y que la clave final logra ser lo más idéntica posible entre Alice y Bob, conforme a las recomendaciones del estándar.
- Seguridad de la clave distribuida: La alta correlación entre los bits de Alice y Bob valida que la clave distribuida es segura y coincide entre ambas partes, cumpliendo con los requisitos del estándar para la integridad de la clave. A partir de esto, se puede realizar la amplificación de información en la que Alice y Bob puedan usar una función criptográfica hash para guardar la clave privada.

4.4.3. Validación según ETSI GS QKD 008 V1.1.1 (2010-12)

El estándar para la especificación de seguridad del modulo especifica los distintos acercamientos que se deben tener en consideración para considerar la implementación realizada para la distribución de llaves cuánticas (QKD) segura y valida para ser integrada a distintos sistemas de seguridad. Estas especificaciones tienen cómo enfoque mostrar los distintos componentes, desde el canal cuántico, el canal clásico, y los tipos de componentes físicos que debe ser utilizado en las distintas etapas [31].

La normativa también habla sobre cuál debe ser el debido proceso para una implementación del protocolo BB84, por lo que nos centraremos en comparar este proceso con la implementación que se realizó, teniendo en consideración las restricciones computacionales y físicas que se definieron para el desarrollo de este proyecto.

- Envío y recepción de fotones: La normativa dicta que se debe iniciar el ciclo de envió y recepción de fotones al mismo tiempo para permitirle a Alice enviar sus fotones uno a la vez y asegurarse de que Bob pueda leerlos y decodificarlos usando sus bases. A su vez estipula de que se debe usar la generación de bits aleatorios para generar las bases de Alice y Bob junto con la cadena de información que va a transmitir Alice en el cúbit, el almacenamiento de esta información para ambos agentes y procedimiento para la lectura de Bob. Esto se puede ver reflejado en la clase 'User', y las funciones 'preparar_bases', 'envio_y_recepcion' y 'generacion_bits_aleatorios'.
- Reconciliación de información: La normativa dicta la existencia de la separación por grupos de la información transmitida por el canal tradicional, el método de corrección de errores y el procedimiento aprobado para la amplificación de la privacidad entre los agentes. Esto se ve reflejado en la subsección 4.2.6.

Capítulo 5

Conclusiones y Trabajos Futuros

5.1. Conclusiones

Este proyecto está enfocado en profundizar sobre la ciberseguridad que se puede tener en un futuro próximo con la llegada de la computación cuántica investigando a cerca del protocolo BB84 en el que se han basado muchos autores para poder establecer diferentes protocolos de distribución de llaves cuánticas seguras. La implementación del protocolo BB84 se revisó minuciosamente en relación con los parámetros de seguridad especificados por el estándar ETSI GS QKD 005 V1.1.1. Se realizaron adaptaciones necesarias para asegurar el cumplimiento con estos parámetros, lo cual incluyó ajustes en el proceso de generación y distribución de llaves cuánticas.

Se llevó a cabo una verificación exhaustiva de los elementos de seguridad como parte de la implementación del protocolo. Esto incluyó la evaluación de la resistencia frente a posibles ataques de un espía y la robustez de los mecanismos de detección de eavesdrop.

Se utilizó el algoritmo LDPC [39] con técnicas de propagación de creencias para la reconciliación de información, lo cuál mejoró significativamente la tasa de corrección de errores y la fiabilidad de la transmisión de datos. Este proceso fue clave para asegurar que las claves compartidas entre las partes fueran coherentes y seguras.

Se realizaron diferentes pruebas, tanto en condiciones ideales (como lo puede ser un canal sin ruido) como en presencia de ruido y posibles ataques. Los resultados demostraron que el protocolo BB84, apoyado por un algoritmo LDPC, mantiene una alta robustez frente a las posibles interferencias en un canal clásico y ataques.

5.2. Trabajos futuros

Con el enfoque dado a este proyecto se considera que puede extenderse a una implementación más cercana a la realidad usando librerías especializadas en los procesos de generación de bits aleatorios implementadas por distintas librerías cómo aquellas desarrolladas por la empresa Qiskit, quienes son lideres en el campo de la computación cuántica y el manejo de simulaciones en entornos controlados para computadores tradicionales junto con integraciones en computadores cuánticos. La modificación del protocolo utilizado es una de las extensiones que se le pueden dar a este proyecto,

ya que a la fecha existen múltiples protocolos que buscan hacer la distribución de llaves cuánticas de una manera más segura y efectiva, cómo lo es el protocolo E91 y aquellas implementaciones derivadas de estos protocolos mencionados.

Otras extensiones que se le pueden dar a este proyecto son aquellas en las que se pueden involucrar dispositivos físicos como un modulador de fotones, cables de fibra óptica para el envío de mensajes junto con interfaces desarrolladas que permitan mostrar el hilo de procesos y etapas en las que se encuentran los computadores involucrados en entablar la comunicación usando este protocolo.

Adicionalmente, considerando la consigna anterior, se podría considerar realizar un ambiente virtual educativo que utilice el código implementado para el protocolo BB84 y el algoritmo LDPC. Este ambiente virtual serviría como una herramienta didáctica para estudiantes e investigadores interesados en la criptografía cuántica y la reconciliación de información. Este recurso educativo no solo facilitaría la compresión teórica y práctica de los conceptos abordados, sino que también fomentaría el aprendizaje interactivo.

Capítulo 6

Anexo A: Código del Protocolo BB84

En este anexo se presenta el código Python utilizado para simular el protocolo BB84.

Es posible encontrar la implementación en el siguiente enlace: https://colab.research.google.com/drive/12rje5gA2LvN6s2IC4qNdKLP9PX-59LN-.

El código se encuentra bajo el estándar de codificación Google Style Guides [44] y está dividido en varias funciones que realizan las siguientes tareas:

```
#Importacion de librerias

import random
import hashlib
import numpy as np
from scipy.stats import chi2
from collections import Counter
import math
import queue
import threading
import time
```

```
#Creacion de las clases que representaran a Alice, Bob y Eva.

"""
Por propositos de simulacion, se contemplaran los fotones de la
siguiente manera:

*La base '+' se representara como un 0 y la base 'X' se
representara como un 1.

*   Asi mismo, de acuerdo al bit aleatorio y la base escogida,
el foton polarizado de '+' tal que '|=1' o '-=0'. De igual forma,
para la base escogida 'X' o '-45/+45', el resultado del foton
```

```
polarizado podria ser '\=1' o '/=0'.


Ejemplo: Para un bit aleatorio igual a 0 y la base escogida aleatoria
'+', el foton polarizado resultante seria '-' que es igual a '0'.
"""

class User():
    def __init__(self):
      self.cadena_bits = []
      self.bases_escogidas = []
      self.cubit = []

    def conciliacion_bases(self, bases_bob):
      """Conciliacion de bases entre Alice y Bob."""
      errores = 0
      bases_correctas = []
      for i in range(len(bases_bob)):
        if self.bases_escogidas[i] != bases_bob[i]:
          errores += 1
        if self.bases_escogidas[i] == bases_bob[i]:
          bases_correctas.append(i)
      return errores, bases_correctas
```

```
#Creacion de bits aleatorios

def validar_aleatorieidad(cadena_bits):
    """
    Genera una respuesta de True o False si la cadena de bits cumple
    con las pruebas estadisticas utilizadas proporcionadas por el
    NIST.

    Args:
      cadena_bits: La cadena de bits generada aleatoriamente.

    Returns:
      True si la cadena supera las pruebas estadisticas.
      False si la cadena no supera alguna prueba estadistica.
    """

    # Prueba de frecuencia monobit
```

```
    num_1s = cadena_bits.count('1')
    num_0s = cadena_bits.count('0')
    prueba_monobit =  abs(num_1s - num_0s) / len(cadena_bits)

    #DFT
    secuencia_numeros = np.array([int(bit) for bit in cadena_bits])
    frecuencia_secuencia = np.fft.fft(secuencia_numeros)
    magnitudes = np.abs(frecuencia_secuencia)

    # Usamos el test de Chi-cuadrado para verificar si las magnitudes
    siguen una distribucion uniforme
    bins = len(magnitudes) // 2 #Solo analizamos la mitad positiva
    #del espectro
    hist, bin_edges = np.histogram(magnitudes[:bins], bins=bins,
    density=True)
    chi_squared = np.sum((hist - np.mean(hist))**2 / np.mean(hist))
    p_value = chi2.sf(chi_squared, bins - 1)

    # Entropia
    frecuencias = Counter(cadena_bits)
    entropia = -sum((frecuencia / len(cadena_bits)) *
    math.log2(frecuencia / len(cadena_bits))
                for frecuencia in frecuencias.values())

    return prueba_monobit < 0.01 and entropia >= 0.95 and p_value >
    0.05

def generar_bits_aleatorios_chunk(longitud_chunk):
    """
    Genera una cadena de bits aleatorios de una longitud especifica.

    Args:
      longitud_chunk: La longitud deseada de la cadena de bits
      aleatorios.

    Returns:
      Una cadena de bits aleatorios.
    """

    bytes_aleatorios = random.getrandbits(longitud_chunk // 8)
    hasher = hashlib.sha256()
```

```
    bytearray_aleatorios = bytearray(bytes_aleatorios)
    hasher.update(bytearray_aleatorios)
    digesto = hasher.digest()
    bits_aleatorios = "".join(f"{bit:08b}" for bit in
    bytearray(digesto))
    return bits_aleatorios[:longitud_chunk]

def generar_bits_aleatorios(longitud):
    """
    Genera una cadena de bits aleatorios con una longitud especifica.

    Args:
      longitud: La longitud deseada de la cadena de bits
      aleatorios.

    Returns:
      Una cadena de bits aleatorios.
    """

    if longitud <= 0:
        raise ValueError("La longitud debe ser un numero entero
        positivo")

    bits_aleatorios = []
    while longitud > 0:
        # Generar una cadena de bits aleatorios de 388 bits como
        maximo.
        chunk_longitud = min(longitud, 200)
        chunk_bits = generar_bits_aleatorios_chunk(chunk_longitud)
        bits_aleatorios.extend(chunk_bits)
        longitud -= chunk_longitud

    return "".join(bits_aleatorios)
```

```
#Preparacion de bases Alice y Bob

"""
Para esta simulacion, generaremos cadenas de numeros aleatorios tal
que cada numero aleatorio $\in$ {0,1}.
"""
```

```
def preparar_bases(alice, bob, n):
    """
    Simulacion de la preparacion de la cadena de Alice y las bases
    escogidas.

    Args:
        alice (User): El objeto representando a Alice.
        bob (User): El objeto representando a Bob.
        n (int): El n mero de bits a generar.

    Returns:
        tuple: Una tupla con Alice y Bob tras haber escogido sus bases.
    """
    alice.cadena_bits = generar_bits_aleatorios(n)
    while validar_aleatorieidad(alice.cadena_bits) != True:
      alice.cadena_bits = generar_bits_aleatorios(n)

    bob.bases_escogidas = generar_bits_aleatorios(n)
    while validar_aleatorieidad(bob.bases_escogidas) != True:
        bob.bases_escogidas = generar_bits_aleatorios(n)

    alice.bases_escogidas = generar_bits_aleatorios(n)
    while validar_aleatorieidad(alice.bases_escogidas) != True:
        alice.bases_escogidas = generar_bits_aleatorios(n)

    #Simulacion del proceso de polarizaci n de fotones de Alice.
    for i in range(n):
      alice.cubit.append(int(alice.cadena_bits[i]) ^
      int(alice.bases_escogidas[i]))
    return alice, bob
```

```
#Simulacion envio y recepcion de fotones polarizados

def envio_y_recepcion(alice, bob, n):
    """
    Simulacion del envio y recepcion de fotones polarizados.

    Args:
        alice (User): El objeto representando a Alice.
        bob (User): El objeto representando a Bob.
        n (int): El numero de bits a generar.
```

```
    Returns:
        tuple: Una tupla de Alice y Bob despues de haber recibido
        los fotones polarizados.
    """
    # Crea una cola
    q = queue.Queue()

    def emisor():
        for i in range(n):
            q.put((alice.cubit[i],i))
            time.sleep(1)

    def receptor():
        while True:
            cubit, i = q.get()
            if cubit is None:  # Senal para terminar
                break
            bob.cadena_bits.append(cubit^int(bob.bases_escogidas[i]))
            q.task_done()

    # Crea e inicia los hilos de productor y consumidor
    producer_thread = threading.Thread(target=emisor)
    consumer_thread = threading.Thread(target=receptor)

    producer_thread.start()
    consumer_thread.start()

    # Espera a que el productor termine
    producer_thread.join()

    # Envia una senal para que el consumidor termine
    q.put((None, None))

    # Espera a que el consumidor termine
    consumer_thread.join()

    return alice, bob
```

```
#Conciliacion de bases

def conciliacion_bases(alice, bob):
```

```
    """
    Conciliacion de bases entre Alice y Bob.

    Args:
        alice (User): El objeto representando a Alice.
        bob (User): El objeto representando a Bob.

    Returns:
        int: El porcentaje de numero de errores cometidos.
    """
    cnt_errores, bases_correctas_bob =
    alice.conciliacion_bases(bob.bases_escogidas)
    cnt_errores > 0.11*len(alice.bases_escogidas),
    bases_correctas_bob
```

```
# LDPC

"""
  Informacion relevante a cerca de las caracteristicas de una matriz
  de paridad.

  k = 972 # Longitud del tamano inicial de la cadena de bits enviada.

  C = 2 * k #La norma IEEE 802.11n define que para una tasa de codigo
  de 1/2 la longitud debe ser el doble.

  R = 1 / 2 # La tasa del codigo que se desea llenar con bits de
  paridad.

  r = C * (1 - R) # r es el numero de bits de paridad.
"""

def generate_parity_check_matrix(n, k):
    """
    Genera la matriz de verificacion de paridad H para un c digo LDPC.

    Args:
        n (int): Numero total de bits.
        k (int): Numero de bits de informacion.

    Returns:
```

```
        np.ndarray: Matriz de verificacion de paridad H.
    """
    r = n - k  # Numero de filas en H
    H = np.zeros((r, n), dtype=int)
    # Llenar H con una matriz de identidad en las primeras columnas
    for i in range(r):
        H[i, i] = 1
    # Llenar el resto de H con una matriz generada aleatoriamente
    for i in range(r):
        for j in range(r, n):
            H[i, j] = np.random.randint(0, 2)
    return H

def reorder_data(data, sequence):
    """
    Reordena los datos segun una secuencia dada.

    Args:
        data (np.ndarray): Los datos a reordenar.
        sequence (list): La secuencia de reordenamiento.

    Returns:
        np.ndarray: Datos reordenados.
    """
    return data[sequence]

def data_grouping_and_bit_filling(data, k, mu, n, V):
    """
    Agrupa datos y llena bits aleatorios.

    Args:
        data (np.ndarray): Datos a agrupar.
        k (int): Numero de grupos.
        mu (int): Tamano de cada grupo.
        n (int): Numero total de bits.
        V (np.ndarray): Bits de relleno.

    Returns:
        tuple: Grupos llenos de datos y grupos V.
    """
    # Creamos los k grupos.
```

```
    grouped_data = [data[i:i+mu] for i in range(0, len(data), mu)]
    #Vamos a rellenar para cada grupo n-mu bits aleatorios
    filling_bits = n-mu
    if len(grouped_data[-1]) < mu:
            tmp = generar_bits_aleatorios(mu-len(grouped_data[-1]))
            while validar_aleatorieidad(tmp) != True:
                tmp = generar_bits_aleatorios(mu-len(grouped_data[-1]))
            grouped_data[-1] = grouped_data[-1].tolist()
            grouped_data[-1].extend(tmp)
    grouped_data[-1] = np.array( list(map(int,grouped_data[-1])) )

    group_filled, V_groups = [], []
    for group in grouped_data:
        insertion_index = random.sample([i for i in range(n)],
        filling_bits)
        insertion_index.sort()
        V_groups.append(insertion_index)
        j = 0
        for i in range(n):
          if i in insertion_index:
            group = np.insert(group, i, V[j])
            j += 1
        group_filled.append(group)
    return group_filled, V_groups

def calculate_syndrome(H, X):
    """Calcula el sindrome de un vector de datos usando la matriz H.

    Args:
        H (np.ndarray): Matriz de verificacion de paridad.
        X (np.ndarray): Vector de datos.

    Returns:
        np.ndarray: Sindrome calculado.
    """
    return np.mod(np.dot(H, X.T), 2)

def prepare_data(n, k, alice, bob):
    """
    Prepara los datos para la simulacion.
```

```
    Args:
        n (int): Numero total de bits.
        k (int): Numero de bits de informaci n.
        alice (User): El objeto representando a Alice.
        bob (User): El objeto representando a Bob.

    Returns:
        tuple: Datos de Alice, Bob, sus respectivos bits de relleno y
        la cantidad de grupos escogidos.
    """
    alice_data = alice.cadena_bits
    bob_data = bob.cadena_bits

    # Introducir errores en los datos de Bob
    error_rate = 0.3
    error_positions = np.random.choice(range(k), int(k * error_rate),
    replace=False)
    bob_data[error_positions] = 1 - bob_data[error_positions]

    #Se busca observar la cantidad de bits diferentes antes de
    realizar el proceso iterativo.
    print("Diferencias originales: ", np.sum(bob_data != alice_data))

    # Creacion del vector G
    shared_sequence = np.random.permutation(len(alice_data))

    # Reordenamiento de datos
    alice_reordered = reorder_data(alice_data, shared_sequence)
    bob_reordered = reorder_data(bob_data, shared_sequence)

    # Agrupamiento y llenado de bits
    alice_groups = np.random.randint(2,4)
    bob_groups = alice_groups # Se establecen Bob y Alice con la
    misma cantidad de grupos por efectos de implementacion

    filling_bits = (n-(math.ceil(n/bob_groups)))
    V = np.random.choice([0,1], size=filling_bits)
    alice_grouped, V_Alice =
    data_grouping_and_bit_filling(alice_reordered, alice_groups,
    math.ceil(n/alice_groups), n, V)
    bob_grouped, V_Bob = data_grouping_and_bit_filling(bob_reordered,
```

```
    bob_groups, math.ceil(n/bob_groups), n, V)
    return alice_grouped, bob_grouped, V_Alice, V_Bob, alice_groups

def initialize_ri(bob_message, p_error, n, k):
    """Inicializa el vector de informacion inicial ri.

    Args:
        bob_message (np.ndarray): Mensaje de Bob.
        p_error (float): Probabilidad de error.
        n (int): Numero total de bits.
        k (int): Numero de bits de informaci n.

    Returns:
        tuple: Matriz M y vector ri inicializados.
    """
    ri = np.zeros(len(bob_message))
    M = np.zeros((n-k, n))
    for i, bit in enumerate(bob_message):
        if bit == 1:
            ri[i] = np.log(p_error / (1 - p_error))
        else:
            ri[i] = np.log((1 - p_error) / p_error)
    for j in range(n-k):
        for i in range(n):
            M[j,i] = ri[i]
    return M, ri

def calculate_extrinsic_messages(H, M, syndrome):
    """Calcula los mensajes extrinsecos para el algoritmo de
    decodificacion.

    Args:
        H (np.ndarray): Matriz de verificacion de paridad.
        M (np.ndarray): Matriz de mensajes.
        syndrome (np.ndarray): Vector de sindrome.

    Returns:
        np.ndarray: Matriz de mensajes extr nsecos.
    """
    rows, cols = H.shape
    E = np.zeros_like(H, dtype=float)
```

```
    for j in range(rows):
        for i in range(cols):
            if H[j, i] == 1:
                product = 1
                for i_prime in range(cols):
                    if i_prime != i and H[j, i_prime] == 1:
                        product *= np.tanh(M[j, i_prime] / 2)
                E[j, i] = np.log((1 + (1 - (2 * syndrome[j])) *
                product) / (1 - (1 - (2 * syndrome[j])) * product))
    return E

def calculate_total_llr(H, E, ri):
    """Calcula la suma de las razones de verosimilitud (LLR).

    Args:
        H (np.ndarray): Matriz de verificacion de paridad.
        E (np.ndarray): Matriz de mensajes extrinsecos.
        ri (np.ndarray): Vector de informacion inicial.

    Returns:
        np.ndarray: Vector de LLR totales.
    """
    rows, cols = H.shape
    L = np.zeros(cols, dtype=float)
    for i in range(cols):
        for j in range(rows):
            if H[j, i] == 1:
                L[i] += E[j, i] + ri[i]
    return L

def iterative_reconciliation(n, k, X, Y, cnt_groups, V_alice, V_bob,
max_iterations=100):
    """Realiza la reconciliacion iterativa para corregir errores.

    Args:
        n (int): Numero total de bits.
        k (int): Numero de bits de informaci n.
        X (np.ndarray): Datos de Alice.
        Y (np.ndarray): Datos de Bob.
        cnt_groups (int): Numero de grupos.
        V_alice (list): Bits de relleno de Alice.
```

```
        V_bob (list): Bits de relleno de Bob.
        max_iterations (int, optional): Maximo numero de iteraciones.
        Defaults to 100.

    Returns:
        tuple: Datos finales de Alice, Bob y sus respectivos bits de
        relleno.
    """
    final_alice, final_bob, final_V_alice, final_V_bob = [], [], [],
    []
    while len(X) > 0:
        H = generate_parity_check_matrix(n, k)
        Si_prima = calculate_syndrome(H, Y[-1])
        Si = calculate_syndrome(H, X[-1])
        iterations = 1
        while np.sum(Si_prima != Si) != 0 and iterations <
        max_iterations:
            M, ri = initialize_ri(Y[-1], 0.1, n, k)
            E = calculate_extrinsic_messages(H, M, Si)
            L = calculate_total_llr(H, E, ri)
            Z = [ 1 if L[j] <= 0 else 0 for j in range(len(L)) ]
            Y[-1] = np.array(Z.copy())
            Si_prima = calculate_syndrome(H, Y[-1])
            iterations += 1
        if np.sum( Si_prima != Si ) != 0:
            g_alice, g_bob = [], []
            for j in range(len(X[-1])):
                if not j in V_alice[-1]:
                    g_alice.append(X[-1][j])
                if not j in V_bob[-1]:
                    g_bob.append(Y[-1][j])
            g_alice, g_bob = np.array(g_alice), np.array(g_bob)
            filling_bits = (len(X[-1])-(math.ceil(len(g_alice)/2)))
            V = np.random.choice([0,1], size=filling_bits)
            alice_new_grouped, V_new_Alice =
            data_grouping_and_bit_filling(g_alice, 2,
            math.ceil(len(g_alice)/2), len(X[-1]), V)
            bob_new_grouped, V_new_Bob =
            data_grouping_and_bit_filling(g_bob, 2,
            math.ceil(len(g_bob)/2), len(Y[-1]), V)
            X = np.delete(X, len(X)-1, axis=0)
```

```
            Y = np.delete(Y, len(Y)-1, axis=0)
            X = np.append(X, alice_new_grouped, axis=0)
            Y = np.append(Y, bob_new_grouped, axis=0)
            del V_alice[-1]
            del V_bob[-1]
            for x in range(len(V_new_Alice)):
                V_alice.append(V_new_Alice[x])
                V_bob.append(V_new_Bob[x])
        else:
            final_alice.append(X[-1])
            final_bob.append(Y[-1])
            final_V_alice.append(V_alice[-1])
            final_V_bob.append(V_bob[-1])
            X = np.delete(X, len(X)-1, axis=0)
            Y = np.delete(Y, len(Y)-1, axis=0)
            del V_alice[-1]
            del V_bob[-1]
    return final_alice, final_bob, final_V_alice, final_V_bob
```

```
# Eliminar los bits de relleno

def eliminar_bits_relleno(alice_grouped, bob_grouped, V_alice, V_bob):
    """
    Elimina los bits de relleno de los grupos de datos.

    Args:
        alice_grouped (list): Grupos de datos de Alice.
        bob_grouped (list): Grupos de datos de Bob.
        V_alice (list): Bits de relleno de Alice.
        V_bob (list): Bits de relleno de Bob.

    Returns:
        tuple: Datos finales de Alice y Bob sin bits de relleno.
    """
    final_alice, final_bob = [], []
    for i in range(len(alice_grouped)):
        for j in range(len(alice_grouped[i])):
            if not j in V_alice[i]:
              final_alice.append(alice_grouped[i][j])
            if not j in V_bob[i]:
              final_bob.append(bob_grouped[i][j])
```

```
    return final_alice, final_bob
```

```
# Protocolo BB84

def protocolo_bb84(alice, bob,n, k):
    """
    Implementacion del protocolo BB84 para la distribucion de claves
    cuanticas.

    Args:
        alice (User): El objeto representando a Alice.
        bob (User): El objeto representando a Bob.
        n (int): El numero de bits a generar.
        k (int): El numero de bits de informacion.
    Returns:
        tuple: Una tupla con las claves generadas por Alice y Bob.
    """
    alice, bob = preparar_bases(alice, bob, n)
    alice, bob = envio_y_recepcion(alice, bob, n)
    flag_umbral, bases_correctas_bob = conciliacion_bases(alice, bob)
    alice.cadena_bits = np.array(list(map(int, alice.cadena_bits)))
    bob.cadena_bits = np.array(list(map(int,bob.cadena_bits)))
    alice.cadena_bits = alice.cadena_bits[bases_correctas_bob]
    bob.cadena_bits = bob.cadena_bits[bases_correctas_bob]
    if not flag_umbral:
        "No supera el supera el umbral establecido para el protocolo
        bb84."
        alice_grouped, bob_grouped, V_Alice, V_Bob, alice_groups =
        prepare_data(n, k, alice, bob)
        alice_grouped, bob_grouped, V_alice, V_bob =
        iterative_reconciliation(n, k, alice_grouped, bob_grouped,
        alice_groups, V_Alice, V_Bob)
        final_alice, final_bob = eliminar_bits_relleno(alice_grouped,
        bob_grouped, V_alice, V_bob)
        return final_alice, final_bob
    else:
        print("Supera el umbral establecido para el protocolo bb84,
        se termina la comunicacion.")
    return alice, bob
```

Bibliografía

[1] R. L. Rivest, A. Shamir, and L. Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21, 2 (Feb. 1978), 120–126. https://doi.org/10.1145/359340.359342

[2] "¿Qué es la ciberseguridad?," IBM. [En línea]. Disponible: https://www.ibm.com/es-es/topics/cybersecurity. [Consultado: 24-Sep-2023]

[3] A. M. Peñas Díaz, "¿Qué es la criptografía?," Buran, no. 6, pp. 6–7, 1995.

[4] "¿Qué es quantum computing?," IBM. [En línea]. Disponible: https://www.ibm.com/mx-es/topics/quantum-computing. [Consultado: 24-Sep-2023]

[5] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge: Cambridge University Press, 2010.

[6] "What is a qubit?: Microsoft Azure," What is a Qubit? | Microsoft Azure. [En línea]. Disponible: https://azure.microsoft.com/en-gb/resources/cloud-computing-dictionary/what-is-a-qubit. [Consultado: 25-Sep-2023]

[7] "What is superposition and why is it important?," Caltech Science Exchange. [En línea]. Disponible: https://scienceexchange.caltech.edu/topics/quantum-science-explained/quantum-superposition. [Consultado: 25-Sep-2023]

[8] D. Garisto, "What is quantum entanglement?," IEEE Spectrum, 22-Nov-2022. [En línea]. Disponible: https://spectrum.ieee.org/what-is-quantum-entanglement. [Consultado: 25-Sep-2023]

[9] C. Bea Monreal, "Criptografía de clave simétrica y de clave asimétrica", Universitat Jaume I, 2019.

[10] "Cryptography in the Quantum age," NIST, 29-Sep-2023. [En línea]. Disponible: https://www.nist.gov/physics/introduction-new-quantum-revolution/cryptography-quantum-age. [Consultado: 25-Sep-2023]

[11] M. S. Sharbaf, "Quantum Cryptography: A New Generation of Information Technology Security System,"2009 Sixth International Conference on Information Technology: New Generations, Las Vegas, NV, USA, 2009, pp. 1644-1648, doi: 10.1109/ITNG.2009.173.

[12] A. Sadovsky and J. Ortiz, Modern cryptography: Theory and applications. [En línea]. Disponible: https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/cryptography/quantum.html. [Consultado: 30-Sep-2023]

[13] C. Portmann and R. Renner, "Cryptographic security of quantum key distribution," arXiv.org, 11-Sep-2014. [En línea]. Disponible: https://arxiv.org/abs/1409.3525. [Consultado: 30-Sep-2023]

[14] W. Peter, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", SIAM J.Sci.Statist.Comput, 1997.

[15] A. G. López y J. G. López, Criptografía cuántica. Departamento Matemática Aplicada. 2005.

[16] D. W. Leung, "Quantum Vernam Cipher", 2000.

[17] D. F. Urrego, "Implementation of the protocol BB84 for encryption of a message", Universidad de Los Andes, Colombia, 2014.

[18] M. H. Saeed, H. Sattar, M. H. Durad and Z. Haider, Ïmplementation of QKD BB84 Protocol in Qiskit,"2022 19th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Islamabad, Pakistan, 2022, pp. 689-695, doi: 10.1109/IBCAST54850.2022.9990073.

[19] V. D. Sánchez, "Implementación del Protocolo de Distribución Cuántica de Claves, Protocolo BB84", Universidad Nacional Autónoma de México, México, 2015.

[20] J. L. Hita, "Así se podrán mejorar las inversiones financieras con computación cuántica", BBVA, 13-jul-2022. [En línea]. Disponible en: https://www.bbva.com/es/innovacion/asi-se-podran-mejorar-las-inversiones-financieras-con-computacion-cuantica/. [Consultado: 29-nov-2023].

[21] Communications, "Así es el mapa mundial de la computación cuántica", BBVA, 25-jul-2023. [En línea]. Disponible en: https://www.bbva.com/es/innovacion/asi-es-el-mapa-mundial-de-la-computacion-cuantica/. [Consultado: 29-nov-2023].

[22] "Computing classification system," ACM Digital Library, https://dl.acm.org/ccs [Consultado: 1-oct-2023].

[23] A. Ruiz, "La ciberseguridad en cifras: los datos muestran incremento en la preocupación empresarial - Tecnología para los negocios", Tecnología para los negocios, 12-jul-2019. [En línea]. Disponible en: https://ticnegocios.camaravalencia.com/servicios/tendencias/la-ciberseguridad-en-cifras-los-datos-muestran-incremento-en-la-preocupacion-empresarial/. [Consultado: 24-dic-2023].

[24] L. Méndez y E. Javier, "Ciberseguridad de Gobierno", Universidad Piloto de Colombia, 2015.

[25] D. Álvarez-Valenzuela y A. Hevia Angulo, "Protección legal para la búsqueda y notificación de vulnerabilidades de ciberseguridad en Chile", Rev. Chil. Derecho Tecnol., vol. 9, núm. 2, p. 1, 2020.

[26] V. Calcaterra Dufour, "Ciberseguridad en Uruguay: hacia la adhesión al Convenio de Budapest", 2022.

[27] U. Bekerman, Ed., Algunas medidas de ciberseguridad en Argentina, Colombia, Cuba, Egipto, Francia, Grecia, Japón, Singapur y Turquía. SSRN, 2021.

[28] D. AL-Mubayedh, M. AL-Khalis, G. AL-Azman, M. AL-Abdali, M. Al Fosail and N. Nagy, "Quantum Cryptography on IBM QX," 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia, 2019, pp. 1-6, doi: 10.1109/CAIS.2019.8769567. https://ieeexplore.ieee.org/document/8769567

[29] A. Biswas, A. Banerji, P. Chandravanshi, R. Kumar and R. P. Singh,"Experimental Side Channel Analysis of BB84 QKD Source," in IEEE Journal of Quantum Electronics, vol. 57, no. 6, pp. 1-7, Dec. 2021, Art no. 8000207, doi: 10.1109/JQE.2021.3111332. https://ieeexplore.ieee.org/document/9531968

[30] S. Wijesekera, S. Palit and B. Balachandran, "Software Development for B92 Quantum Key Distribution Communication Protocol," 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007), Melbourne, VIC, Australia, 2007, pp. 274-278, doi: 10.1109/ICIS.2007.168. https://ieeexplore.ieee.org/document/4276394

[31] Group Specification, "ETSI GS QKD 008 V1.1.1 (2010-12)", Etsi.org. [En línea]. Disponible en: https://www.etsi.org/deliver/etsi_gs/qkd/001_099/008/01.01.01_60/gs_qkd008v010101p.pdf. [Consultado: 29-may-2024].

[32] L. E. Bassham III et al., "A statistical test suite for random and pseudorandom number generators for cryptographic applications", National Institute of Standards and Technology, Gaithersburg, MD, 2010.

[33] "random — Generate pseudo-random numbers", Python documentation. [En línea]. Disponible en: https://docs.python.org/3/library/random.html. [Consultado: 11-jul-2024].

[34] G. Brassard y L. Salvail, "Secret-key reconciliation by public discussion", en Advances in Cryptology — EUROCRYPT '93, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 410–423.

[35] A. Leverrier, R. Alléaume, J. Boutros, G. Zémor, y P. Grangier, "Multidimensional reconciliation for a continuous-variable quantum key distribution", Phys. Rev. A, vol. 77, núm. 4, 2008.

[36] A. Mink y A. Nakassis, "LDPC for QKD Reconciliation", 2012.

[37] C. H. Bennett, G. Brassard, y J.-M. Robert, "Privacy amplification by public discussion", SIAM J. Comput., vol. 17, núm. 2, pp. 210–229, 1988.

[38] Sugimoto, A. y Yamazaki, T. (2000). A study on secret key reconciliation protocol Çascade". In IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences (Vol. E83-A, No. 10, pp. 1987-1991).

[39] G. Limei, R. Qi, J. Di, y H. Duan, "QKD iterative information reconciliation based on LDPC codes", Int. J. Theor. Phys., vol. 59, núm. 6, pp. 1717–1729, 2020.

[40] T. J. Richardson y R. L. Urbanke, "Efficient encoding of low-density parity-check codes", IEEE Trans. Inf. Theory, vol. 47, núm. 2, pp. 638–656, 2001.

[41] C. H. Bennett, G. Brassard, C. Crepeau, y U. M. Maurer, "Generalized privacy amplification", IEEE Trans. Inf. Theory, vol. 41, núm. 6, pp. 1915–1923, 1995.

[42] B. J. Frey y D. MacKay, .ª Revolution: Belief Propagation in Graphs with Cycles,.ª Advances in Neural Information Processing Systems, vol. 10, M. Jordan, M. Kearns y S. Solla, Eds. Cambridge, MA, USA: MIT Press, 1997. [En línea]. Disponible: https://proceedings.neurips.cc/paper_files/paper/1997/file/0245952ecff55018e2a459517fdb40e3-Paper.pdf.

[43] Group Specification, "ETSI GS QKD 005 V1.1.1 (2010-12)", Etsi.org. [En línea]. Disponible en: https://www.etsi.org/deliver/etsi_gs/qkd/001_099/005/01.01.01_60/gs_qkd005v010101p.pdf [Consultado: 29-may-2024].

[44] "Styleguide", styleguide. [En línea]. Disponible en: https://google.github.io/styleguide/pyguide.html. [Consultado: 30-jul-2024].