ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Mecánica cuántica y comunicaciones seguras: Simulación y análisis de prestaciones del protocolo BB84

Quantum Mechanics and Secure Communications: Simulation and Performance Analysis of the BB84 Protocol

Para acceder al Titulo de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Juan Ignacio Bilbao

Septiembre-2025

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Juan Ignacio Bilbao
Director del TFG: Luis Muñoz
Título: "Mecánica cuántica y comunicaciones seguras: Simulación y análisis de prestaciones del protocolo BB84"
Title: "Quantum Mechanics and Secure Communications: Simulation and Performance Analysis of the BB84 Protocol"

Presentado a examen el día:

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del tribunal:
Presidente (Apellidos, nombre): De la Fuente Rodríguez, Luisa María
Secretario (Apellidos, nombre): Díez Fernández, Luis Francisco
Vocal (Apellidos, nombre): Domingo Gracia, Marta

Este Tribunal ha resuelto otorgar la calificación de:
.....

Fdo: El Presidente **Fdo: El Secretario**

Fdo: El Vocal **Fdo: El Director del TFG**
(sólo si es distinto del Secretario)

V° B° del Subdirector Trabajo Fin de Grado N°
(a asignar por Secretaría)

Agradecimientos

A mi director, por su orientación y paciencia. Brindándome un apoyo fundamental para mantener la motivación en este largo trayecto.

A mis compañeros, amigos y pareja, por las conversaciones, el ánimo y las horas compartidas.

A mi familia, por traerme hasta aquí.

Índice general

Índice de figuras

Índice de cuadros

Índice de códigos

Resumen

Este Trabajo de Fin de Grado explora la implementación y análisis del protocolo BB84 de distribución cuántica de claves, integrando fundamentos teóricos de la mecánica cuántica con un desarrollo experimental en entornos de simulación y hardware cuántico real. El documento aborda, en primer lugar, los principios físicos esenciales para la seguridad de las comunicaciones, incluyendo fenómenos como el entrelazamiento, la superposición y la medición cuántica, así como los retos y ventajas que estas propiedades ofrecen frente a la criptografía clásica. Posteriormente, se presenta el estado del arte de la distribución cuántica de claves y de los sistemas cuánticos actuales, detallando arquitecturas, implementaciones y desarrollos recientes.

En la parte experimental, se diseña y construye un circuito cuántico que implementa el protocolo BB84, incorporando las fases de generación de bits aleatorios, transmisión, recepción y detección de intentos de espionaje. Se realizan simulaciones parametrizadas con modelos de ruido realistas y se ejecutan pruebas en un procesador cuántico de IBM, evaluando el impacto de la tasa de error cuántico (*Quantum Bit Error Rate* (QBER)) sobre el éxito del protocolo.

Palabras clave: distribución cuántica de claves; criptografía cuántica; computación cuántica; QKD; BB84.

Abstract

This Bachelor's dissertation explores the implementation and analysis of the BB84 quantum key distribution protocol, integrating the theoretical foundations of quantum mechanics with experimental development in both simulated environments and real quantum hardware. The document first addresses the essential physical principles underpinning secure communications, including phenomena such as entanglement, superposition, and quantum measurement, as well as the challenges and advantages these properties offer over classical cryptography. Subsequently, it presents a state-of-the-art review of quantum key distribution and current quantum systems, detailing architectures, implementations, and recent developments.

In the experimental section, a quantum circuit implementing the BB84 protocol is designed and built, incorporating the phases of random bit generation, transmission, reception, and detection of eavesdropping attempts. Simulations parameterized with realistic noise models are carried out, and tests are executed on an IBM quantum processor, assessing the impact of the quantum bit error rate (QBER) on the success of the protocol.

Keywords: quantum key distribution; quantum cryptography; quantum computing; QKD; BB84.

Capítulo 1 Introducción a la criptografía y las comunicaciones

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no hacía mucho tiempo que vivía un hidalgo al que no le llegaba el 5G [1]. Parece mentira, pues vivimos en un mundo global, interconectado, donde sin apenas esfuerzo los individuos pueden intercambiar inmensas cantidades de información a distancias y velocidades tan solo limitadas por la velocidad de la luz. Es por esto que resulta verdaderamente complicado imaginar un mundo donde la gente camina por la calle sin estar mirando el móvil.

Sin embargo, el momento actual es único y distinto en la historia. A través de los tiempos, los seres vivos han desarrollado distintos métodos para relacionarse con su entorno y compartir información. Muchos de los cuales pueden considerarse formas primitivas de comunicación. Desde señales químicas entre microorganismos hasta los elaborados bailes de las abejas, estas interacciones demuestran una capacidad inherente para transmitir información. Sin embargo, fue el desarrollo del lenguaje por parte de los homínidos lo que trasnsformó radicalmente esta capacidad.

Un sistena de comunicación codificado mediante vibraciones en el aire que con el tiempo aumentó en complejidad y abrió un mundo de posibilidades: sus reglas estructuradas y riqueza expresiva ofrecieron a los interlocutores la capacidad de compartir ideas, emociones y conocimientos. Gracias a ello, los seres humanos no solo comenzaron a comunicarse como lo entendemos hoy en día, sino también a crear historia. Una historia diferente a la de cualquier otra especie, donde por primera vez el ingenio superaría a la fuerza como motor de dominación en el planeta.

Desde tiempos antiguos, esa necesidad de dominación llevó a continuos conflictos con las demás especies, pero sobre todo, entre los propios humanos. Durante estos enfrentamientos, la necesidad de transmitir información de manera segura impulsó el desarrollo de técnicas rudimentarias de encriptación. Los primeros hitos criptográficos provienen de la Antigua Grecia y el Imperio Romano. La escítala espartana [2] o el ampliamente conocido cifrado del César [2] son ejemplos de métodos ideados para establecer la comunicación entre los mandos de los ejércitos, en un contexto donde garantizar la privacidad de las comunicaciones resultaba crucial para obtener una ventaja estratégica.

Sin embargo, fue en el siglo XX cuando esta disciplina alcanzó un punto de inflexión con la invención de la máquina Enigma [2] por parte de los alemanes y su posterior descifrado mediante la Bomba [2] . Esta última fue una máquina de

cálculo inventada por Alan Turing y otros criptoanalistas británicos en Bletchley Park, diseñada para descifrar los mensajes cifrados por la Enigma alemana.

La ruptura del cifrado de Enigma permitió a los Aliados interceptar y descifrar comunicaciones críticas del ejército alemán, obteniendo una ventaja estratégica que aceleró el final de la guerra y cambió la historia militar y tecnológica para siempre. Este hito sentó las bases de la computación electrónica moderna, introduciendo principios fundamentales como el procesamiento sistemático de datos y la verificación automática de hipótesis.

El perfeccionamiento de estos sistemas dio lugar a códigos más abstractos y compactos, como el sistema binario, esencial en el desarrollo de las telecomunicaciones modernas y la informática. Este código, basado en los valores 0 y 1, permitió transmitir grandes volúmenes de información de manera eficiente y sentó las bases para las tecnologías digitales.

Tras la Segunda Guerra Mundial, se desarrollaron enormemente los sistemas binarios, especialmente gracias a la publicación por parte de Claude Shannon de un artículo seminal en el Bell System Technical Journal [3], que completaría en su libro junto a Weaver, The Mathematical Theory of Communication [4]. Este trabajo sentó las bases de la teoría de la información, estableciendo los límites de la comunicación y definiendo las normas y procesos que permiten alcanzar dichos límites.

Gracias a diversos descubrimientos de la época se produjeron grandes avances en la electrónica y la computación que transformaron radicalmente la criptografía, extendiéndola más allá del ámbito militar. Dando comienzo la presente era de la información, donde el mundo gira entorno a Internet, la protección de datos personales, financieros y gubernamentales impulsan la creación de protocolos criptográficos cada vez más sofisticados y seguros.

En este contexto surgen protocolos de cifrado que protejan las comunicaciones frente a posibles ataques. Uno de los primeros estándares modernos fue el *Data Encryption Standard* (DES) [5], adoptado en 1975 por el entonces llamado National Bureau of Standards, actualmente conocido como *National Institute of Standards and Technology* (NIST). Aunque revolucionario en su época, el aumento de la capacidad computacional lo hizo vulnerable a ataques de fuerza bruta, siendo eventualmente reemplazado por el *Advanced Encryption Standard* (AES) [6] en 2001.

La criptografía experimentó otra revolución en 1976 con la introducción de la criptografía de clave asimétrica, gracias a Whitfield Diffie y Martin Hellman. Estos desarrollaron el algoritmo de Diffi-Helman [7], un método que utiliza dos claves: una pública para cifrar y una privada para descifrar, resolviendo el problema de la distribución segura de claves. Poco después, en 1978, el algoritmo Rivest, Shamir y Adleman (RSA) [8] introdujo un enfoque basado en la factorización de números primos, ofreciendo una combinación de seguridad y practicidad que lo convirtió en un estándar de la criptografía moderna.

En la década de 1980, se desarrollaron las técnicas criptográficas basadas en curvas elípticas, *Elliptic Curve Cryptography* (ECC) [2], que ofrecen seguridad equivalente a RSA con claves más pequeñas, haciendo más eficiente su implementación.

Este enfoque se ha vuelto esencial en aplicaciones modernas como dispositivos móviles y sistemas de *Internet of Things* (IoT), donde los recursos son limitados.

La red se ha convertido, así, en un puente seguro y virtual que trasciende fronteras y culturas, permitiendo un acceso sin igual a conocimientos, recursos y oportunidades. Internet es la herramienta que permite que este mundo global, donde la información fluye a velocidades inimaginables, tan solo limitado por la velocidad de la luz, sea una realidad.

Para alcanzar este hito, ha sido necesario un esfuerzo conjunto sin apenas precedentes en la historia. Sin embargo, se vislumbra una revolución potencialmente catastrófica que amenaza con desbaratar los cimientos de la seguridad digital: la llegada de la computación cuántica.

Desde sus primeras concepciones teóricas, la computación cuántica ha planteado un desafío fundamental a la seguridad de las comunicaciones cifradas. La forma en la que esta nueva tecnología procesa la información le permite descifrar en tiempo polinomial los sistemas de intercambio de clave actuales. Un eventual despliegue a gran escala de esta tecnología corrompería el estado actual de la red volviéndola vulnerable.

La seguridad de gran parte de las comunicaciones y transacciones digitales se basa en algoritmos de cifrado robustos, como el algoritmo RSA y el Whitfield Diffie and Martin Hellman (Diffie-Hellman). Estos algoritmos se basan en una clave privada protegida por la complejidad computacional de ciertos problemas matemáticos, como la factorización de números enteros y el problema del logaritmo discreto.

Pues bien, estos algoritmos están en peligro de quedar obsoletos de la mano de la computación cuántica. Algoritmos como el de Shor [9] han demostrado ser capaces de factorizar grandes números enteros en tiempo polinomial, lo que supondría una amenaza inminente para los sistemas de cifrado de clave pública.

La seguridad de las transacciones financieras, la privacidad de las comunicaciones personales y la integridad de los datos confidenciales estarían comprometidas. En este contexto de creciente urgencia, se vuelve imperativo explorar soluciones que puedan resistir el auge de la computación cuántica. La criptografía cuántica, y más concretamente la distribución cuántica de claves, *Quantum Key Distribution* (QKD) [10], emerge como un campo prometedor que utiliza los principios de la mecánica cuántica para garantizar la seguridad de las comunicaciones en un mundo post-cuántico.

La evolución de las comunicaciones nos lleva hoy al umbral de las comunicaciones cuánticas, donde los estados cuánticos ofrecen un código de dimensiones inéditas. Este avance no sólo promete revolucionar la seguridad de la comunicación, sino también la forma en que entendemos el intercambio de información.

De la palabra hablada al lenguaje cuántico, la historia de la comunicación es una crónica de la creatividad humana, impulsada por el deseo de superar las barreras del tiempo y el espacio de transmitir información. En el presente trabajo se analizará cómo el protocolo BB84, emplea las leyes de la física cuántica como el teorema de no clonación o el principio de incertidumbre para garantizar la privacidad de las

comunicaciones. Incluso frente a intentos de interceptación por parte de un espía. Destacando su seguridad eterna, o *everlasting security*, la cual, garantiza que si un atacante registra toda la comunicación, tanto cuántica como clásica, no podrá acceder a la clave secreta en el futuro, aunque disponga de capacidad computacional ilimitada.

Capítulo 2 Física cuántica y seguridad en las comunicaciones

La física cuántica ha transformado profundamente nuestra comprensión del universo, revolucionando desde los fundamentos teóricos hasta las aplicaciones tecnológicas más avanzadas de nuestros días. Aunque actualmente el término "cuántico" se asocia con ideas modernas y sofisticadas, esta contraintuitiva forma de comprender el universo surgió hace más de un siglo.

Este campo de la Física trata de describir el comportamiento de la materia y la luz en todos sus detalles, a escala atómica. A esa escala, los objetos no se comportan como nada con lo que tengamos experiencia directa: no son ondas, ni partículas, ni nubes, ni bolas de billar, ni resortes, ni nada conocido. [11].

Newton pensaba que la luz estaba compuesta por partículas, pero se descubrió que se comportaba como una onda. Más tarde, se encontró que a veces la luz también se comporta como una partícula. De forma similar, se creía que el electrón era una partícula, pero en muchos aspectos actúa como una onda. Por tanto, no es ni lo uno ni lo otro. Comportamiento que también se aplica a la luz y otros objetos atómicos (electrones, protones, neutrones, fotones, etc.). Todos muestran un comportamiento dual, de onda-partícula. Esta concepción surgió en 1924, cuando el físico francés Louis de Broglie propuso que toda partícula con masa podía describirse como una onda, completando la visión de la dualidad onda-partícula

Este comportamiento único se produce a escalas en las que ciertos pares de magnitudes como posición y momento, o posición y velocidad no se pueden definir con prescisión al mismo tiempo (escalas atómicas y subátomicas). Esta limitación propuesta en 1927 por Heisenberg en su principio de incertidumbre significaría la culminación de los fundamentis de la teoría cuántica.

El camino hacia esta nueva comprensión de la materia comenzó a principios del siglo XX, cuando varios fenómenos no podían ser explicados por la física clásica. En 1900, el físico alemán Max Planck propuso que la energía no se intercambiaba de forma continua, sino en pequeñas unidades discretas que denominó [12]. Este planteamiento surgió como solución al problema de la radiación del cuerpo negro y marcó el inicio de la física cuántica.

Pocos años después, en 1905, Albert Einstein amplió esta idea al estudiar el efecto fotoeléctrico. Propuso que la luz también estaba compuesta por partículas, o cuantos de luz, que más tarde serían conocidos como fotones. Este enfoque explicaba cómo la luz podía arrancar electrones de una superficie metálica, conocido como el efecto

fotoeléctrico, un fenómeno que la teoría ondulatoria no lograba justificar.

En 1913, Niels Bohr aplicó el concepto de cuantización al modelo atómico. Sugirió que los electrones no orbitaban el núcleo de forma continua, como los planetas alrededor del Sol, sino que solo podían ocupar ciertos niveles de energía fijos. Estos niveles representaban los únicos estados estables posibles, y un electrón solo podía pasar de uno a otro emitiendo o absorbiendo energía en forma de fotón.

Estos descubrimientos sentaron las bases de una teoría radicalmente nueva: la mecánica cuántica, un marco teórico que describe el comportamiento de la materia y la energía a escalas subatómicas, donde las reglas del mundo clásico o macroscópico dejan de ser válidas.

Una de las demostraciones más claras y fascinantes de los principios de la mecánica cuántica es el experimento de la doble rendija [13], que revela cómo partículas como los electrones pueden comportarse como ondas bajo ciertas condiciones. Este experimento no solo contrasta con la intuición clásica sobre la materia, sino que también expone el núcleo de los misterios cuánticos: la dualidad onda-partícula, la interferencia cuántica y el papel fundamental del observador en la determinación del resultado de un fenómeno físico.

2.1 Experimento de la doble rendija

El experimento de la doble rendija, propuesto inicialmente por Thomas Young en 1801, demuestra la dualidad onda-partícula de la luz y otras partículas subatómicas. Consiste en dirigir una fuente de partículas (como electrones o fotones) hacia una barrera con dos rendijas paralelas y observar el patrón de interferencia resultante en una pantalla situada detrás de la barrera.

Cuando ambas rendijas están abiertas y se envían partículas de una en una pero no se observa cuál atraviesa, se forma un patrón de interferencia característico de las ondas, con franjas alternas de máximos y mínimos de intensidad como se muestra en la Figura 2.1. Esto sugiere que cada partícula interfiere consigo misma, comportándose como una onda que pasa por ambas rendijas simultáneamente.

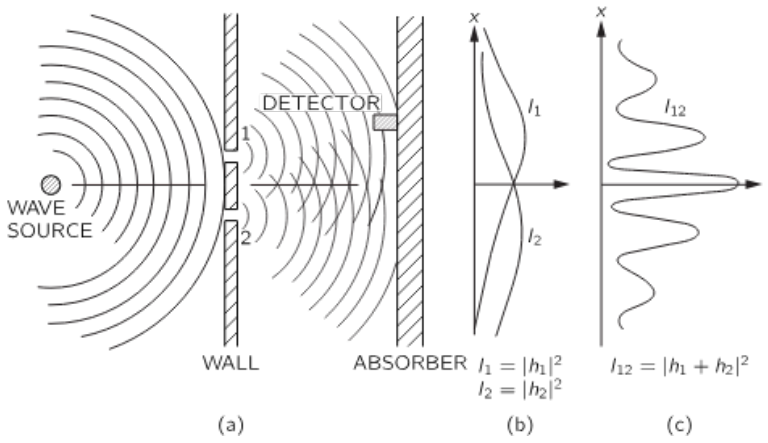

Figura 2.1: Comportamiento de electrón sin observar

Sin embargo, si se coloca un detector para determinar por qué rendija pasa cada partícula el patrón de interferencia desaparece y se observa un patrón de dos bandas, propio de partículas clásicas, Figura 2.2. Este resultado indica que las partículas se comportan en libertad como ondas, pero que en el momento en el que se mide en busca del electrón se le fuerza a determinar su posición pasando por una sola de las rendijas.

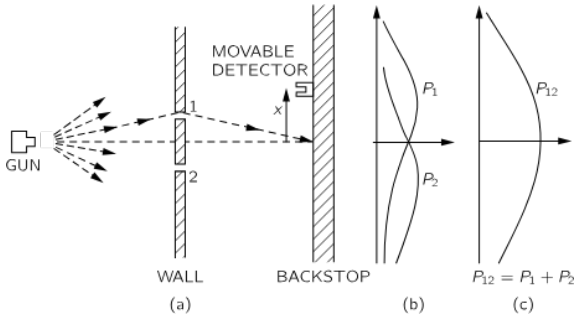

Figura 2.2: Comportamiento electron siendo observado

A la luz de este experimento, parece que ciertas partículas no se encuentran definidas en un único lugar mientras no se mida su posición, sino que existen en una superposición de estados posibles. En ausencia de observación, cada partícula se comporta como si atravesara simultáneamente ambas rendijas, interferiendo consigo misma. Esta sorprendente propiedad no solo desafía la lógica clásica, sino que constituye uno de los pilares fundamentales de la mecánica cuántica: el principio de superposición, según el cual un sistema cuántico puede encontrarse en múltiples estados a la vez hasta que una medición obliga al sistema a colapsar en uno de ellos.

2.2 Estado cuántico y la función de onda

El estado de un sistema físico está determinado por el conjunto completo de sus propiedades observables. Estas propiedades, tales como la posición, la velocidad o la temperatura, permiten caracterizar de manera precisa el comportamiento del sistema en estudio.

En mecánica clásica, el estado del sistema puede describirse mediante funciones deterministas de posición y tiempo. Por ejemplo, considerando un sistema con vector posición $\mathbf{r}$ y tiempo t, su estado puede definirse por una función del tipo:

$$Estado = x(\mathbf{r}, \mathbf{p}) \tag{2.1}$$

Dado que estas funciones son completamente deterministas, es posible conocer simultáneamente, y con precisión arbitraria, todas las propiedades medibles del sistema, tales como su posición y velocidad. De este modo, el sistema se caracteriza por ser predecible y determinista, siendo el estado del sistema completamente conocido en cualquier instante futuro mediante el conocimiento exacto de sus condiciones iniciales.

Por otro lado, los estados cuánticos o estados de los sistema cuánticos siguen estando relacionados con el conjunto de las propiedades medibles del sistema. Su evolución es caracterizable mediante la función de onda o ecuación de Shrodinger siempre que no se efectúen medidas experimentales sobre la misma. [14]

2.2.1 Ecuación de Schrödinger

La ecuación de Schrödinger constituye el fundamento esencial de la mecánica cuántica, adoptando diferentes formas según el contexto físico específico que se analice. En su formulación más general, esta ecuación se expresa como una ecuación en derivadas parciales, cuya solución $\Psi(\mathbf{r}, t)$ describe cómo evoluciona un sistema cuántico con el tiempo.

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r}, t) = \hat{H}\Psi(\mathbf{r}, t) \tag{2.2}$$

El operador $\hat{H}$ (2.3), conocido como *Hamiltoniano*, representa la energía total del sistema (energía cinética más energía potencial) y determina cómo cambia la función de onda en el tiempo, donde el primer término es la energía cinética y $V(\mathbf{r}, t)$ la energía potencial. Así, $\hat{H}$ es el operador responsable de la dinámica cuántica del sistema.

$$\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + V(\mathbf{r}, t) \tag{2.3}$$

Durante esta evolución previa a cualquier medición, el estado de una partícula no se encuentra definido de forma única, sino que puede describirse como una superposición de varios estados posibles. Esta propiedad, conocida como superposición cuántica, implica que el sistema puede existir simultáneamente en múltiples configuraciones hasta que se realiza una observación. Es precisamente el acto de medir el que provoca el colapso de la función de onda, forzando al sistema a adoptar uno de los posibles estados definidos con cierta probabilidad.

2.2.2 Principio de superposición

En los sistemas cuánticos, los estados pueden representarse como combinaciones lineales de dos o más estados independientes. Generalizando para un conjunto de N estados ortonormales $\{|\phi_i\rangle\}_{i=1}^{N}$, un estado cuántico arbitrario $|\psi\rangle$ puede escribirse como:

$$|\psi\rangle = \sum_{i=1}^{N} \alpha_i \, |\phi_i\rangle \tag{2.4}$$

donde los coeficientes $\alpha_i \in \mathbb{C}$, en su forma $|\alpha_i|^2$ representan la probabilidad de que el sistema se encuentre en el estado $|\phi_i\rangle$. Estos coeficientes deben satisfacer la condición de normalización:

$$\sum_{i=1}^{N} |\alpha_i|^2 = 1 \tag{2.5}$$

Para ilustrarlo con un caso particular, consideremos $N = 2$. El estado cuántico puede entonces representarse como en la Ecuación 2.6 donde $|\phi_1\rangle$ y $|\phi_2\rangle$ son estados base ortonormales, y $\alpha, \beta \in \mathbb{C}$, tal que:

$$|\psi\rangle = \alpha|\phi_1\rangle + \beta|\phi_2\rangle \Leftrightarrow |\alpha|^2 + |\beta|^2 = 1 \tag{2.6}$$

Se puede observar que, el concepto de superposición cuántica guarda una estrecha relación con los juegos de azar. Por ejemplo, un dado posee seis posibles estados en superposición antes de ser lanzado. Al realizar una *medición* —es decir, al lanzar el dado— el sistema colapsa a uno de los seis posibles resultados.

2.3 Colapso de la función de onda

En la interpretación estándar de la Mecánica Cuántica (o interpretación de Copenhague), se postula que la función de onda *colapsa* tras una medición, quedando proyectada en un autovalor determinado del observable medido. Así, la medición

introduce un cambio discontinuo del estado cuántico, lo cual difiere de la evolución temporal descrita por la ecuación de Schrödinger, que es un proceso continuo.

Este colapso de la función de onda se corresponde con el proceso por el cual el estado cuántico deja de estar caracterizado como combinación lineal de uno o más estados. Adquiriendo las características de tan solo uno de ellos, obviando al resto. Este proceso no se produce de forma aleatoria, sino que da lugar en función de una serie de los coeficientes de la función de onda como sugirio Born.

El cuadrado del módulo de la función de onda, $|\psi(\mathbf{r},t)|^2$, representa la función densidad de probabilidad, Ecuación 2.7, de encontrar la partícula en la posición $\mathbf{r}$ en el instante t.

$$f(\mathbf{r},t) = |\psi(\mathbf{r},t)|^2 = |\psi\rangle\langle\psi| \tag{2.7}$$

De este modo, la probabilidad de hallar a la partícula en una región espacial $\mathbb{R}^3$ está dada por

$$P(\text{partícula en } \mathbb{R}^3) = \int_{\mathbb{R}^3} |\psi(\mathbf{r},t)|^2 \, d^3r. \Leftrightarrow \int_{\mathbb{R}^3} |\psi(\mathbf{r},t)|^2 \, d^3r \;=\; 1. \tag{2.8}$$

Para que la función de onda describa adecuadamente un estado físico, esta debe estar normalizada, es decir:

$$\int_{\mathbb{R}^3} |\psi(\mathbf{r},t)|^2 \, d^3r \;=\; 1.$$

2.4 Principio de Incertidumbre de Heisenberg

Este contexto cuántico en el que el que el resultado de la medición (u observación) no se puede conocer sin perturbar el estado original del sistema fue descrito por Werner Heinseberg en 1927. Este problema intrínseco de la naturaleza se conoce como principio de incertidumbre de Heinsenberg o principio de indeterminación [14].

Este postulado representa una restricción física de la medida, independiente de los instrumentos empleados en el proceso de medida. Establece que no es posible conocer con precisión arbitraria ciertos pares de observables canónicamente conjugados, como la posición x y el momento p, de manera simultánea. Matemáticamente, esta limitación se expresa mediante la desigualdad (2.9).

$$\Delta x \, \Delta p \geq \frac{\hbar}{2} \tag{2.9}$$

donde Δx y Δp representan las desviaciones estándar de la posición y el momento, respectivamente, y $\hbar$ es la constante de Planck reducida, un valor fijo de la naturaleza que equivale aproximadamente a 1.05×10^{-34} julios por segundo.

2.5 Entrelazamiento cuántico y producto tensorial

El entrelazamiento es otro de los de los fenómenos cuánticos intrigantes y, por qué no, contraintuitivos de la mecánica cuántica. Se refiere a la correlación entre estados de dos o más partículas, de forma que el estado de cada una no puede describirse independientemente del estado de las demás, incluso si se encuentran separadas por grandes distancias.

Cuando dos partículas están entrelazadas, una medición realizada sobre una de ellas afecta instantáneamente el estado de la otra sin importar la distancia que las separe. Este comportamiento, que Einstein denominó "acción fantasmal a distancia", ha sido ampliamente confirmado experimentalmente y es la base de tecnologías emergentes como la computación cuántica, la criptografía cuántica y la teletransportación cuántica.

El entrelazamiento no implica transmisión de información más rápida que la velocidad de la luz, ya que las correlaciones cuánticas no pueden ser utilizadas para comunicar información sin un canal clásico adicional.

2.5.1 Producto tensorial

Desde el punto de vista matemático, el entrelazamiento se describe utilizando el formalismo de espacios de Hilbert [14]. Si consideramos dos sistemas cuánticos A y B con espacios de Hilbert $\mathcal{H}_A$ y $\mathcal{H}_B$, el sistema conjunto se describe en el espacio tensorial $\mathcal{H}_A \otimes \mathcal{H}_B$.

El producto tensorial $\mathcal{H}_A \otimes \mathcal{H}_B$ es un nuevo espacio de Hilbert que contiene todos los posibles estados conjuntos del sistema compuesto $A \otimes B$. Si $\mathcal{H}_A$ y $\mathcal{H}_B$ son espacios de dimensión finita, por ejemplo, $\mathcal{H}_A = \mathbb{C}^2$ y $\mathcal{H}_B = \mathbb{C}^2$, un vector en el espacio conjunto puede escribirse como $|\psi\rangle_A \otimes |\phi\rangle_B$, donde $|\psi\rangle_A \in \mathcal{H}_A$ y $|\phi\rangle_B \in \mathcal{H}_B$. Por ejemplo, si $|\psi\rangle_A = \begin{pmatrix}1\\0\end{pmatrix}$ y $|\phi\rangle_B = \begin{pmatrix}0\\1\end{pmatrix}$, entonces su producto tensorial es

$$|\psi\rangle_A \otimes |\phi\rangle_B = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}.$$

Este nuevo vector habita en $\mathbb{C}^4$ y es isomorfo a $\mathcal{H}_A \otimes \mathcal{H}_B$. No todos los vectores en este espacio pueden escribirse como productos tensoriales de dos vectores separados; aquellos que no pueden descomponerse de esta forma se denominan estados entrelazados. Un ejemplo típico de estado entrelazado es el estado de Bell $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, que no puede factorizarse de la forma $|\psi\rangle_A \otimes |\phi\rangle_B$.

2.5.2 Estados separables y entrelazados

Un estado puro $|\psi\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$ es aquel que describe completamente el sistema cuántico y puede representarse mediante un único vector en el espacio de Hilbert. Se dice que un estado es separable (o no entrelazado) si puede escribirse como el producto tensorial de dos estados individuales:

$$|\psi\rangle = |u\rangle \otimes |v\rangle \tag{2.10}$$

En caso contrario, si no puede factorizarse de esta forma, el estado se considera **entrelazado**, lo que implica la existencia de correlaciones cuánticas entre los subsistemas.

La expresión siguiente muestra un estado aparentemente complejo:

$$|\psi\rangle = \frac{1}{2}|a_1\rangle \otimes |b_1\rangle + \frac{1}{2}|a_1\rangle \otimes |b_2\rangle + \frac{1}{2}|a_2\rangle \otimes |b_1\rangle + \frac{1}{2}|a_2\rangle \otimes |b_2\rangle \tag{2.11}$$

Sin embargo, este estado puede factorizarse como:

$$|\psi\rangle = \left(\frac{1}{\sqrt{2}}(|a_1\rangle + |a_2\rangle)\right) \otimes \left(\frac{1}{\sqrt{2}}(|b_1\rangle + |b_2\rangle)\right) \tag{2.12}$$

Dado que la ecuación (2.11) puede escribirse como el producto tensorial mostrado en la ecuación (2.12), concluimos que se trata de un estado puro y separable, es decir, no presenta entrelazamiento cuántico.

De forma opuesta, un estado puro que no pueda expresarse como producto tensorial de estados individuales se clasifica como un estado entrelazado. Estos estados manifiestan correlaciones cuánticas que no pueden explicarse mediante probabilidades clásicas y representan uno de los recursos fundamentales de la computación y la información cuántica.

El siguiente estado de Bell [15] es un ejemplo típico:

$$|\phi\rangle = \frac{1}{\sqrt{2}}(|a_1\rangle \otimes |b_1\rangle + |a_2\rangle \otimes |b_2\rangle) \tag{2.13}$$

Este estado no admite una descomposición en forma de producto tensorial de dos vectores individuales, lo cual se puede verificar intentando factorizarlo algebraicamente. La imposibilidad de escribirlo como $|u\rangle \otimes |v\rangle$ indica que existe entrelazamiento cuántico entre los subsistemas A y B.

Por tanto, el estado descrito en la ecuación (2.13) es un estado puro entrelazado, lo que implica que la medición de uno de los subsistemas afecta instantáneamente

al estado del otro. Dando valores, los cuatro estados de Bell forman una base ortonormal de estados entrelazados en el espacio de dos qubits como se muestra en la Ecuación 2.14.

$$\begin{aligned} |\Phi^+\rangle &= \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \\ |\Phi^-\rangle &= \frac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right) \\ |\Psi^+\rangle &= \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right) \\ |\Psi^-\rangle &= \frac{1}{\sqrt{2}}\left(|01\rangle - |10\rangle\right) \end{aligned} \tag{2.14}$$

Al medir uno de los qubits de un estado entrelazado, el estado del sistema colapsa instantáneamente en un estado correspondiente. Por ejemplo, si medimos el primer qubit de $|\Phi^+\rangle$ y obtenemos $|0\rangle$, el estado total colapsa a $|00\rangle$; si obtenemos $|1\rangle$, el sistema colapsa a $|11\rangle$.

2.6 Teorema de no clonación

El teorema de no clonación es un principio esencial en física cuántica que afirma que no es posible clonar de manera exacta el estado cuántico desconocido de un sistema físico sin alterar el estado original. Este teorema, propuesto en 1982 por Wootters y Zurek [15], es la continuación lógica del colapso de la función de onda y el principio de incertidumbre.

Al saber que no se puede conocer un estado sin obligarlo a colapsar y tomar un valor concreto —ya que en este proceso se pierde la información del resto de estados en superposición—, parece coherente que no se podría copiar o clonar dicho estado sin conocerlo. Pero, como se ha visto hasta ahora, la realidad de la mecánica cuántica no siempre coincide con la intuición.

Para poder demostrar este postulado matemáticamente, se emplea nuevamente el producto tensorial y se llamará U al operador de clonación deseado. Este operador debería ser capaz de actuar sobre el estado inicial $|u\rangle \otimes |0\rangle$ y convertirlo en el estado final $|u\rangle \otimes |u\rangle$. Es decir, convertir el estado $|0\rangle$ en el estado $|u\rangle$.

$$U(|u\rangle \otimes |0\rangle) = |u\rangle \otimes |u\rangle. \tag{2.15}$$

Pero esa operación debe poder ejecutarse para un estado desconocido (un estado arbitrario cualquiera). Se supone entonces que se parte de un estado diferente, $|v\rangle$. Entonces:

$$U(|v\rangle \otimes |0\rangle) = |v\rangle \otimes |v\rangle. \tag{2.16}$$

De todos modos, una operación unitaria debe preservar el producto escalar, de manera que:

$$(\langle u| \otimes \langle 0|)(|v\rangle \otimes |0\rangle) = (\langle u| \otimes \langle u|)(|v\rangle \otimes |v\rangle). \tag{2.17}$$

Simplificando:

$$\langle u|v\rangle\langle 0|0\rangle = \langle u|v\rangle\langle u|v\rangle. \tag{2.18}$$

Dado que $\langle 0|0\rangle = 1$, esto implica que:

$$\langle u|v\rangle = (\langle u|v\rangle)^2 . \tag{2.19}$$

La ecuación anterior sólo se cumple si $\langle u|v\rangle = 0$ o $\langle u|v\rangle = 1$, lo que significa que únicamente se pueden clonar estados cuánticos si son ortogonales o idénticos.

Esto demuestra que el operador U no puede ser lineal y universal al mismo tiempo, lo cual contradice los principios fundamentales de la mecánica cuántica y hace imposible la clonación de un estado cuántico arbitrario. En consecuencia, los estados cuánticos desconocidos no pueden ser clonados, lo que implica que un observador externo no puede obtener información exacta sobre el estado de un sistema sin perturbarlo de forma irreversible. Cualquier intento de medir o copiar información cuántica altera inevitablemente el sistema original, haciendo evidente que ha sido observado.

2.7 Introducción a la computación e intercambio de clave cuántica

La primera formulación clara y explícita de la idea moderna de computación cuántica se atribuye al físico Richard Feynman, quien en 1982 señaló que una computadora cuántica podría simular sistemas físicos cuánticos de manera eficiente, algo extremadamente difícil o imposible para una computadora clásica [16]. Posteriormente, en 1985, David Deutsch [17] planteó formalmente un modelo teórico completo y general de computadora cuántica, conocido como la máquina de Turing cuántica.

La computación cuántica representa un cambio de paradigma en el procesamiento de la información con respecto a la computación tradicional. Aunque esta última ha servido como inspiración y base conceptual, la computación cuántica opera bajo principios diferentes, fundamentados en la mecánica cuántica.

La computación clásica ha sido clave en el desarrollo tecnológico desde finales del siglo XX gracias a su facilidad de implementación, simplicidad matemática debido al modelo binario basado en bits, unidad mínima de información que adopta únicamente los valores 0 o 1. Esta lógica binaria se implementa a través del álgebra booleana, utilizando puertas lógicas organizadas en circuitos electrónicos que procesan datos de manera secuencial o paralela a velocidades que pasaron de pocos miles de instrucciones a miles de millones de instrucciones por segundo. A pesar de este desarrollo, la computación clásica encuentra limitaciones frente a ciertos problemas altamente complejos, como la factorización eficiente de grandes números o la optimización de sistemas con múltiples variables interdependientes.

Es en este contexto en el que la computación cuántica cobra importancia. Si bien su implementación es considerablemente más compleja, introduce conceptos fundamentales como la superposición y el entrelazamiento cuántico para realizar operaciones que no tienen un equivalente clásico. Estas propiedades permiten procesar simultáneamente múltiples estados posibles, transformando un conjunto de estados superpuestos en una distribución probabilística de resultados en lugar de una única salida concreta.

2.8 Qubit

Un qubit, o bit cuántico, es la unidad básica de información utilizada para codificar datos en computación cuántica y puede entenderse mejor como el equivalente cuántico del bit tradicional utilizado por los ordenadores clásicos para codificar información en binario. [18]

Sin embargo, los qubits no se representan como variables discretas que toman valores fijos de 0 o 1, sino que pueden mantenerse en un estado de superposición cuántica, donde coexisten simultáneamente como una combinación lineal de estados base.

Al igual que el sistema de coordenadas cartesiano se describe mediante los ejes x, y y z, el estado de un qubit puede representarse en un espacio tridimensional utilizando la esfera de Bloch.

2.8.1 Esfera de Bloch

La esfera de Bloch [16] es una representación geométrica del estado de un qubit en un espacio tridimensional. En computación cuántica, cualquier estado cuántico puro de un qubit se expresa como una combinación lineal de sus estados base $|0\rangle$ y $|1\rangle$, mediante la ecuación (2.20)

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle. \tag{2.20}$$

Donde θ y ϕ representan las coordenadas angulares en la esfera de Bloch. El ángulo θ define la inclinación del estado del qubit respecto al eje z, determinando la proporción en que los estados base $|0\rangle$ y $|1\rangle$ contribuyen a la superposición. Por otro lado, ϕ representa la fase relativa entre ambos estados base, afectando la interferencia cuántica entre ellos. Además, el término $e^{i\phi}$ introduce un componente de fase global, que es fundamental en la manipulación y evolución de los estados cuánticos.

Así como se denominan a $|0\rangle$ y $|1\rangle$ los estados base correspondientes al eje z, para los ejes x e y también hay bases de representación, siendo $|+\rangle$ y $|-\rangle$ para el eje x e $|i\rangle$ y $|-i\rangle$ para el eje y, como se muestra en la Figura 2.3.

Esta representación visual y matemática permite describir cualquier qubit puro combinación lineal de sus estados base como un punto en la superficie de la esfera de Bloch, facilitando la interpretación de las operaciones y su evolución dinámica bajo diferentes puertas lógicas.

2.8.2 Representación matricial

Los distintos estados base de un qubit se pueden expresar en forma matricial de la siguiente forma:

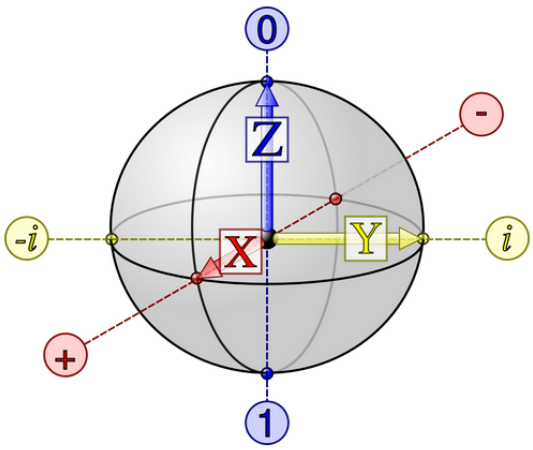

Figura 2.3: Representación de la Esfera de Bloch.

Para los estados $|0\rangle$ y $|1\rangle$, donde su base es la del eje Z, base horizontal o base computacional:

- $|0\rangle$: Polarización horizontal (0°).
- $|1\rangle$: Polarización vertical (90°).

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Por otro lado, en el eje X, la base diagonal o base Hadamard se encuentran estos dos estados:

- $|+\rangle$: Polarización diagonal derecha (45°).
- $|-\rangle$: Polarización diagonal izquierda (135°).

$$|+\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad |-\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Los estados $|+\rangle$ y $|-\rangle$ se relacionan con $|0\rangle$ y $|1\rangle$ mediante la siguiente transformación:

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

En definitiva, cualquier qubit puede describirse como combinacion lineal de los estados base $|0\rangle$ y $|1\rangle$. Incluyendo la superposición de ambos, como cualquier otro estado cuántico, tal y como se muestra en la ecuación (2.21).

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \Leftrightarrow |\alpha|^2 + |\beta|^2 = 1 \tag{2.21}$$

2.9 Puertas cuánticas

En la computación clásica, las operaciones se realizan mediante puertas lógicas que manipulan los bits. De una forma análoga, en la computación cuántica se utilizan puertas cuánticas para operar sobre los qubits.

Las puertas cuánticas son los bloques fundamentales para construir circuitos cuánticos. A diferencia de las puertas clásicas, que procesan bits de manera determinista, las puertas cuánticas pueden crear superposiciones, introducir entrelazamiento y manipular qubits en el espacio de la esfera de Bloch. .

Las puertas lógicas clásicas trabajan con n bits de entrada y producen una única salida. En cambio, las puertas cuánticas pueden actuar sobre n qubits y generar una salida en superposición de hasta 2^n estados posibles de forma simultánea. Esta característica otorga a la computación cuántica una capacidad de procesamiento exponencialmente mayor frente a la computación clásica. Estas operaciones permiten el desarrollo de algoritmos cuánticos como el algoritmo de Shor para factorización y el algoritmo de Grover [15] para búsqueda en bases de datos

Estas operaciones cuánticas (puertas cuánticas) se representan mediante matrices unitarias, lo que garantiza la reversibilidad de las operaciones cuánticas. Matemáticamente, una puerta cuántica U debe satisfacer la propiedad de ser unitaria, esto es:

$$UU^\dagger = U^\dagger U = I, \tag{2.22}$$

siendo $U^\dagger$ la traspuesta conjugada de U e I la matriz identidad.

2.9.1 Puertas de un solo qubit

Las puertas cuánticas que actúan sobre un solo qubit están pensadas para hacer rotaciones del estado. Por ejemplo, una rotación de 180^{o} sin realizar cambios sobre los ejes de polarización sería el equivalente a una puerta NOT clásica en el dominio cuántico. Mientras que rotaciones o proyecciones sobre los ejes no tendrían un equivalente clásico.

El conjunto de las puertas NOT-cuánticas que se obtiene mediante las matrices de Pauli [16] tienen una estructura distinta en cada base de polarización.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.23}$$

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{2.24}$$

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.25}$$

Por otro lado, la puerta encargada de cambiar las bases de polarización se conoce como puerta de Hadamard o puerta H. Una de las puertas más utilizadas, permi-

te transformar la base computacional $\{|0\rangle, |1\rangle\}$ en la base diagonal $\{|+\rangle, |-\rangle\}$, y viceversa.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.26}$$

Las transformaciones que realiza la puerta H sobre los principales estados son las siguientes:

$$H|0\rangle = |+\rangle \qquad H|1\rangle = |-\rangle \qquad H|+\rangle = |0\rangle \qquad H|-\rangle = |1\rangle$$

2.9.2 Puertas de Dos Qubits

Por otro lado, las puertas de dos qubits permiten la interacción entre qubits y generan entrelazamiento cuántico. Algunas de las más utilizadas son:

- **Puerta CNOT (Controlled-NOT)**: Actúa sobre dos qubits, donde el primer qubit es el *control* y el segundo es el *objetivo*. Su matriz es:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.27}$$

  Si el qubit de control está en $|1\rangle$, el qubit objetivo cambia su estado $|0\rangle \leftrightarrow |1\rangle$.

- **Puerta SWAP**: Intercambia dos qubits:

$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.28}$$

- **Puerta de Toffoli** (CCNOT): Es una generalización de CNOT donde dos qubits controlan la inversión de un tercer qubit.

2.10 Principios de QKD

Como se comentó en la introducción, el avance de la computación cuántica ha puesto en entredicho la seguridad de muchos sistemas criptográficos clásicos. Ante este escenario, surge la necesidad de replantear los métodos de intercambio de claves, manteniendo la confidencialidad en entornos potencialmente adversos. En este contexto aparece la QKD, una de las aplicaciones más prometedoras de la mecánica cuántica a la criptografía. A diferencia de los sistemas clásicos, cuya seguridad depende de la dificultad computacional de ciertos problemas matemáticos, la QKD se

fundamenta en principios físicos inviolables de la teoría cuántica, como el principio de incertidumbre de Heisenberg y el teorema de no clonación.

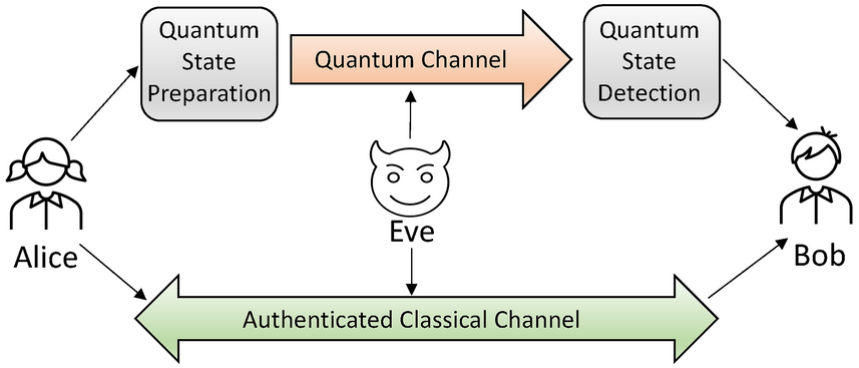

Figura 2.4: Protocolo habitual QKD

Estos principios aseguran que cualquier intento de interceptación por parte de un tercero, comúnmente referido como *eavesdropper* (Eve), deja huellas detectables en el sistema, permitiendo al transmisor legítimo (Alice) y receptor legítimo (Bob) identificar la presencia de un espía y abortar la comunicación si es necesario. Así, QKD ofrece un paradigma de seguridad cuyas garantías no dependen de la capacidad computacional del adversario, sino de las leyes fundamentales de la física.

En un sistema típico QKD, las dos partes desean generar una clave secreta compartida utilizando un canal cuántico (por ejemplo, fibra óptica o un canal libre) y un canal clásico autenticado, Figura 2.4 [19]. Un posible esquema de funcionamiento básico es el siguiente:

El protocolo comienza cuando Alice codifica una secuencia de qbits aleatoria en los estados cuánticos correspondientes, utilizando propiedades como la polarización (vertical u horizontal). Para ello, emplea dos bases que son mutuamente incompatibles, de forma que una medición en la base incorrecta introduce incertidumbre y rompe la coherencia cuántica del sistema.

Una vez transmitidos los fotones, Bob los mide aleatoriamente en una de las bases disponibles. Como no conoce la base utilizada por Alice, sus mediciones solo serán correctas cuando, por azar, haya coincidido con la base de codificación.

Posteriormente, Alice y Bob emplean un canal clásico para comunicar qué base utilizaron en cada caso, sin revelar los valores medidos. Descartan entonces todos los casos en los que sus bases no coincidieron, conservando únicamente los resultados obtenidos con bases compatibles. De esta forma, ambos extraen una clave bruta, *raw key*, a partir de los datos coincidentes.

A continuación, realizan un análisis de la tasa de error cuántica (*Quantum Bit Error Rate* (QBER)), que permite detectar la posible presencia de un espía, al que se suele denominar Eve. Si el valor de QBER se encuentra por debajo de un umbral aceptable, se cancela la compartición de la clave o se aplican protocolos de corrección de errores y de refuerzo de la privacidad, como el protocolo *CASCADE*, con el fin de obtener una clave final secreta, segura y compartida.

Si bien hay diversos ejemplos de protocolos QKD funcionales, eficientes en mayor o menor medida, destacan: Ekert 1991 (E91), basado en el entrelazamiento cuántico, propuesto por Ekert en 1991; Bennett 1992 (B92), variante más simple del Bennett y Brassard 1984 (BB84), que utiliza solo dos estados cuánticos; *Differential Phase Shift* (DPS), *Continuous Variable QKD* (CV-QKD) y *Measurement Device Independent QKD* (MDI-QKD), protocolos avanzados que abordan problemas prácticos como el ataque del detector y la implementación en redes metropolitanas.

2.11 Protocolo BB84

El protocolo más conocido para QKD y fundamento de este trabajo es el BB84, propuesto por Bennett y Brassard en 1984. Este esquema utiliza cuatro estados de polarización en dos bases distintas para codificar la información.

Alice genera una secuencia de bits aleatorios y los codifica empleando dos bases de polarización elegidas también de forma aleatoria. La primera es la base computacional, con los estados $|0\rangle$ y $|1\rangle$, y la segunda es la base diagonal, con los estados $|+\rangle$ y $|-\rangle$. Estos estados cuánticos se transmiten a Bob a través de un canal cuántico, donde este intenta medirlos utilizando igualmente bases seleccionadas al azar.

Posteriormente, Alice y Bob comparan públicamente (mediante un canal clásico) qué bases emplearon en cada posición de la transmisión, descartando aquellas en las que no coincidieron. El resultado de este proceso es una clave compartida, compuesta únicamente por los bits correspondientes a las mediciones realizadas en bases compatibles. Como se muestra en la Figura 2.5 [20].

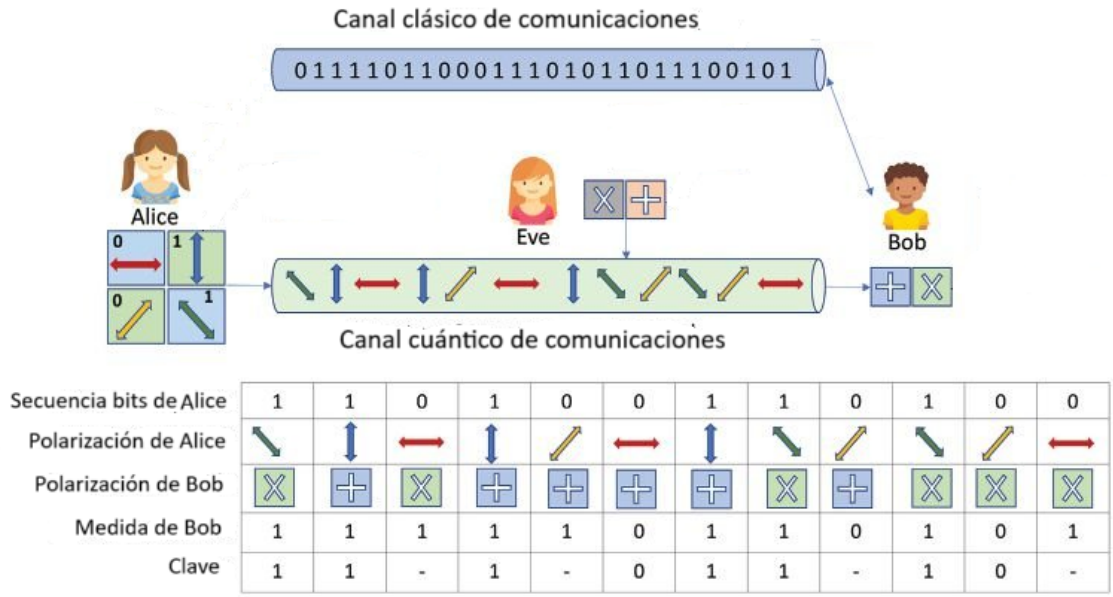

Figura 2.5: Ilustración BB84

La Tabla 2.1 ilustra un ejemplo concreto del protocolo BB84, donde Alice transmite la secuencia de bits "0010010110" codificados en estados cuánticos utilizando

bases aleatorias. Bob mide los estados también de forma aleatoria. Como se observa, únicamente en aquellos casos en los que ambas partes utilizaron la misma base de codificación y medición (por ejemplo, los bits 1, 4, 5, 8 y 9), el resultado de la medición es coherente con el mensaje original, permitiendo su incorporación a la clave bruta. Los casos restantes son descartados por haber utilizado bases distintas, perdiendo la información del qubit. Este ejemplo refleja cómo, incluso en ausencia de un espía, las propiedades estadísticas del protocolo implican que solo una fracción de los bits transmitidos son útiles, destacando la necesidad del paso de depuración posterior antes de obtener una clave secreta final segura.

No. de bit	1	2	3	4	5	6	7	8	9	10
Mensaje	0	0	1	0	0	1	0	1	1	0
Eje Alice	X	Z	X	X	Z	X	Z	Z	X	Z
Estado enviado	$\|+\rangle$	$\|1\rangle$	$\|-\rangle$	$\|+\rangle$	$\|0\rangle$	$\|+\rangle$	$\|1\rangle$	$\|1\rangle$	$\|-\rangle$	$\|0\rangle$
Eje Bob	X	X	Z	X	Z	Z	X	Z	X	X
Estado recibido	$\|+\rangle$	$\|-\rangle$	$\|1\rangle$	$\|+\rangle$	$\|0\rangle$	$\|0\rangle$	$\|-\rangle$	$\|1\rangle$	$\|-\rangle$	$\|+\rangle$
¿Válido?	Sí	No	No	Sí	Sí	No	No	Sí	Sí	No

Cuadro 2.1: Ejemplo del protocolo BB84: comparación de bases y estados.

Como se ha mencionado, un potencial espía, Eve, no puede observar ni medir los estados cuánticos sin perturbarlos debido al teorema de no clonación y al principio de incertidumbre. No obstante, puede intentar interceptar los fotones y medirlos, actuando como si fuera Bob. Para ello, debe elegir aleatoriamente una base para medir cada fotón. Luego, tras medir, intentará reenviar un nuevo fotón a Bob en el estado que cree que fue enviado por Alice.

Este proceso introduce inevitablemente errores. Cuando Eve elige una base distinta a la utilizada por Alice, su medición colapsa el estado cuántico de forma aleatoria. En consecuencia, cuando Bob mide ese fotón (reenviado incorrectamente por Eve) con la base correcta, puede obtener un resultado erróneo respecto al bit original. Este tipo de interferencia se manifiesta como un aumento del QBER, el cual puede ser detectado durante la fase de verificación.

No. de bit	1	2	3	4	5	6	7	8	9	10
Mensaje	0	0	1	0	0	1	0	1	1	0
Eje Alice	X	Z	X	X	Z	X	Z	Z	X	Z
Estado enviado	$\|+\rangle$	$\|1\rangle$	$\|-\rangle$	$\|+\rangle$	$\|0\rangle$	$\|+\rangle$	$\|1\rangle$	$\|1\rangle$	$\|-\rangle$	$\|0\rangle$
Eje Eve	Z	X	Z	X	X	Z	X	Z	Z	Z
Medición Eve	$\|1\rangle$	$\|+\rangle$	$\|1\rangle$	$\|+\rangle$	$\|+\rangle$	$\|1\rangle$	$\|-\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$
Eje Bob	X	X	Z	X	Z	Z	X	Z	X	X
Estado recibido	$\|-\rangle$	$\|-\rangle$	$\|1\rangle$	$\|+\rangle$	$\|0\rangle$	$\|1\rangle$	$\|+\rangle$	$\|1\rangle$	$\|+\rangle$	$\|+\rangle$
¿Válido?	No	No	Sí	Sí	Sí	No	No	Sí	No	No

Cuadro 2.2: Protocolo BB84 con presencia de un espía

Si el QBER supera un umbral predeterminado (por ejemplo, un 11 %), Alice y Bob concluyen que la clave ha sido comprometida y abortan el proceso. En caso contrario,

si el error es aceptablemente bajo, aplican técnicas de corrección de errores (como el protocolo *Cascade*) y de reducción de información disponible para Eve (*privacy amplification*), obteniendo así una clave secreta segura.

Capítulo 3 Distribución cuántica de claves y sistemas cuánticos: Estado del arte

La investigación y el desarrollo en tecnologías cuánticas atraviesan una etapa de intensa actividad, impulsada tanto por avances experimentales como por iniciativas de estandarización. Aunque la computación cuántica de propósito general aún está lejos de superar de forma sostenida a las supercomputadoras clásicas, el progreso en hardware, protocolos de comunicación y criptografía post-cuántica refleja un panorama en rápida evolución. La transición hacia un ecosistema de comunicaciones seguras en la era cuántica dependerá de la convergencia entre estos tres ejes.

Actualmente, la mayoría de los dispositivos cuánticos se encuentran en la denominada era NISQ (*Noisy Intermediate-Scale Quantum*), caracterizada por procesadores con un número limitado de qubits y una elevada tasa de errores. Si bien estos sistemas son funcionales, carecen de la robustez y la capacidad de escalado necesarias para un despliegue masivo. La extrema fragilidad de los estados cuánticos exige condiciones físicas muy controladas para evitar la pérdida de coherencia, lo que convierte el desarrollo de hardware en uno de los principales cuellos de botella.

Aun con estas limitaciones, empresas como IBM, Google, Amazon, IonQ, Rigetti, Xanadu o Quantinuum han logrado prototipos con decenas y cientos de qubits, alcanzando hitos técnicos que, aunque específicos, sientan las bases para futuras aplicaciones prácticas. El reto ahora es pasar de demostraciones puntuales de "ventaja cuántica" a sistemas versátiles, escalables y resistentes a fallos.

Por ejemplo, la Figura 3.1 muestra la hoja de ruta de IBM y sus planes de desarrollo a futuro.

3.1 Arquitecturas hardware de referencia

El desarrollo de qubits estables y escalables es el núcleo de la carrera cuántica. Entre las arquitecturas más relevantes destacan los siguientes:

En primer lugar, los qubits superconductores. Estos se basan en circuitos superconductores operando a temperaturas criogénicas, con el transmon como diseño predominante. Ofrecen alta fidelidad (superior al 99,9 %), compatibilidad con procesos

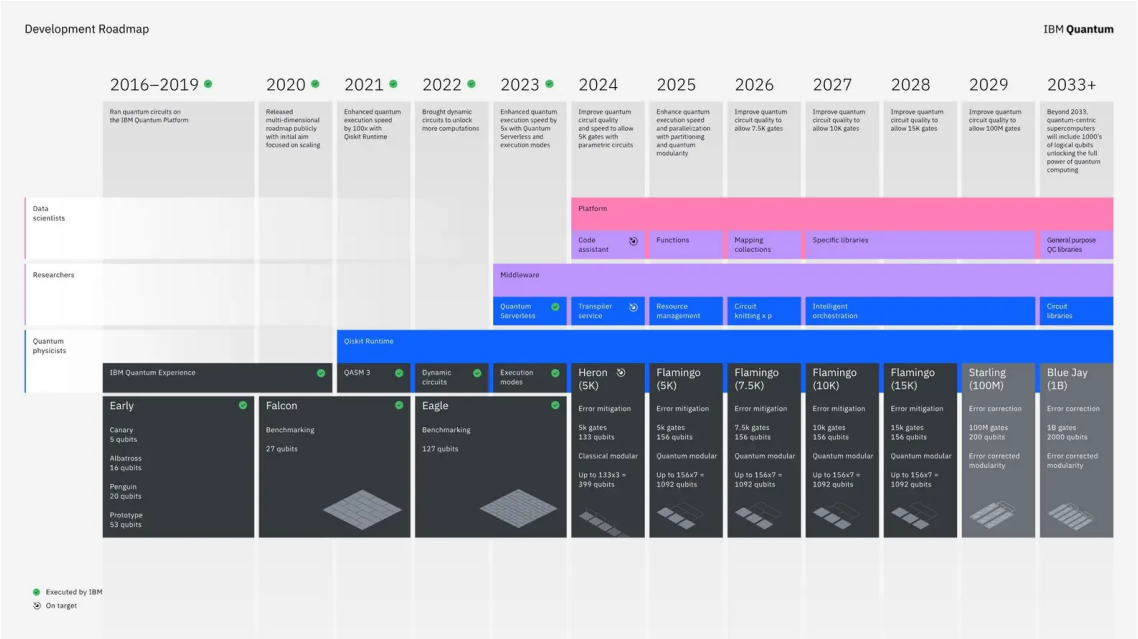

Figura 3.1: Hoja de ruta de IBM

CMOS y velocidades de operación en nanosegundos. No obstante, requieren refrigeración extrema (milikelvins) y presentan tiempos de coherencia limitados. IBM Figura 3.1 [21] lidera este campo con su procesador Condor de 1121 qubits [22], mientras Google y Rigetti [23] trabajan en arquitecturas modulares para mejorar la escalabilidad.

Qubits de iones atrapados. Utilizan campos electromagnéticos y pulsos láser para manipular iones individuales. Destacan por su fidelidad superior al 99,99 % y tiempos de coherencia de varios minutos, aunque las operaciones son más lentas y el escalado a gran número de qubits sigue siendo complejo. IonQ y Quantinuum apuestan por arquitecturas modulares y aplicaciones en química cuántica y optimización [24].

Por otro lado, los qubits fotónicos codifican información en estados de fotones, resistentes a la decoherencia y operativos a temperatura ambiente. Son ideales para redes cuánticas, aunque la implementación de compuertas universales requiere interacciones no lineales complejas. Empresas como Xanadu (con su sistema Borealis) y PsiQuantum exploran chips fotónicos integrados en silicio [25].

Finalmente, los qubits topológicos basados en partículas de Majorana, prometen corrección de errores intrínseca mediante codificación no local. Aún en fase experimental, Microsoft impulsa esta línea con su proyecto *StationQ* y el transistor cuántico topológico. Su validación supondría un avance disruptivo al simplificar drásticamente los requisitos de corrección de errores [26].

No obstante, conviene cuestionarse hasta qué punto estos logros representan avances reales y no meras estrategias de marketing corporativo orientadas a generar expectación, proyectar una imagen más innovadora y, en última instancia, obtener un rédito económico.

3.2 Estándares FIPS Post-Cuánticos

La llegada de la computación cuántica amenaza con quebrar los cimientos de la criptografía clásica basada en problemas como la factorización (RSA) o el logaritmo discreto (ECC). En respuesta, el NIST ha estandarizado algoritmos resistentes a ataques cuánticos, publicados en los *Federal Information Processing Standards* (FIPS) 203, 204 y 205 [27].

El FIPS 203 define el estándar para *Cryptographic Suite for Algebraic Lattices – Kyber* (CRYSTALS-Kyber) (*Module-Lattice Key Encapsulation Mechanism* (ML-KEM)), un esquema de encapsulación de claves basado en el problema *Learning With Errors* (LWE) y su variante en retículas modulares. Este algoritmo combina eficiencia, seguridad y compacidad, y está llamado a sustituir intercambios clásicos como RSA o Diffie-Hellman en entornos críticos.

Por otro lado, el FIPS 204 estandariza *Cryptographic Suite for Algebraic Lattices – Dilithium* (CRYSTALS-Dilithium) (*Module-Lattice Digital Signature Algorithm* (ML-DSA)), un esquema de firma digital sobre retículas estructuradas que ofrece un equilibrio entre tamaño de clave, velocidad y robustez frente a ataques cuánticos, con aplicaciones en autenticación de software y comunicaciones seguras.

Finalmente, el FIPS 205 establece *Stateless Lattice-based Hash Digital Signature Algorithm* (SLH-DSA) (*Stateless Practical Hash-based Incredibly Nice Cryptographic Signature Plus* (SPHINCS+)), un esquema de firma basado únicamente en funciones hash seguras, sin necesidad de estado interno. Destaca por su resistencia a fallos de implementación y su fuerte fundamentación teórica, lo que lo hace atractivo para entornos de alta criticidad.

A diferencia de la QKD, estos estándares no emplean tecnología cuántica para protegerse de ataques provenientes de ella, sino que recurren a estrategias criptográficas clásicas diseñadas para afrontar este desafío.

3.3 Estado actual de la QKD

La QKD se perfila como una de las tecnologías más prometedoras para el establecimiento de claves secretas con seguridad informacionalmente teórica, independiente del poder computacional presente o futuro. Esta propiedad, conocida como *everlasting security*, es clave en sectores donde la confidencialidad debe preservarse durante décadas. En la práctica, la implementación de QKD enfrenta retos como vulnerabilidades de canal lateral (*side-channel attacks*), que explotan imperfecciones de hardware. Ejemplos incluyen ataques de luz brillante, de tipo caballo de Troya, emisión de múltiples fotones y *back-flash* en detectores. Para mitigarlos, se han desarrollado técnicas como la *Privacy Amplification*, los estados señuelo (*decoy states*) o el protocolo MDI-QKD [28].

La estandarización de QKD avanza de la mano de organismos como *European Telecommunications Standards Institute* (ETSI), que a través de su *Industry Speci-*

fication Group on QKD (ISG-QKD) establece requisitos técnicos y de certificación. Estos esfuerzos buscan garantizar que los sistemas comerciales cumplan niveles de seguridad adecuados incluso frente a ataques avanzados.

Algunos de estos protocolos estandarizados son: el protocolo E91 (Ekert, 1991) [29] emplea pares de fotones entrelazados y la violación de desigualdades de Bell para garantizar la seguridad, mientras que Bennett, Brassard y Mermin 1992 (BBM92) [30] es una variante de BB84 que también utiliza entrelazamiento en lugar de estados de polarización individuales. Por otro lado, *Differential Phase Shift QKD* (DPS-QKD) [31] codifica la información en la diferencia de fase entre pulsos consecutivos, lo que lo hace menos sensible a ciertos tipos de ruido. Por su parte, el CV-QKD [32] transmite información cuántica mediante cuadraturas del campo electromagnético, aprovechando la tecnología de detección homodina y heterodina.

También existen otros no estandarizados como el protocolo B92 (Bennett, 1992) [33] que simplifica el esquema BB84 utilizando solo dos estados cuánticos, lo que reduce la complejidad pero también la tasa de clave segura. Scarani, Acín, Ribordy y Gisin 2004 (SARG04) [34] es una variante diseñada para mejorar la resistencia frente a ataques de discriminación de estados, manteniendo compatibilidad con la infraestructura de BB84. El *Twin-Field QKD* (TF-QKD) [35] permite alcanzar distancias mucho mayores sin repetidores cuánticos mediante interferencia de pulsos débiles en un nodo intermedio.

El *Round-Robin Differential Phase Shift* (RR-DPS) [36] mejora la seguridad frente a ataques de memoria cuántica al aumentar el número de fases posibles en la codificación. En estos ataques Eve intercepta un fotón y lo guarda sin perturbarlo, con dicha memoria podrá esperar hasta que Alice y Bob revelen públicamente ciertos parámetros del protocolo (por ejemplo, la base de codificación usada) y entonces medirlo en la base correcta, evitando los errores que delatarían su presencia. Es una amenaza seria si las memorias cuánticas alcanzan suficiente fidelidad y tiempo de almacenamiento, aunque actualmente la tecnología es muy limitada.

Por último, el *floodlight QKD*[37] emplea un ancho de banda muy elevado y pulsos multicanal para incrementar la tasa de clave segura, aunque aún se encuentra en fase experimental.

En cuanto a despliegues reales, se han alcanzado distancias superiores a 400 km sobre fibra óptica y se han realizado demostraciones satelitales pioneras. Se han registrado tasas de clave de hasta 115,8 *Mbps* sobre 10 km de fibra óptica estándar [38] y en sistemas de CV-QKD multicarrier (OFDM-CV-QKD) se alcanzaron hasta 1,78 *Gbps* a 5 km, 1,03 *Gbps* a 10 km y varios *Mbps* incluso a 100 km [39]. En cuanto a las distancias, se ha alcanzado transmisión QKD de hasta 403 km sobre fibra estándar con protocolos como Mode-Pairing QKD, superando ampliamente el límite sin repetidores [40] y se realizaron experimentos reales con enlaces urbanos de más de 410 km entre Bristol y Cambridge [41].

Respecto al plano espacial, el satélite chino *Micius*, lanzado en 2016, estableció enlaces QKD de más de 7.600 km entre China y Austria, así como teletransportación cuántica asistida por satélite [42], [43]. En marzo de 2025, el microsatélite *Jinan-1* permitió un enlace de 12.900 km entre China y Sudáfrica, marcando un paso hacia

una constelación de microsatélites cuánticos y una red global de comunicaciones seguras mediante distribución cuántica de claves [44].

Capítulo 4 Realización del experimento

El desarrollo del experimento se puede dividir en cuatro fases diferenciadas, tanto técnica como cronológicamente. En primer lugar, se realizó la preparación del entorno de desarrollo, lo que incluyó la instalación de las librerías necesarias y la familiarización con las herramientas utilizadas. A continuación, se procedió al diseño de circuitos y algoritmos cuánticos mediante el uso de puertas cuánticas, con el objetivo de representar el problema de forma eficiente. Posteriormente, se llevaron a cabo simulaciones y ejecuciones sobre hardware cuántico, cuando estuvo disponible, para validar el funcionamiento del diseño. Finalmente, se realizó el procesado y análisis de los datos obtenidos, con el fin de evaluar el rendimiento y la precisión de los resultados experimentales.

4.1 Entorno de desarrollo

El entorno de simulación y la conexión remota con el hardware cuántico se ha construido sobre el framework Qiskit de IBM [45]. Qiskit es un conjunto de herramientas y bibliotecas en Python que proporciona un entorno completo para la definición, simulación y ejecución de circuitos cuánticos. Internamente, emplea el módulo Aer para simulaciones de alto rendimiento [46] y el IBM Quantum Provider para la ejecución en procesadores reales a través de la nube [47].

Como editor y plataforma interactiva de desarrollo se utilizó Visual Studio Code combinado con Jupyter Notebook [48], [49]. Esta configuración permite escribir y depurar código en Python, intercalar explicaciones y ecuaciones en LaTeX, y visualizar resultados de forma instantánea. Además, VSCode ofrece extensiones para linting, autocompletado y control de versiones, mientras que Jupyter facilita la creación de documentos reproducibles y su compartición como informes ejecutables.

Dada la rápida evolución de Qiskit y sus dependencias (con actualizaciones frecuentes que pueden cambiar la sintaxis de funciones o incluso eliminar módulos) resulta imprescindible gestionar con cuidado las versiones del entorno. Para ello se emplea Miniconda3 [50] como gestor de entornos, lo que permite aislar proyectos y fijar versiones específicas de Python, Qiskit y librerías auxiliares. A modo de ejemplo, la secuencia mínima para crear y activar un entorno compatible sería:

Código 4.1: Creación del entorno con Miniconda

```
conda create -n qc-env python=3.9 -y
conda activate qc-env
```

```
pip install qiskit
pip install matplotlib scipy jupyterlab
```

Para congelar el estado del entorno y facilitar su reproducibilidad en otros equipos, basta con exportar un archivo YAML:

Código 4.2: Exportar configuración de Conda

```
conda env export --name qc-env --file environment.yml
```

Y luego, en un nuevo sistema:

Código 4.3: Recrear entorno desde YAML

```
conda env create --file environment.yml
conda activate qc-env
```

Con este flujo de trabajo, se asegura que cualquier colaborador o máquina de implementación disponga exactamente de las mismas versiones de Python, Qiskit y librerías auxiliares, evitando errores de incompatibilidad y manteniendo la reproducibilidad de los experimentos.

Por otro lado, para el aprendizaje y uso adecuado de estas herramientas, se aprovechó la amplia documentación que ofrece IBM, complementada con recursos educativos como tutoriales y vídeos disponibles en YouTube [51], lo que permitió un proceso de aprendizaje rápido y efectivo para dominar la programación en este entorno cuántico.

4.2 Código

En cuanto al código, está almacenado en un archivo con extensión .ipynb, característico de Jupyter Notebook. Este archivo se estructura en dos partes principales: funciones y código llamador. Las funciones son las responsables de generar la estructura de los circuitos y definir los procesos a ejecutar, mientras que el código llamador utiliza estas funciones para operar con datos concretos y realizar simulaciones o interacciones con el hardware cuántico.

Las funciones contenidas en el archivo se pueden clasificar según su propósito. Entre ellas, se encuentran funciones para la generación de circuitos cuánticos, incluyendo aquellos destinados a procesos clave como recepción, transmisión, interceptación (espía) y las conexiones entre estos. Además, hay funciones dedicadas a la ejecución de los circuitos, tanto mediante simuladores locales como a través de hardware cuántico remoto. Por último, se incluyen funciones para el tratamiento y análisis de los datos, que facilitan la interpretación de los resultados obtenidos.

Esta estructura organizada del archivo permite una separación clara entre la lógica del sistema (definida en las funciones) y su ejecución práctica (controlada desde el código llamador), facilitando la comprensión, mantenimiento y futuras mejoras del proyecto.

En conjunto, disponer de Qiskit para la programación y ejecución a nivel cuántico, Jupyter Notebook para la documentación ejecutable y Miniconda para el control de versiones constituye un entorno sólido y flexible, capaz de adaptarse a las continuas actualizaciones de la plataforma y de facilitar tanto la simulación local como el acceso remoto a procesadores cuánticos reales.

4.2.1 Diseño de circuitos

Como se indicó previamente, la interacción con los qubits en un circuito cuántico se articula mediante puertas cuánticas. En el esquema que nos ocupa se emplean únicamente las puertas Hadamard (H) y Pauli-X (X). Junto a ellas, resultan esenciales los registros cuánticos y clásicos, los operadores de medida (por defecto en la base computacional), los condicionales clásicos, que permiten aplicar una puerta solo si se verifica una condición previa, y las barreras (`barriers`), útiles para separar y clarificar etapas lógicas del circuito.

El circuito se organiza en tres bloques funcionales aislados y bien definidos: el transmisor (Alice), que codifica un mensaje aleatorio; el receptor (Bob), que intenta decodificarlo; y el posible atacante (Eve), que trata de interceptar la comunicación. Esta separación facilita el análisis y la trazabilidad de los efectos de cada operación sobre el estado cuántico.

4.2.1.1. Generador cuántico de valores aleatorios

Para que BB84 funcione correctamente se requieren bits verdaderamente aleatorios e independientes, tanto para elegir la base de preparación como para fijar el valor lógico a transmitir. En este trabajo la aleatoriedad se obtiene a partir de un circuito mínimo: cada qubit se inicializa en $|0\rangle$, se le aplica una puerta Hadamard para crear la superposición de los estados $|0\rangle$ y $|1\rangle$ en el estado $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ y, finalmente, se mide en la base computacional $\{|0\rangle, |1\rangle\}$. La medida colapsa el estado con probabilidad 1/2 en cada resultado, generando así un bit equiprobable. El Código 4.4 recoge la implementación y la Figura 4.1 ilustra el circuito resultante.

Código 4.4: Aleatorizador en BB84

```
def randomizer(nqbits):
    from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
    # ______ Registros ____________________________________________________
    # Registro cuantico
    qreg_randValue = QuantumRegister(nqbits, 'qRegRand') # qbit ayuda
    # Registro clasico
    creg_rand  = ClassicalRegister(nqbits, 'cRegRand')   # arrayGenerado
    # ______ Circuito _____________________________________________________
    qc = QuantumCircuit(qreg_randValue, creg_rand, name='randomizer')
    # ______ Gen aleatoria ________________________________________________
    qc.reset(qreg_randValue)
    qc.h(qreg_randValue)     # clave
    qc.measure(qreg_randValue, creg_rand)
```

```
    return qc, creg_rand
```

En el código anterior, las funciones de Qiskit `QuantumRegister` y `ClassicalRegister` definen, respectivamente, los registros cuánticos y clásicos donde residen los qubits y los bits medidos; su tamaño viene determinado por `nqbits`. La clase `QuantumCircuit` construye el circuito que orquesta las operaciones sobre ambos registros y permite encapsularlo con un nombre identificativo para su reutilización. La llamada `qreg_randValue` fuerza el estado inicial $|0\rangle$ en todos los qubits objetivo, garantizando un punto de partida limpio y reproducible. A continuación, `h(qreg_randValue)` aplica la puerta de Hadamard para crear superposiciones uniformes, fundamento de la generación de aleatoriedad cuántica. Finalmente, `measure(qreg,creg)` proyecta los qubits en la base computacional y vuelca los resultados en el registro clásico, desde donde pueden extraerse como bits aleatorios. La función retorna el propio circuito y el registro clásico de salida, lo que facilita su composición con el resto de módulos del protocolo.

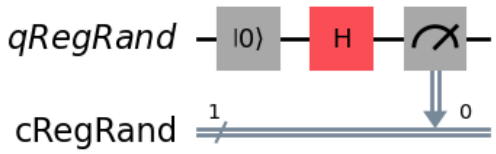

Figura 4.1: Circuito aleatorizador

4.2.1.2. Transmisor

El Código 4.5 implementa la lógica del emisor (Alice) y la Figura 4.2 representa el circuito cuántico resultante, *qc*. La función `BB84TX(nqbits)` construye un circuito con tres registros cuánticos: el de transmisión `qTRX`, el generador de bits de clave `qRandTX` y el generador de bases `qPolTX`, además de dos registros clásicos para almacenar los resultados de medida, `cKeyTX` y `cPolTX`. Tras un *reset* explícito, se crea aleatoriedad cuántica como en el apartado anterior y se generan dos cadenas clásicas independientes y equiprobables: la clave k_{TX} y la base b_{TX}.

Código 4.5: Transmisor en BB84

```
def BB84TX(nqbits):
    from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
    # ______ Registros ____________________________________________________
    # Registros cuanticos
    qreg_TRX    = QuantumRegister(nqbits, 'qTRX')
    qreg_randValue = QuantumRegister(nqbits, 'qRandTX')
    qreg_polTX  = QuantumRegister(nqbits, 'qPolTX')
    # Registros clasicos
    creg_keyTX  = ClassicalRegister(nqbits, 'cKeyTX')
    creg_polTX  = ClassicalRegister(nqbits, 'cPolTX')
```

```
    # ______ Circuito ____________________________________________________
    qc = QuantumCircuit(qreg_TRX, qreg_randValue, qreg_polTX,
                        creg_keyTX, creg_polTX, name='BB84_TX')
    # ______ Inicializacion ______________________________________________
    qc.reset(qreg_TRX)
    qc.reset(qreg_randValue)
    qc.reset(qreg_polTX)
    # ______ Gen aleatoria _______________________________________________
    qc.h(qreg_randValue)     # clave
    qc.measure(qreg_randValue, creg_keyTX)
    qc.h(qreg_polTX)      # base
    qc.measure(qreg_polTX, creg_polTX)
    # ______ Codificacion condicional____________________________________
    for i in range(nqbits):
        with qc.if_test((creg_keyTX[i], 1)):
            qc.x(qreg_TRX[i]) # Si el bit de clave es 1 -> aplicar X

        with qc.if_test((creg_polTX[i], 1)):
            qc.h(qreg_TRX[i]) # Si la base es 1 -> aplicar H

    return qc, qreg_TRX, qreg_randValue, qreg_polTX, creg_keyTX, creg_polTX
```

La codificación de cada símbolo se realiza mediante control clásico: si $k_{\mathrm{TX}}[i] = 1$ se aplica X sobre $qTRX[i]$, y si $b_{\mathrm{TX}}[i] = 1$ se aplica H sobre ese mismo qubit. Con esta regla, cuando $b_{\mathrm{TX}}[i] = 0$ (base Z) el estado preparado es $|0\rangle$ o $|1\rangle$ según la clave, y cuando $b_{\mathrm{TX}}[i] = 1$ (base X) el estado preparado es $|+\rangle$ o $|-\rangle$. Finalmente, Alice conserva los *strings* clásicos de base y clave para el proceso de cribado y posterior corrección de errores.

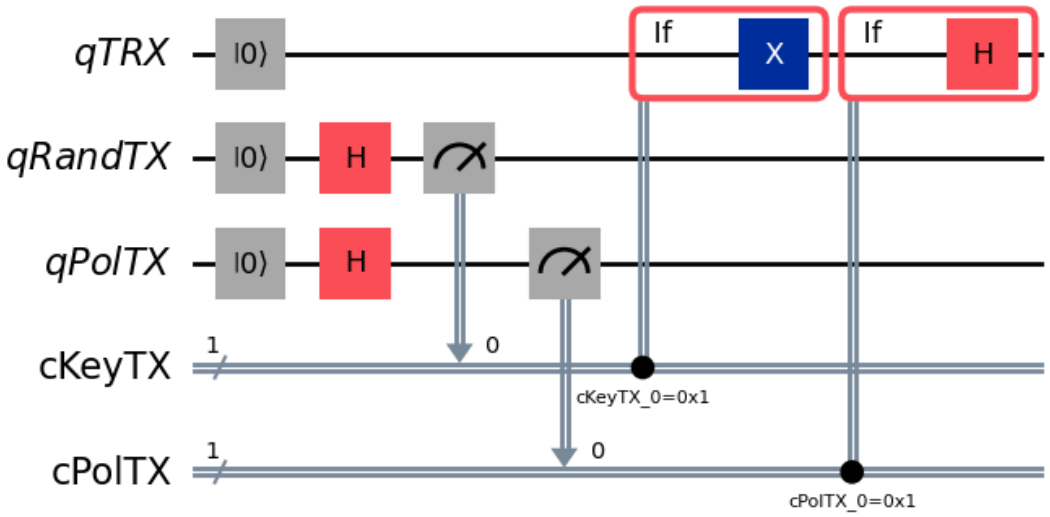

Figura 4.2: Transmisor cuántico

4.2.1.3. Receptor

La estructura del receptor se ilustra en la Figura 4.3. Siguiendo el procedimiento del protocolo BB84, Bob recibe los qubits preparados por Alice y los mide en bases elegidas al azar, tal y como implementa el Código 4.6. La función construye un circuito que, además del registro entrante `qTRX`, incorpora un registro cuántico auxiliar para generar aleatoriedad local (`qRandRX`) y dos registros clásicos para almacenar, respectivamente, la base elegida y el resultado de las medidas (`cPolRX` y `cKeyRX`). Tras una barrera que separa la transmisión de la decodificación, se resetea `qRandRX`, se aplica Hadamard y se mide en la base computacional para obtener una cadena binaria equiprobable que determina la base de medida de cada qubit recibido.

La decodificación se realiza con control clásico por posición: cuando `cPolRX[i]=1`, se aplica una puerta Hadamard sobre `qTRX[i]` antes de medir; esto equivale a medir en la base X. En caso contrario, se mide directamente en Z. Finalmente, se mide `qTRX` en `cKeyRX` y la función devuelve el circuito y los registros implicados para su uso posterior en el cribado y la validación de bases.

Dado que la base de Alice es desconocida para Bob, la elección aleatoria de base implica que, en promedio, la mitad de las posiciones se medirán en una base distinta a la de preparación. Esas posiciones producen resultados no informativos y se descartan durante el cribado. En las posiciones con coincidencia de bases, el resultado debería coincidir idealmente con el bit preparado; cualquier discrepancia residual se atribuye al ruido del canal o a la presencia de un atacante, y se cuantifica mediante el QBER.

Código 4.6: Receptor en BB84

```
def BB84RX(qreg_TRX):
    from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
    nqbits = len(qreg_TRX)
    # ______ Registros ____________________________________________________
    qreg_polarizationRX = QuantumRegister(nqbits, 'qRandRX')  # rand RX
    creg_keyRX = ClassicalRegister(nqbits, 'cKeyRX')
    creg_polarizationRX = ClassicalRegister(nqbits, 'cPolRX')
    # ______ Circuito _____________________________________________________
    qc = QuantumCircuit(qreg_TRX, qreg_polarizationRX, creg_keyRX,
   creg_polarizationRX, name="BB84_RX")
    # ______ Inicializacion _______________________________________________
    qc.barrier()
    qc.reset(qreg_polarizationRX)
    qc.h(qreg_polarizationRX)
    qc.measure(qreg_polarizationRX, creg_polarizationRX)
    # ______ Decodificacion condicional ___________________________________
    for i in range(nqbits):
        with qc.if_test((creg_polarizationRX[i], 1)):
            qc.h(qreg_TRX[i])
    qc.measure(qreg_TRX, creg_keyRX)

    return qc, qreg_TRX, qreg_polarizationRX, creg_keyRX,
   creg_polarizationRX
```

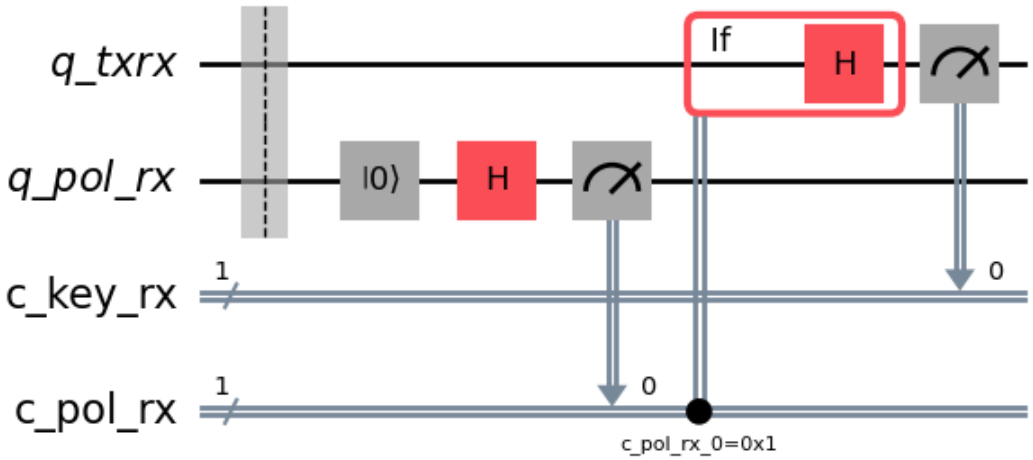

Figura 4.3: Receptor cuántico

4.2.1.4. Espía

En el protocolo BB84, Eve implementa un ataque de interceptación y reenvío para intentar extraer información de la clave. El Código 4.7 recoge la implementación empleada y la Figura 4.4 muestra la estructura del circuito del espía. La función `espia(qreg_data)` recibe el registro cuántico en tránsito y crea un registro auxiliar de aleatoriedad (`qRandSpy`) para decidir, bit a bit, la base de medida; además, reserva registros clásicos para almacenar la base elegida y el resultado de sus mediciones (`cPolSpy` y `cKeySpy`). Tras una barrera que separa la fase de transmisión de la intervención del atacante, aplica Hadamard sobre `qRandSpy` y mide en la base computacional para obtener una secuencia binaria equiprobable que fija la base de medida de Eve.

La decodificación se realiza bajo control clásico: si `cPolSpy[i]=1`, el espía aplica una puerta H previa sobre `qreg_data[i]` para medir efectivamente en la base X; en caso contrario, mide directamente en Z. Una vez obtenida su lectura, el espía vuelve a preparar el qubit para reenviarlo a Bob: cuando su base fue X, aplica de nuevo H (*ocultación de pruebas*) para devolver el estado a la base original del canal. Este procedimiento implementa el ataque de *intercept–resend*: cuando la base de Eve coincide con la de Alice, el bit reenviado suele conservarse; cuando no coincide, el colapso introduce perturbaciones que se traducen en errores detectables tras el cribado, elevando el QBER de la muestra pública.

Código 4.7: Circuito del espía en BB84

```
def espia(qreg_data):
    from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
    nqbits = len(qreg_data)
    # ______ Registros ____________________________________________________
    qreg_polarizationEspia = QuantumRegister(nqbits, 'qRandSpy')  # base
    creg_keyEspia = ClassicalRegister(nqbits, 'cKeySpy')
    creg_polarizationEspia = ClassicalRegister(nqbits, 'CPolSpy')
```

```
    # ______ Circuito ____________________________________________________
    qc = QuantumCircuit(qreg_data, qreg_polarizationEspia,
                        creg_keyEspia, creg_polarizationEspia,
                        name="BB84_Spy")
    # ______ Inicializacion ______________________________________________
    qc.barrier()
    qc.reset(qreg_polarizationEspia)
    qc.h(qreg_polarizationEspia)
    qc.measure(qreg_polarizationEspia, creg_polarizationEspia)
     # ______ Decodificacion condicional _________________________________
    for i in range(nqbits):
        with qc.if_test((creg_polarizationEspia[i], 1)):
            qc.h(qreg_data[i])
    qc.measure(qreg_data, creg_keyEspia)
    # ______ Ocultacion de pruebas _______________________________________
    for i in range(nqbits):
        with qc.if_test((creg_polarizationEspia[i], 1)):
            qc.h(qreg_data[i])

    return qc, qreg_data, qreg_polarizationEspia, creg_keyEspia,
    creg_polarizationEspia
```

En suma, la intervención del espía es operacionalmente análoga a la del receptor, con la diferencia crucial de que su medida interrumpe el canal y fuerza un colapso intermedio. La posterior comparación pública de una fracción de los bits permite a Alice y Bob estimar el QBER y, en su caso, inferir la presencia de Eve.

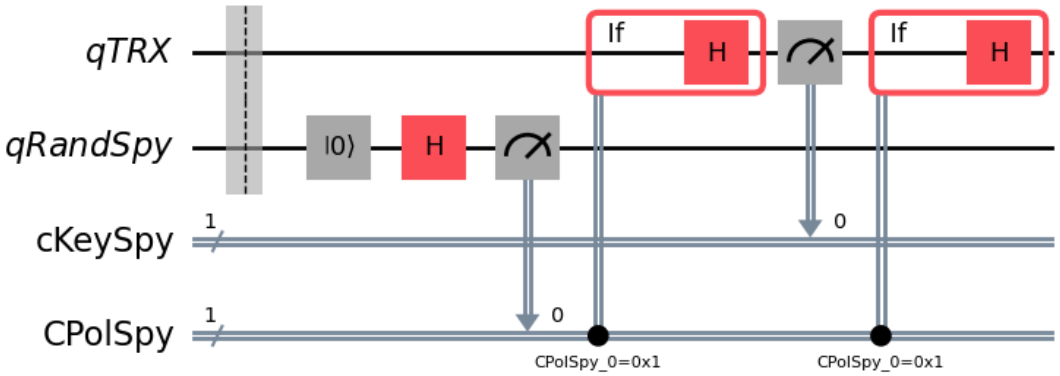

Figura 4.4: Espía cuántico

4.2.1.5. Circuito sin espía

Para construir el circuito completo sin intruso se ensamblan los circuitos del transmisor y del receptor compartiendo exclusivamente el canal cuántico `qTRX`. La Código 4.8 muestra la rutina de construcción y la Figura 4.5 ilustra el esquema resultante. La función `BB84sinEspia(nqbits)` ejecuta primero los módulos `BB84TX` y `BB84RX` tomando como nexo el registro `qTRX`; a continuación crea un circuito maes-

tro que agrega los registros cuánticos y clásicos de ambos subcircuitos y, mediante `compose` con `inplace=True`, acopla el flujo de operaciones de transmisión y medida sobre el mismo canal. De este modo, las cadenas clásicas de bases y claves (`cPolTX`, `cKeyTX`, `cPolRX`, `cKeyRX`) quedan disponibles para el cribado y la estimación posterior del QBER.

Código 4.8: Circuito completo sin espía

```
def BB84sinEspia(nqbits): #combinacion de circuito TX y RX
    from qiskit import QuantumCircuit
    # ______ Registros ____________________________________________________
    circTX, qrTRX, qrRand, qrPolTX, crKeyTX, crPolTX = BB84TX(nqbits)
    circRX, qrTRX, qrPolRX, crKeyRX, crPolRX = BB84RX(qrTRX)
    # ______ Circuito _____________________________________________________
    combinedCircuit = QuantumCircuit(qrTRX, qrRand, qrPolTX, qrPolRX,
                                     crKeyTX, crPolTX, crKeyRX, crPolRX,
                                     name = "BB84SE")
    # ______ Acoplado de circuitos ________________________________________
    combinedCircuit.compose(circTX, inplace=True)
    combinedCircuit.compose(circRX, qubits=[qrTRX[0], qrPolRX[0]],
                            clbits=[crKeyRX[0], crPolRX[0]], inplace=True)

    return circuito_combinado
```

En esta implementación, `compose` se utiliza para posicionar el subcircuito del receptor detrás del transmisor manteniendo `qTRX` como hilo conductor del canal cuántico; los registros auxiliares de aleatoriedad en el transmisor (`qRandTX`, `qPolTX`) y en el receptor (`qRandRX`) permanecen locales a cada bloque, garantizando el desacoplo entre la preparación y la medida salvo por el canal compartido.

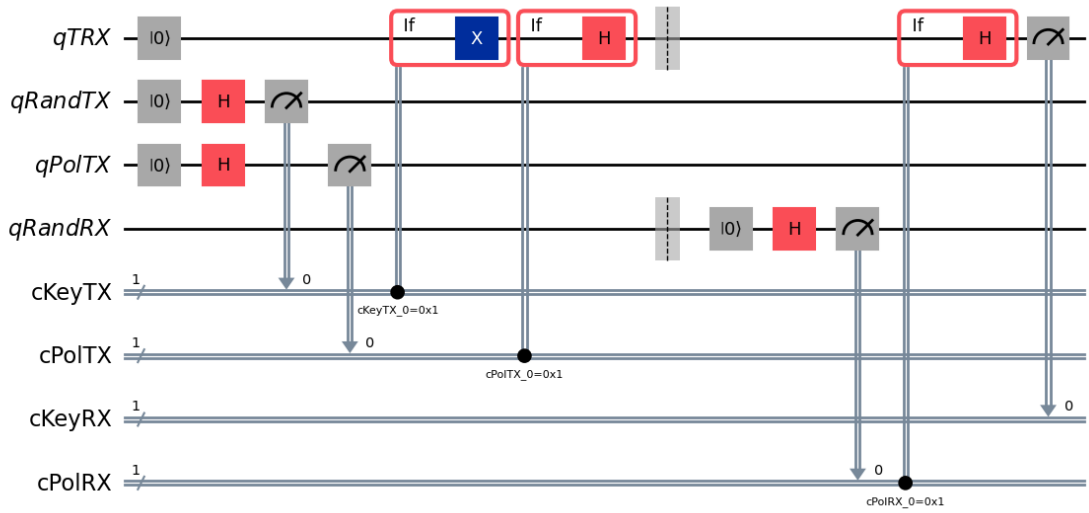

Figura 4.5: Circuito completo sin espía

4.2.1.6. Circuito con espía

El escenario con intruso es idéntico al anterior salvo por el módulo de espía que se introduce entre Alice y Bob. Este módulo espía intercepta y reenvía la información por el canal cuántico `qTRX`. Tal y como se muestra en la función `BB84conEspia`, Código 4.9, que detalla la construcción y la Figura 4.6 que muestra el circuito resultante.

Código 4.9: Circuito completo con espía

```
def BB84conEspia(nqbits): #combinacion de circuito TX, espia y RX
    from qiskit import QuantumCircuit
    # ______ Registros ____________________________________________________
    circuitTX, qrTRX, qreg_rand, qrPolTX, crKeyTX, crPolTX = BB84TX(nqbits)
    circuitSpy, qrTRX, qrPolSpy, crKeySpy, crPolSpy = espia(qrTRX)
    circuitRX, qrTRX, qrPolRX, crKeyRX, crPolRX = BB84RX(qrTRX)
    # ______ Circuito _____________________________________________________
    combCirc = QuantumCircuit(qrTRX, qreg_rand, qrPolTX, qrPolRX, qrPolSpy,
                              crKeyTX, crPolTX, crKeyRX, crPolRX, crKeySpy,
                              crPolSpy, name = "BB84CE")
    # ______ Acoplado de circuitos ________________________________________
    combCirc.compose(circuitTX, inplace=True)
    combCirc.compose(circuitSpy, qubits=[qrTRX[0], qrPolSpy[0]],
                     clbits=[crKeySpy[0], crPolSpy[0]], inplace=True)
    combCirc.compose(circuitRX, qubits=[qrTRX[0], qrPolRX[0]],
                     clbits=[crKeyRX[0], crPolRX[0]], inplace=True)

    return combCirc
```

Un circuito integrado que modela el ataque *intercept–resend*: Eve mide en una base aleatoria y, en función de su elección, vuelve a preparar y reenvía el qubit por el canal. Esta intervención colapsa el estado intermedio y, cuando su base no coincide con la de Alice, introduce discrepancias que aumentan el QBER observable por Alice y Bob al comparar una fracción pública de sus resultados.

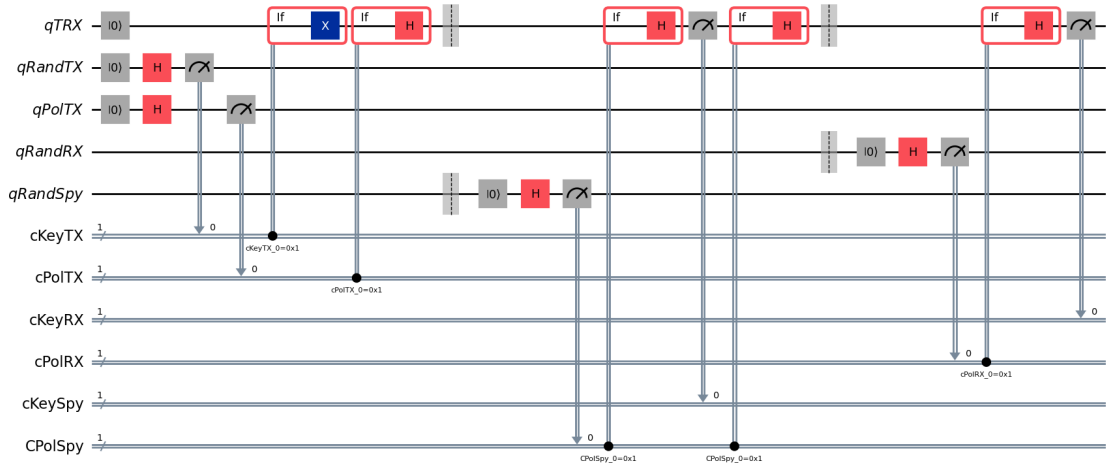

Figura 4.6: Circuito completo con espía

4.2.2 Simulación y resultados

Para la realización de las pruebas experimentales se utiliza *Aer*, la librería de IBM que permite simular circuitos cuánticos tanto en condiciones ideales como incorporando modelos de ruido que aproximan el comportamiento del hardware real. Al ejecutarse de forma local, el uso de *Aer* abarata y acelera las pruebas, ya que evita las solicitudes remotas y las colas de los servidores de IBM.

El simulador admite la inclusión de parámetros y modelos de ruido específicos. En una primera fase, con el objetivo de validar el protocolo, resulta conveniente trabajar en un régimen ideal y con un número elevado de disparos (*shots*), de manera que las frecuencias observadas se aproximen a las teóricas (apoyándose en la ley de los grandes números). A continuación se muestran, Código 4.10 y Código 4.11, las dos rutinas de simulación empleadas: una ideal y otra con ruido extraído de un backend de IBM, respectivamente.

Código 4.10: Simulador ideal

```
def simulator(circuito, qbitslanzados):
    from qiskit_aer import Aer
    from qiskit_ibm_runtime import SamplerV2
    # ______ Backend ____________________________________________________
    myBackend = Aer.get_backend("qasm_simulator")
    # ______ Preparacion y ejecucion ____________________________________
    mySampler = SamplerV2(myBackend)
    job = mySampler.run([circuito], shots=qbitslanzados)

    return job.result()
```

La función `simulator` selecciona el backend local `qasm_simulator` de *Aer* y ejecuta el circuito con `SamplerV2` indicando el número de disparos deseado (`qbitslanzados`). El retorno es el objeto de resultados del muestreo, a partir del cual se obtienen los conteos y probabilidades necesarios para estimar magnitudes como el QBER en un entorno sin ruido (esto facilita comprobar que la implementación reproduce el comportamiento esperado del protocolo).

Código 4.11: Simulador con ruido, IBM Sherbrooke

```
def realistic_simulator(circuito,qbits):
    from qiskit_ibm_runtime import QiskitRuntimeService, SamplerV2
    from qiskit_aer import AerSimulator
    from qiskit_aer.noise import NoiseModel
    # ______ Llamada a qiskit servers ___________________________________
    myService = QiskitRuntimeService()
    # ______ Backend con ruido __________________________________________
    backend = myService.backend('ibm_sherbrooke')
    noise_model = NoiseModel.from_backend(backend)
    sim = AerSimulator(
        noise_model=noise_model,
        coupling_map=backend.configuration().coupling_map,
        basis_gates=noise_model.basis_gates
    )
```

```
    # ______ Preparacion y ejecucion ______________________________________
    sampler = SamplerV2(sim)
    job = sampler.run([circuito], shots=qbits)

    return job.result()
```

Por su parte, `realistic_simulator` construye un *AerSimulator* parametrizado con un `NoiseModel` generado a partir del backend `ibm_sherbrooke`. Con ello se incorporan errores de compuerta y de lectura, así como las restricciones físicas del dispositivo (*coupling map* y puertas nativas) para obtener una aproximación más fiel al hardware. El resultado devuelto por `SamplerV2` permite comparar el rendimiento ideal frente al realista y analizar el impacto del ruido en el QBER y en la tasa de clave obtenida (esta comparación es la base de los experimentos que se presentan en este apartado).

4.2.2.1. Simulación sin espía

En el escenario ideal sin intrusos, la Figura 4.7 muestra un ejemplo de la distribución de resultados para 10^6 transmisiones. Cada barra codifica, en ese orden, la base empleada por el transmisor, el bit transmitido, la base elegida por el receptor y el bit medido finalmente (esta convención permite reconocer las coincidencias y discrepancias entre preparación y medida).

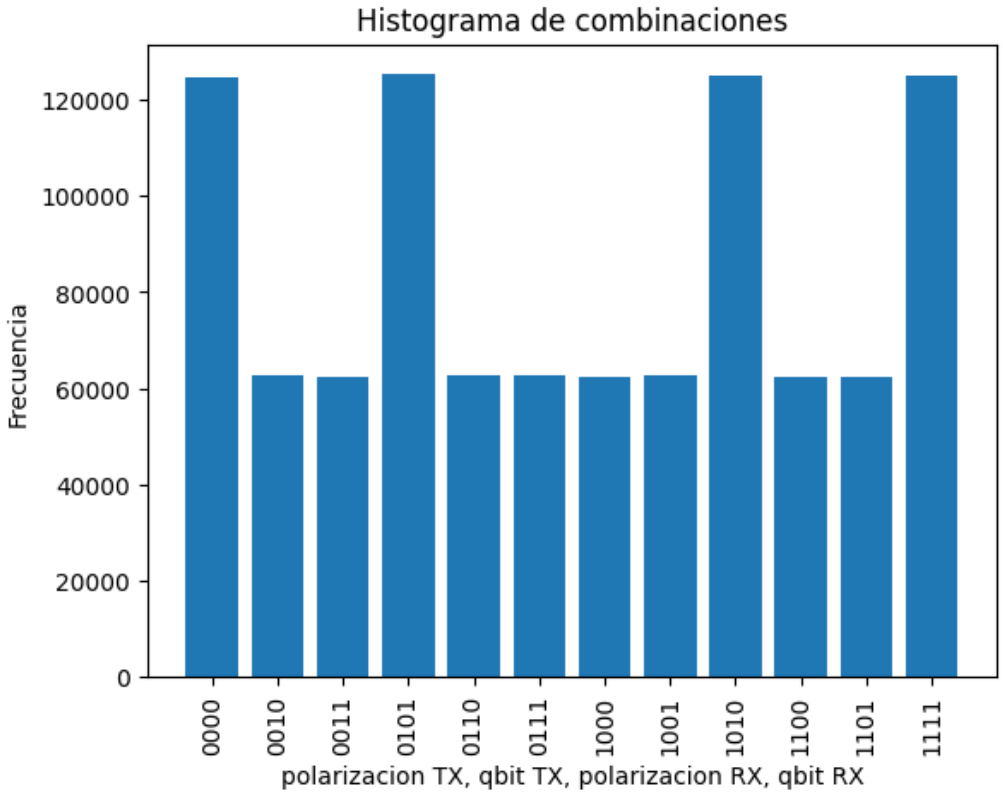

Figura 4.7: Histograma de simulación de 10^6 transmisiones sin espía

Cuando las bases de Alice y Bob coinciden, el resultado de Bob reproduce el bit transmitido por Alice y estas combinaciones suman aproximadamente el 50 % del total (la elección de bases es equiprobable). En cambio, cuando las bases no coinciden, la medida de Bob es aleatoria (la proyección en una base incompatible colapsa el estado y el bit observado coincide o no con igual probabilidad), de modo que estas combinaciones representan también alrededor del 50 %. En conjunto, la simulación refleja el comportamiento teórico del protocolo: cerca de la mitad de los registros se descartan por bases distintas y la otra mitad constituye candidatos a clave tras el cribado (esta proporción se aproxima al 50/50 al aumentar el número de disparos por la ley de los grandes números).

4.2.2.2. Simulación con espía

Con un espía activo (Eve) que aplica una estrategia de intercepción y reenvío, cada qubit es interceptado, medido en una base aleatoria y reenviado al receptor (esta intervención introduce perturbaciones adicionales en el canal). Si la base de Eve no coincide con la de Alice, el estado reenviado no está alineado con la preparación original y, cuando Bob mide en la base correcta, aparece un error con probabilidad 1/2. Manteniendo bases equiprobables, aproximadamente el 50 % de los registros se descarta por no coincidir las bases de Alice y Bob, mientras que el 50 % restante constituye la clave bruta. En ese subconjunto útil, el error esperado es 25 % (si la base de Eve coincide con la de Alice no hay error y si no coincide se erra la mitad de las veces), por lo que, sobre el total, se observan en torno a $37,5\,\%$ de coincidencias válidas y 12,5 % de errores (por ejemplo, unos 375.000 aciertos y 125.000 errores sobre 10^6 transmisiones).

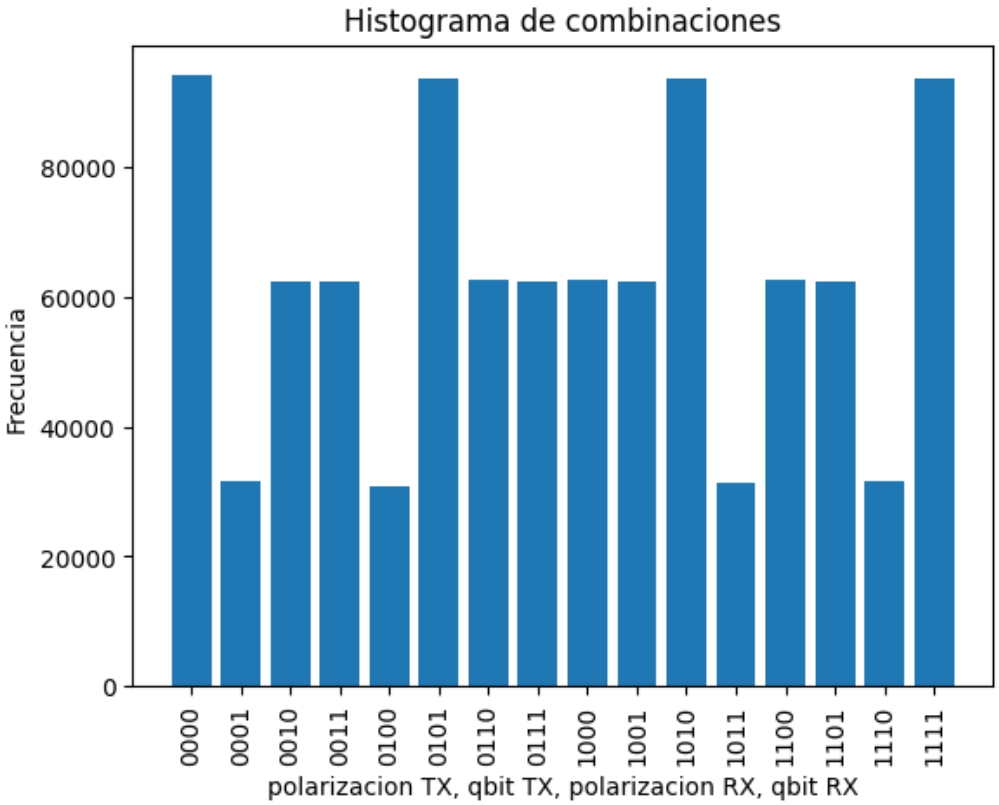

Figura 4.8: Histograma de simulación de 10^6 transmisiones con espía

Estos errores, (combinaciones que no deberían aparecer como "0001", "0100", etc. donde las bases coinciden pero la clave transmitida no) sólo se manifiestan cuando las bases coinciden, esto es, sobre la clave cribada, que típicamente abarca la mitad de los casos. solo se manifiestan cuando las bases de Alice y Bob coinciden (es decir, dentro de la clave cribada). Por esa razón, la QBER se mide respecto al tamaño de la clave tras haber descartado los bits de las posiciones donde la base difiere. En dispositivos reales, además de un posible atacante, contribuyen fuentes adicionales como decoherencia, desalineaciones y ruido de lectura, que elevan la QBER observada respecto al caso ideal).

4.2.3 Ejecución en hardware real

Para la ejecución en hardware cuántico real se utiliza la librería de acceso a IBM Quantum (IBM Quantum Provider), que permite enviar los mismos circuitos empleados en las simulaciones y obtener resultados experimentales. A efectos de reproducibilidad se conservan las métricas operativas más relevantes del backend empleado (esto facilita interpretar las desviaciones respecto al caso ideal sin necesidad de introducir supuestos adicionales).

En este trabajo se ha utilizado el procesador ibm_sherbrooke, ubicado en Washington DC, con 127 qubits. La mediana del error de ECR es $7{,}801 \times 10^{-3}$, la de `SX` es $2{,}37 \times 10^{-4}$ y el error de lectura tiene una mediana de $1{,}587 \times 10^{-2}$. El dispositivo

se basa en la arquitectura *Eagle r3* y reporta un rendimiento aproximado de 150.000 CLOPS. En estas ejecuciones se pierde el comportamiento ideal debido al ruido del canal físico y a la estadística finita de disparos (cuando el número de repeticiones no es muy grande, la ley de los grandes números no garantiza la convergencia de frecuencias y los histogramas pueden variar apreciablemente entre lanzamientos).

4.2.3.1. Ejecución sin espía

En la Figura 4.9 se observa que la distribución experimental de resultados no reproduce exactamente ni la simulación ideal sin espía ni la simulación con espía. La estructura parece más ruidosa (con ciertas tendencias), y aparecen combinaciones que en el régimen ideal se considerarían errores. Estos eventos se explican por la QBER intrínseca del dispositivo y del proceso de lectura (esto dificulta la detección de un atacante basándose únicamente en desviaciones pequeñas). En el caso mostrado se realizaron 10^3 disparos, por lo que la variabilidad muestral añade dispersión adicional a las frecuencias observadas.

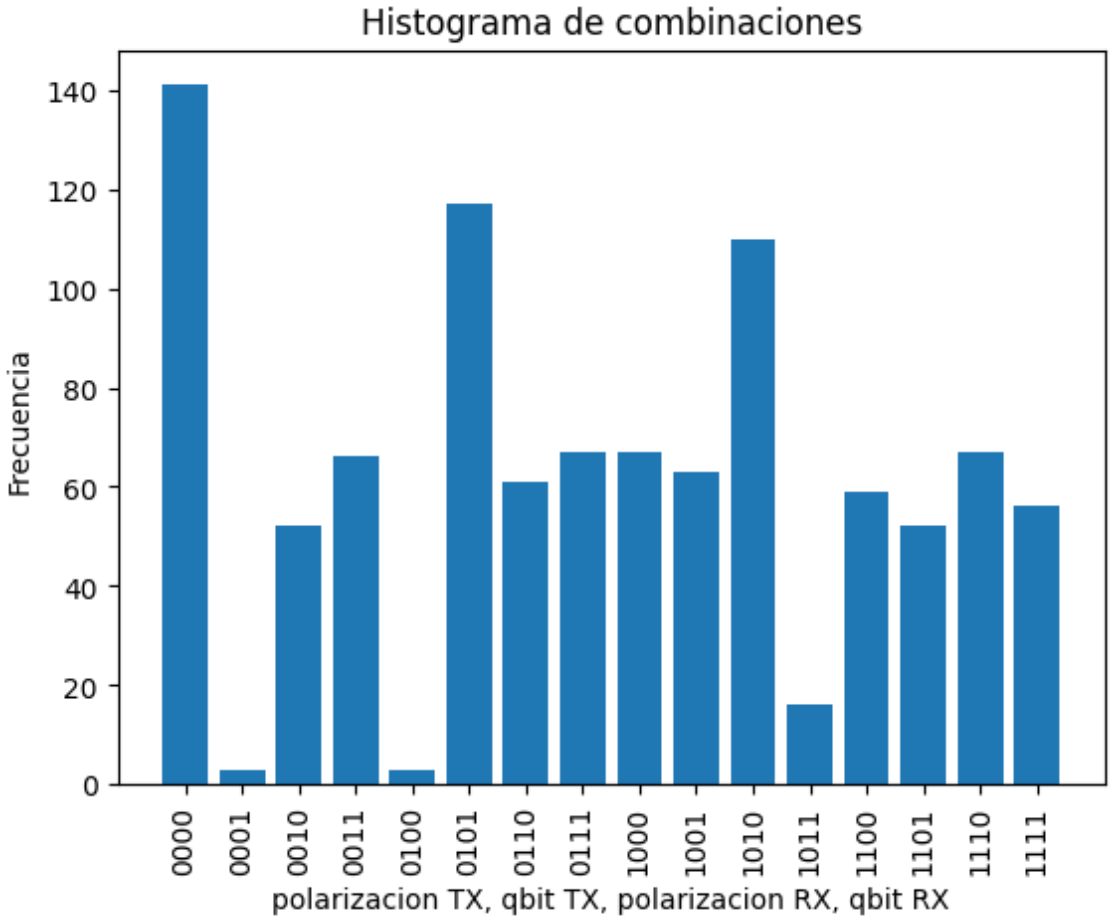

Figura 4.9: Histograma de mil ejecuciones remotas sin espía

4.2.3.2. Ejecución con espía

Finalmente, se ejecutó el circuito con espía en la plataforma remota, obteniéndose el histograma de la Figura 4.10. En este escenario la tasa de errores aumenta de manera apreciable respecto a la ejecución sin espía (se incrementa la proporción de combinaciones incompatibles con la preparación cuando las bases de Alice y Bob coinciden), lo que proporciona una señal operativa de la intervención de Eve. En la práctica, la cuantificación del incremento de QBER debe interpretarse teniendo en cuenta tanto las imperfecciones del hardware como la estadística finita de disparos (esto evita atribuir al atacante fluctuaciones que pueden deberse al dispositivo).

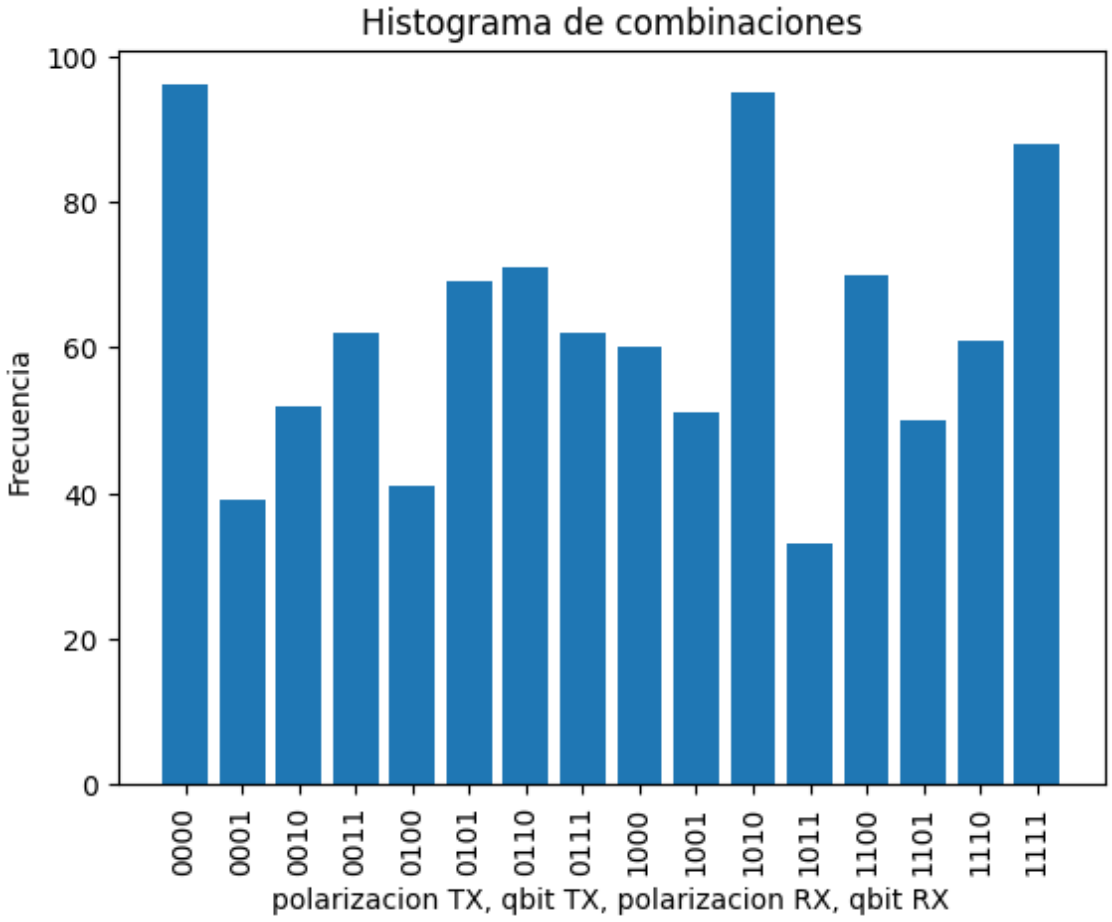

Figura 4.10: Histograma de mil ejecuciones remotas con espía

4.2.4 Tratamiento de los datos

El tratamiento de los datos se apoya en tres funciones clave que actúan de forma secuencial para depurar y evaluar la calidad de la clave intercambiada. La primera de ellas, `dataExtract` (recogida en la Código 4.12), sirve como punto de partida para el análisis. Esta función recibe el resultado bruto de la ejecución del circuito y extrae, a partir de su estructura interna, las bases de transmisión y recepción junto con las claves brutas de Alice y Bob. Para ello, accede al primer elemento de la lista

de resultados (`result[0]`), y utiliza métodos específicos para obtener las cadenas de bits asociadas a la clave transmitida (`cKeyTX`), la clave recibida (`cKeyRX`), la base de transmisión (`cPolTX`) y la base de recepción (`cPolRX`). El retorno consiste en estas cuatro secuencias, que serán procesadas por las funciones siguientes.

Código 4.12: Función dataExtract

```
def dataExtract(result):
    pub_result = result[0]
    keyTX = pub_result.data.cKeyTX.get_bitstrings()
    keyRX = pub_result.data.cKeyRX.get_bitstrings()
    baseTX = pub_result.data.cPolTX.get_bitstrings()
    baseRX = pub_result.data.cPolRX.get_bitstrings()
    return baseTX, keyTX, baseRX, keyRX
```

La segunda función, `siftingKey` (mostrada en la Código 4.13), toma como entrada las bases y claves obtenidas en el paso anterior. Su propósito es filtrar únicamente aquellos bits en los que Alice y Bob hayan medido con la misma base, ya que son los únicos candidatos a formar parte de la clave final. Para ello, primero convierte las secuencias de bits en arrays booleanos, lo que facilita las operaciones lógicas posteriores. A continuación, recorre todas las posiciones y aplica un operador XOR entre las bases de transmisión y recepción; un resultado cero indica que ambas coinciden en la orientación de medida. En ese caso, los bits correspondientes de Alice y Bob se copian en dos vectores (`filter_alice` y `filter_bob`) que se mantienen compactos mediante un índice independiente. Al final, se recortan ambos vectores a la longitud efectiva y se devuelven como salida, conteniendo únicamente los bits válidos para la generación de la clave.

Código 4.13: Función siftingKey

```
def siftingKey(baseTX, alice, baseRX, bob):
    import numpy as np
    # ______ Conversion a booleanos ________________________________________
    alice_arr  = np.array(alice, dtype='U1')
    bob_arr    = np.array(bob,   dtype='U1')
    alice_bool = (alice_arr.astype(int).astype(bool))
    bob_bool   = (bob_arr.astype(int).astype(bool))
    # ______ Validaciones __________________________________________________
    if bob_bool.size != alice_bool.size:
        raise ValueError("Alice and Bob arrays must have the same length")
    # ______ Inicializaciones ______________________________________________
    filter_alice = np.zeros(len(baseTX), dtype=bool)
    filter_bob   = np.zeros(len(baseTX), dtype=bool)
    index2: int  = 0
    # ______ Bucle principal _______________________________________________
    for i in range(len(baseTX)):  # xor en python = ^
        anal_base = int(baseTX[i]) ^ int(baseRX[i])  # se comprueba si
    coincide la polarizacion
        if anal_base == 0:
            filter_alice[index2] = alice_bool[i]
            filter_bob[index2]   = bob_bool[i]
            index2 = index2 + 1
```

```
    # ______ Post-proceso y retorno ____________________________________
    filter_alice = filter_alice[0:index2]
    filter_bob   = filter_bob[0:index2]
    return filter_alice, filter_bob
```

Por último, la función `qberAnalisis` (descrita en la Código 4.14) realiza una evaluación exhaustiva de los errores en la clave resultante del cribado. Recibe las cuatro secuencias originales (bases y claves de ambas partes) y analiza únicamente las posiciones donde las bases coinciden. En cada una de estas posiciones, efectúa un XOR entre los bits de clave; si el resultado es distinto de cero, registra la discrepancia en `array_detection` y aumenta el contador de errores detectados. Si, por el contrario, los bits coinciden, se añaden al vector `key`, que acumula la porción libre de errores de la clave. Finalizado el recorrido, se calcula el QBER como el cociente entre el número total de discrepancias y el número de posiciones evaluadas, multiplicado por cien para obtener un porcentaje. Esta información permite cuantificar de forma directa la calidad del enlace y sirve como criterio para decidir la viabilidad de la clave.

Código 4.14: Función qberAnalisis

```
def qberAnalisis(baseTX, keyTX, baseRX, keyRX):
    import numpy as np
    # ______ Validaciones ___________________________________________
    if not (len(baseTX) == len(baseRX) == len(keyTX) == len(keyRX)):
        raise ValueError("Los vectores deben tener la misma longitud")
    # ______ Inicializaciones _______________________________________
    array_detection = np.full(len(baseTX), None, dtype=object)
    key             = np.full(len(baseTX), None, dtype=object)
    detecciones     = 0
    index           = 0
    key_counter     = 0
    # ______ Bucle principal ________________________________________
    for i in range(len(baseTX)):
        # xor en python = ^
        anal_base = int(baseTX[i]) ^ int(baseRX[i])    # se comprueba si
   coincide la polarizacion
        if anal_base == 0:
            key_counter = key_counter + 1
            anal_key = int(keyTX[i]) ^ int(keyRX[i])   # se comprueba si no
    coincide la clave
            if anal_key != 0:
                detecciones += 1
                array_detection[i] = keyTX[i]
            else:
                key[index] = keyTX[i]
                index += 1
    # ______ Post-proceso ___________________________________________
    porcentaje = (detecciones / key_counter) * 100
    key = [x for x in key if x is not None]
    return array_detection, key, porcentaje  # porcentaje total
```

4.2.5 Corrección de errores con CASCADE

Las gráficas de barras previas describen el protocolo desde la perspectiva de un observador que conoce ambas claves (permite verificar divergencias de forma directa). En la práctica esa comparación en claro no es viable: es necesario reconciliar la clave bruta para corregir los errores del canal sin exponer información. Para ello se emplea *CASCADE*, un protocolo interactivo que corrige discrepancias mediante rondas sucesivas de comprobaciones de paridad y búsquedas binarias. La implementación utilizada se organiza en tres componentes principales, referenciados en el Código 4.15 (función principal), el Código 4.16 (búsqueda binaria) y la Código 4.17 (paridad por bloques). La Figura 4.11 muestra el funcionamiento típico del protocolo.

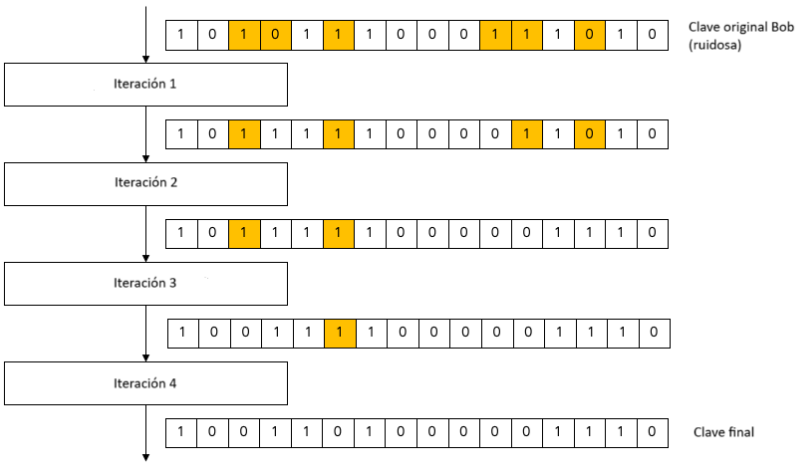

Figura 4.11: Esquema de las iteraciones

Código 4.15: Función principal protocolo CASCADE

```
def cascade_bb84(alice, bob, block_size: int = 8, cascade_iterations:
                 int = 4, seed: int = 1234, verbose: bool = False):
    import numpy as np
    # ______ Conversion a booleanos ______________________________________
    alice = np.asarray(alice, dtype=bool)
    bob   = np.asarray(bob,   dtype=bool)
    n_orig = alice.size
    # ______ Padding hasta multiplo de block_size ________________________
    pad_len = (-n_orig) % block_size
    if pad_len:
        pad   = np.zeros(pad_len, dtype=bool)
        alice = np.concatenate([alice, pad])
        bob   = np.concatenate([bob,   pad])
    # ______ QBER inicial (clave bruta) __________________________________
    qber_rawKey = (np.sum(alice ^ bob) / n_orig) * 100
    # ______ Inicializaciones ____________________________________________
    ncorrections = 0
    rng          = np.random.default_rng(seed)
    # ______ Rondas de Cascade ___________________________________________
```

```
    for r in range(cascade_iterations):
        bsz = block_size * (2**r)
        if bsz > alice.size:
            break
        # ______ Permutacion global determinista _________________________
        perm = rng.permutation(alice.size)
        inv  = np.argsort(perm)
        A    = alice[perm].copy()
        B    = bob[perm].copy()
        # ______ Bucle interno (pasadas) _________________________________
        changed = True
        while changed:
            changed = False
            for start in range(0, A.size, bsz):
                end   = start + bsz
                a_seg = A[start:end]
                b_seg = B[start:end]
                # ______ Paridad distinta, error impar ___________________
                if not parityCheck(a_seg, b_seg):
                    flips = binary_search(a_seg, b_seg)   # ______ Binaria
                    if flips is not None:
                        B[start:end] = flips
                        ncorrections += 1
                        changed      = True
        # ______ Deshacer permutacion de la ronda ________________________
        alice = A[inv]
        bob   = B[inv]
    # ______ Recorte al tamano original _____________________________________
    alice = alice[:n_orig]
    bob   = bob[:n_orig]
    err_rate = (np.sum(alice ^ bob) / n_orig) * 100
    # ______ Salida detallada (verbose) _____________________________________
    if verbose:
        if err_rate == 0:
            print("CASCADE was succesfull")
            print("QBER raw key:", qber_rawKey)
            print("Key =")
            print(alice)
        else:
            print("CASCADE failed")
            print("QBER raw key:", qber_rawKey)
            print("Alice =")
            print(alice)
            print("Bob =")
            print(bob)

    return alice, bob, ncorrections
```

Como se aprecia en la Código 4.15, `cascade_bb84` recibe las secuencias de Alice y Bob, las convierte a booleanos y aplica *padding* si la longitud no es múltiplo de `block_size` (esto permite dividir en bloques sin resto). Calcula la QBER inicial

sobre la clave bruta y recorre `cascade_iterations` rondas con tamaño de bloque $b_{\text{sz}} = \texttt{block\_size} \cdot 2^r$.

En cada ronda baraja las posiciones mediante una permutación determinista controlada por `seed`, procesa todos los bloques y, cuando la paridad difiere, delega la localización del error en la búsqueda binaria de la Código 4.16. Tras estabilizar la ronda (no hay más cambios), deshace la permutación, recorta el *padding* al final del proceso y calcula la tasa residual `err_rate` (en modo `verbose` imprime además métricas útiles).

Código 4.16: Busqueda binaria

```
def binary_search(a_seg: np.ndarray, b_seg: np.ndarray) -> np.ndarray |
    None:
    # ______ Copias locales _____________________________________________
    a = a_seg
    b = b_seg.copy()
    # ______ Busqueda binaria ___________________________________________
    left  = 0
    right = a.size
    while right - left > 1:
        mid = (left + right) // 2
        # La discrepancia de paridad esta en la mitad izquierda?
        if not parityCheck(a[left:mid], b[left:mid]):
            right = mid
        else:
            left = mid
    # ______ Correccion del bit _________________________________________
    b[left] = ~b[left]
    return b
```

La Código 4.16 implementa la localización del bit erróneo en segmentos con paridad impar. Divide el bloque por la mitad y consulta la paridad de la submitad izquierda para decidir dónde persiste la discrepancia hasta quedar un único elemento, que se invierte para corregir la diferencia. El número de consultas queda acotado por $\lceil \log_2(b_{\text{sz}}) \rceil$ (esto hace eficiente la corrección incluso con bloques moderados). El esquema de esta función se puede ver en la Figura 4.12.

La comprobación de paridad de la Código 4.17 calcula la suma modular dos de cada segmento. Devuelve `True` cuando las paridades coinciden (número par de discrepancias, posiblemente cero) y `False` si difieren (existe exactamente un error en el segmento), que es la condición que explota *CASCADE* junto con la búsqueda de la Código 4.16. En conjunto, las rutinas enlazadas por la Código 4.15 implementan la reconciliación: permutar para revelar errores, detectar discrepancias por paridad y corregir por búsqueda binaria hasta lograr una clave consistente.

Código 4.17: Comprobación de paridad

```
def parityCheck(alice, bob):
    # True si paridades iguales (sin error impar), False si hay error impar
    return (np.sum(alice) & 1) == (np.sum(bob) & 1)
```

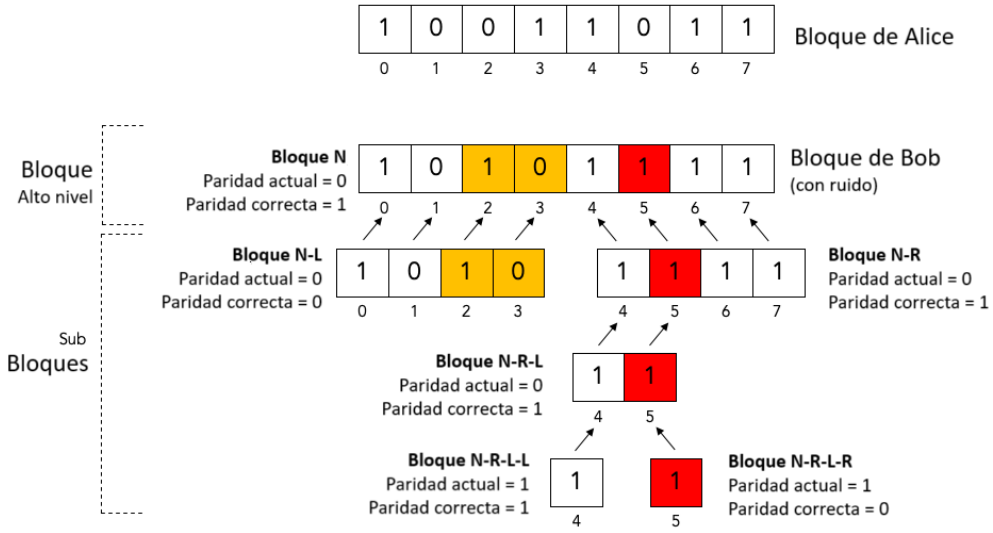

Figura 4.12: Esquema de la búsqueda binaria

4.2.6 Transmisión de clave completa

Para evaluar el comportamiento del protocolo BB84 en un entorno real, se ejecuta una secuencia mínima de funciones que permiten transmitir una clave de 1000 bits en dos configuraciones: sin presencia de espía y con un observador adverso entre Alice y Bob. En ambos casos, la estructura de llamadas es similar, diferenciándose únicamente en la función que genera el circuito cuántico inicial.

En el Código 4.18 se muestra el código para la transmisión sin espía. La primera línea, `BB84sinEspia(1)`, construye un circuito que implementa el esquema ideal de BB84. A continuación, `selectorBackend()` consulta los dispositivos disponibles en la cuenta de IBM Quantum y devuelve el backend con menor cola de espera. Con estos elementos definidos, `ejecutarIBMremoto(circuitoSinEspia, 1000, myBackend)` envía el trabajo con 1000 *shots* al hardware seleccionado, monitorizando el estado de la tarea hasta su finalización y devolviendo el resultado en un objeto *Result* de Qiskit.

Seguidamente, `dataExtract(myResult)` procesa el resultado separando las bases y claves de transmisor y receptor, convirtiéndolas en listas binarias aptas para su tratamiento posterior. La función `qberAnalisis` calcula la QBER de la clave bruta. Si esta es inferior al umbral establecido (en este experimento, 15 %). A continuación, se repitiría la ejecución para obtener la clave definitiva, validando previamente las posiciones con bases coincidentes mediante cribado. Finalmente, `cascade_bb84` aplica el protocolo *Cascade* para reconciliar las claves y corregir posibles errores sin exponer información sensible.

Código 4.18: Ejecución remota sin espía

```
# ____ fase 1 ____________________________________________________________
circuitoSinEspia = BB84sinEspia(1)
myBackend = selectorBackend()
myResult = ejecutarIBMremoto(circuitoSinEspia, 1000, myBackend)
```

```
bTX, kTX, bRX, kRX = dataExtract(myResult)
_,_,qber_data = qberAnalisis(bTX,kTX,bRX,kRX)
print("QBER(%): ",qber_data)
# ____ fase 2 ____________________________________________________________
if qber_data < 15:
    myResult = ejecutarIBMremoto(circuitoSinEspia, 1000, myBackend)
    bTX, kTX, bRX, kRX = dataExtract(myResult)
    kTX, kRX = sifting(bTX, kTX, bRX, kRX)
    alice, bob, otro = cascade_bb84(kTX, kRX)
```

El Código 4.19 muestra un ejemplo de ejecución en el que el backend seleccionado es `ibm_brisbane`, con una QBER inicial cercana al 4 %. En la segunda fase, tras aplicar *Cascade*, las claves de Alice y Bob coinciden y se obtiene una clave reconciliada de 502 bits. La salida incluye la clave final completa y confirma la eficacia del proceso de corrección de errores en condiciones de canal sin intrusiones.

Código 4.19: Resultado de ejecución remota sin espía

```
ibm_brisbane - Trabajos pendientes: 525 - Qubits: 127
ibm_torino - Trabajos pendientes: 4590 - Qubits: 133
El mejor hardware remoto es: ibm_brisbane

QBER raw key:  4.10958904109589

Clave Alice y clave Bob coinciden, QBER raw key:  4.183266932270916
Tamano de la clave:  502

Clave =
0x386E3685992F64376F8342DCF13721616C0833626709DD346C02AF9960E6136981DA3E35F
16F0A47D3E893E4D5DC172AA91ABF59CB10D8EFC8F2F28F5E3EDF
```

En el caso con espía, mostrado en el Código 4.20, la única diferencia en la fase de generación es el uso de `BB84conEspia(1)`, que añade un módulo de interceptación y reenvío (*intercept–resend*) por parte de Eve. El resto de llamadas se mantiene, aunque en este escenario no se aplica el umbral de QBER antes de la corrección, ejecutando *Cascade* de forma directa.

Código 4.20: Ejecución remota con espía

```
# ____ fase 1 ____________________________________________________________
circuitoSinEspia = BB84conEspia(1)
myBackend = selectorBackend()
myResult = ejecutarIBMremoto(circuitoSinEspia, 1000, myBackend)
bTX, kTX, bRX, kRX = dataExtract(myResult)
_,_,qber_data = qberAnalisis(bTX,kTX,bRX,kRX)
print("QBER(%): ",qber_data)
# ____ fase 2 ____________________________________________________________
myResult = ejecutarIBMremoto(circuitoSinEspia, 1000, myBackend)
bTX, kTX, bRX, kRX = dataExtract(myResult)
kTX, kRX = sifting(bTX, kTX, bRX, kRX)
alice, bob, qber = cascade_bb84(kTX, kRX)
```

El Código 4.21 recoge el resultado de este segundo escenario. La QBER inicial se aproxima al 26,45 %, en línea con el valor teórico esperado para el ataque *intercept–resend* sobre bits con bases coincidentes ($\sim 25\,\%$). En esta ejecución, *Cascade* no logra reconciliar las claves, lo que evidencia que a partir de ciertos niveles de error la técnica deja de ser eficaz.

Código 4.21: Resultado de ejecución remota con espía

```
ibm_brisbane - Trabajos pendientes: 525 - Qubits: 127
ibm_torino - Trabajos pendientes: 4591 - Qubits: 133
El mejor hardware remoto es: ibm_brisbane

QBER(%) raw key:  25.9047619047619

CASCADE failed
QBER raw key:  26.446280991735538

Alice =
0xC510DF79D71B3A5DB331E0914DEDD5CD178B6A4B46BC7B273BB6A50868610DFB9859D18F7
27D53DBC1C29FE467E992B0734AB9B4E7E022DE07F2BDE7C0

Bob =
0xC530DF59FF193ADD3331B083CDFDD6CE478A6E4B16BC7BE729B3B5082C710D77B84EDB8D7
27D5193C1C29FE66FE152B0716AD994E74028DED7F2BDE5D0
```

La comparación de ambos escenarios permite cuantificar el impacto de la presencia de un espía en la tasa de error y en la capacidad del protocolo para producir claves secretas coincidentes. En ausencia de intrusión, la QBER es baja y *Cascade* consigue generar claves idénticas, mientras que con un ataque *intercept–resend* la QBER se eleva por encima del umbral práctico de funcionamiento, imposibilitando la reconciliación.

Por otro lado, con presencia de espía y de acuerdo con las llamadas del Código 4.20, la ejecución remota selecciona automáticamente el backend con menor cola y lanza la tarea. El Código 4.21 recoge el registro en consola con los backends disponibles y el elegido, así como el resultado de *Cascade*.

En este escenario resulta esperable un incremento del QBER respecto al caso sin espía; en el experimento se observa que las claves de Alice y Bob no coinciden y que la QBER en la clave cribada es de aproximadamente 26,45 %. Este valor es coherente con el ideal teórico del ataque *intercept–resend* sobre bits con bases coincidentes, cuyo QBER se aproxima a 25 %. La ligera desviación puede atribuirse a la aleatoriedad en la selección de bases, al tamaño muestral de disparos (1000) y al ruido propio del hardware en el momento de la ejecución.

4.2.7 Análisis de rendimiento

Una vez evaluada la transmisión completa de la clave, resulta evidente que el QBER desempeña un papel determinante en la viabilidad del protocolo. A medida que esta tasa de error aumenta, el proceso de reconciliación se enfrenta a un mayor

número de discrepancias, lo que complica la corrección y reduce la probabilidad de obtener una clave final libre de errores. Con el fin de caracterizar de forma cuantitativa esta relación, en este apartado se analiza el rendimiento de *CASCADE* para distintos valores de QBER, evaluando su capacidad de corrección y delimitando el rango operativo óptimo del sistema.

El rendimiento medio del protocolo *CASCADE* en función de la QBER introducida, se presenta en la Figura 4.13. Con el fin de aproximar un escenario realista, se empleó el simulador *Aer* y artificialmente, se introducen discrepancias en la clave. Se realizó el proceso de cribado y se aplicó *CASCADE* sobre la clave resultante. Cada punto de la curva corresponde a la media de múltiples repeticiones independientes (hasta 10^3 ejecuciones por configuración), mientras que las barras reflejan la dispersión de los resultados. Estas barras de error son asimétricas y han sido recortadas para ajustarse al rango físico $[0, 100]\,\%$, evitando así interpretar como valores negativos o superiores al $100\,\%$ fluctuaciones puramente estadísticas.

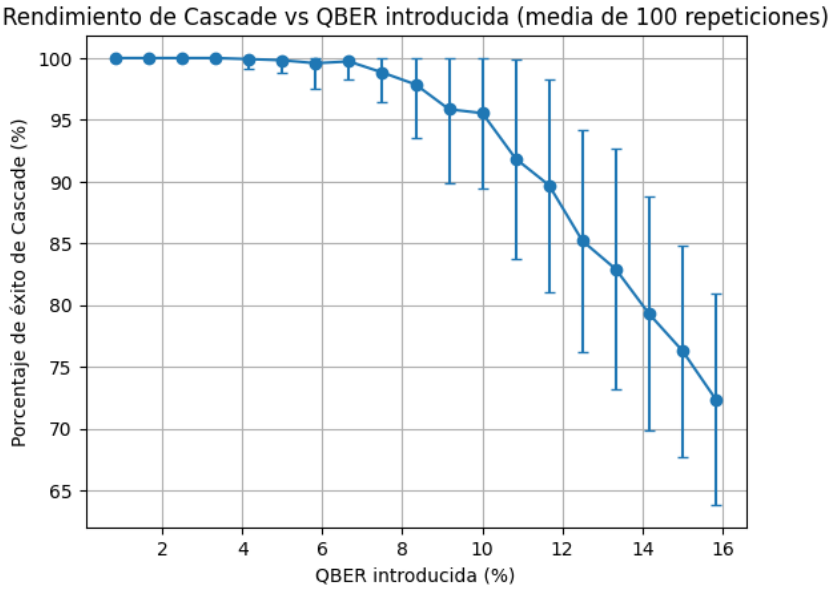

Figura 4.13: Gráfica de rendimiento del protocolo CASCADE

En el eje horizontal se representa la QBER introducida (en %), y en el vertical el porcentaje de éxito de *CASCADE*, definido como el cociente entre los errores corregidos y los errores detectados, normalizado al tamaño total de la muestra (en %). En el régimen de QBER baja (hasta aproximadamente 5–6 %), la tasa de éxito se mantiene próxima al 100 %, con una variabilidad reducida. A partir de valores cercanos a 7 % se observa un descenso gradual acompañado de un incremento en la dispersión entre repeticiones, lo que concuerda con la sensibilidad del protocolo a tasas de error más elevadas y con la probabilidad de que múltiples errores en un mismo bloque queden temporalmente enmascarados por paridades pares. En torno a 9–10 %, la eficacia comienza a degradarse de forma notable, en línea con los umbrales prácticos documentados para *CASCADE*.

Esta caracterización cuantitativa permite determinar el rango operativo óptimo del sistema y anticipar el coste, tanto en número de rondas como en tiempo de

ejecución, necesario para obtener claves reconciliadas a distintos niveles de QBER.

Código 4.22: Resultado de ejecución remota con espía

```
import numpy as np
import matplotlib.pyplot as plt

circuito123 = BB84sinEspia(1)
n = 20
m = n - 1
reps = 100
nqbits = 1000
# Rejilla de QBER teorica (fraccion) para cada i
p_grid = np.arange(1, n) / (6 * n)      # shape (m,)

tasaExito   = np.full((reps, m), np.nan, dtype=float)

for j in range(reps):
    for i, p in enumerate(p_grid):
        result = simulator(circuito123, nqbits)
        bTX, kTX, bRX, kRX = dataExtract(result)
        # _______ inyecta qber artificial _____________________________
        kRX = flip_bits(kRX, p)
        # _______ qber ________________________________________________
        _, _, QBERdetected = postAnalisis(bTX, kTX, bRX, kRX)
        # _______ Sifting + Cascade ___________________________________
        kTX_s, kRX_s = sifting(bTX, kTX, bRX, kRX)
        alice, bob, ncorrect = cascade_bb84(kTX_s, kRX_s)
        # _______ porcentaje exito ____________________________________
        if QBERdetected and QBERdetected != 0:
            tasaExito[j, i] = (ncorrect / QBERdetected) * len(bTX) / 100.0
        else:
            tasaExito[j, i] = 100.0
# __________ media y desviacion _____________________________________
mean_tasa = np.nanmean(tasaExito, axis=0)   # shape (m,)
std_tasa  = np.nanstd(tasaExito,  axis=0)
yerr_lower = np.maximum(0.0, np.minimum(std_tasa, mean_tasa - 0.0))
yerr_upper = np.maximum(0.0, np.minimum(std_tasa, 100.0 - mean_tasa))
# __________ Graficar _________________________________________________
x = p_grid * 100.0
plt.figure()
plt.errorbar(x, mean_tasa, yerr=[yerr_lower, yerr_upper], fmt='-o', capsize
    =3)
plt.xlabel("QBER introducida (%)")
plt.ylabel("Porcentaje de exito de Cascade (%)")
plt.title(f"Rendimiento de Cascade vs QBER introducida (media de {reps}
    repeticiones)")
plt.grid(True)
plt.show()
```

En el código de la Código 4.22, la ejecución repite el proceso de transmisión y corrección de errores para una rejilla de valores de QBER teórica, introduciendo

ruido artificial y aplicando *CASCADE* a la clave resultante. La tasa de éxito se calcula como el porcentaje de errores corregidos respecto al total detectado, y posteriormente se obtiene la media y desviación estándar para graficar el rendimiento del protocolo.

El análisis confirma que *CASCADE* logra una corrección prácticamente perfecta para valores de QBER moderados, pero su rendimiento decrece rápidamente una vez superado el umbral aproximado del 9 %. Este comportamiento delimita de forma clara su rango de operación óptimo y pone de manifiesto la importancia de mantener la tasa de errores por debajo de dicho límite para garantizar la viabilidad de la reconciliación de claves en un sistema QKD.

Capítulo 5 Conclusión

El recorrido seguido en este trabajo demuestra que aquello que comenzó como un ejercicio intelectual de la física teórica en el siglo XX, aunque aún lejos de su máximo potencial, hoy se materializa en un experimento funcional de distribución de clave cuántica.

La computación cuántica, ha actuado como puente entre ambos mundos, proporcionando las herramientas para manipular estados, construir circuitos y simular escenarios de comunicación que 60 años antes Richard Fayman solo podía imaginar.

La ejecución remota sobre hardware cuántico real de IBM ha permitido reproducir, de principio a fin, el flujo de un sistema QKD a nivel cuántico: generación de estados, transmisión y medición; con la posterior reconciliación de claves mediante *Cascade*.

En ausencia de espía, los resultados muestran que el protocolo corrige con éxito errores debidos al ruido del dispositivo; bajo un ataque *intercept–resend*, la QBER se incrementa hasta superar un umbral, detectando así la intrusión. La comparación entre simulaciones ideales, simulaciones con ruido controlado y ejecuciones reales confirma que el rendimiento final es el producto inseparable de las propiedades cuánticas de los estados y de las limitaciones físicas del hardware.

No obstante, estas pruebas aún estan alejadas de la realidad. La ausencia de transmisión física a través de un canal óptico impide evaluar fenómenos de la transmisión como pérdidas en fibra, dispersión o problemas de alineación entre emisores y receptores distantes. Además, la tecnología actual no está optimizada lo que añade restricciones de topología y genera ruido adicional que degrada la fidelidad de los resultados.

De cara al futuro, las líneas de desarrollo en QKD se centran en tres aspectos principales: la el aumento de las velocidades de transmisión, con sistemas experimentales que ya alcanzan tasas de generación de clave de varios cientos de Mbps; la extensión de las distancias, actualmente limitadas a unos pocos centenares de kilómetros en fibra óptica o enlaces satelitales, y que requieren el uso de repetidores cuánticos para posibilitar redes verdaderamente globales; y, finalmente, la integración en infraestructuras de telecomunicación a gran escala, donde los nodos de confianza, las arquitecturas de red cuántica y la interoperabilidad con algoritmos poscuánticos jugarán un papel clave. Estos avances permitirán que la distribución cuántica de claves evolucione desde implementaciones puntuales hacia una infraestructura segura y escalable para las comunicaciones del futuro.

Con la física definiendo la red y los límites del campo, la criptografía diseñando las estrategias y nosotros, los humanos, asumiendo el papel de jugadores. Solo falta garantizar que la pelota (estados cuánticos) y las cuerdas de la raqueta (transceptor cuántico) no se deteriore por el camino o presenten deficiencias, respectivamente. Para una QKD plenamente operativa, no solo se requiere control del hardware cuántico, sino también de toda la infraestructura de transmisión, sincronización y detección en condiciones reales.

Lista de Acrónimos

AES *Advanced Encryption Standard*. 12

Alice transmisor legítimo. 30–32, 37, 44, 48, 60

B92 Bennett 1992. 31, 37

BB84 Bennett y Brassard 1984. 31, 37

BBM92 Bennett, Brassard y Mermin 1992. 37

Bob receptor legítimo. 30–32, 37, 44, 48, 60

CRYSTALS-Dilithium *Cryptographic Suite for Algebraic Lattices – Dilithium*. 36

CRYSTALS-Kyber *Cryptographic Suite for Algebraic Lattices – Kyber*. 36

CV-QKD *Continuous Variable QKD*. 31, 37

DES *Data Encryption Standard*. 12

Diffie-Hellman Whitfield Diffie and Martin Hellman. 13, 36

DPS *Differential Phase Shift*. 31

DPS-QKD *Differential Phase Shift QKD*. 37

E91 Ekert 1991. 31, 37

ECC *Elliptic Curve Cryptography*. 12

ETSI *European Telecommunications Standards Institute*. 36

Eve *eavesdropper*. 30, 32, 33, 37, 45, 51, 54, 61

FIPS *Federal Information Processing Standards*. 36

IoT *Internet of Things*. 13

ISG-QKD *Industry Specification Group on QKD*. 36

MDI-QKD *Measurement Device Independent QKD*. 31, 36

ML-DSA *Module-Lattice Digital Signature Algorithm*. 36

ML-KEM *Module-Lattice Key Encapsulation Mechanism*. 36

NIST *National Institute of Standards and Technology*. 12, 36

QBER *Quantum Bit Error Rate*. 10, 30, 32, 52–54, 56, 58, 60–66

QKD *Quantum Key Distribution*. 13, 29–31, 36, 37, 65–67

RR-DPS *Round-Robin Differential Phase Shift*. 37

RSA Rivest, Shamir y Adleman. 12, 13, 36

SARG04 Scarani, Acín, Ribordy y Gisin 2004. 37

SLH-DSA *Stateless Lattice-based Hash Digital Signature Algorithm*. 36

SPHINCS+ *Stateless Practical Hash-based Incredibly Nice Cryptographic Signature Plus*. 36

TF-QKD *Twin-Field QKD*. 37

Bibliografía

[1] M. de Cervantes Saavedra, *El ingenioso hidalgo Don Quijote de la Mancha*. Madrid, Spain: Francisco de Robles, 1605.

[2] S. Singh, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. New York, NY, USA: Doubleday, 1999.

[3] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, 1948. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[4] C. E. Shannon y W. Weaver, *The Mathematical Theory of Communication*. Urbana, IL, USA: University of Illinois Press, 1949.

[5] National Bureau of Standards, *Data Encryption Standard (DES)*, FIPS PUB 46, Federal Information Processing Standards, 1977. dirección: https://csrc.nist.gov/files/pubs/fips/46/final/docs/nbs.fips.46.pdf.

[6] National Institute of Standards and Technology (NIST), *Advanced Encryption Standard (AES)*, FIPS PUB 197, nov. de 2001. dirección: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf.

[7] W. Diffie y M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, 1976. DOI: 10.1109/TIT.1976.1055638.

[8] R. L. Rivest, A. Shamir y L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, 1978. DOI: 10.1145/359340.359342.

[9] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," en *Proc. 35th Annu. Symp. Foundations of Computer Science*, IEEE Computer Society, 1994, págs. 124-134. DOI: 10.1109/SFCS.1994.365700.

[10] H. D. Pfister, C. Piveteau, J. M. Renes y N. Rengaswamy, "Belief propagation for classical and quantum systems: Overview and recent results," *IEEE BITS Inf. Theory Mag.*, 2022. DOI: 10.1109/MBITS.2023.3285848.

[11] R. P. Feynman, R. B. Leighton y M. Sands. "The Feynman lectures on physics, Volume III, Chapter 1: Quantum behavior." (2022), dirección: https://www.feynmanlectures.caltech.edu/III_01.html.

[12] M. Planck, "Zur Theorie des Gesetzes der Energieverteilung im Normalspectrum," *Verh. Dtsch. Phys. Ges.*, 1900. dirección: https://www.ub.edu/hcub/hfq/sites/default/files/planck-energieverteilung.pdf.

[13] R. P. Feynman, R. B. Leighton y M. Sands, *Quantum behavior - The Feynman lectures on physics, Vol. III, Chapter 1*, 1965. dirección: https://www.feynmanlectures.caltech.edu/III_01.html.

[14] P. G. Gonzalez, J. E. A. Bermejo y J. J. G. Sanz, *Física cuántica I*. Madrid, Spain: UNED, 2012.

[15] P. G. Molina y P. G. Esteban, *Mecánica Cuántica I*. Madrid, Spain: UNED, 2021.

[16] C. Bernhardt, *Quantum Computing for Everyone*. Cambridge, MA, USA: MIT Press, 2019, ISBN: 9780262039253.

[17] D. Deutsch, "Quantum theory, the Church–Turing principle and the universal quantum computer," *Proc. R. Soc. Lond. Ser. A*, 1985. DOI: 10.1098/rspa.1985.0070.

[18] IBM. "¿Qué es un qubit?" (2024), dirección: https://www.ibm.com/es-es/topics/qubit.

[19] R. Liu, G. G. Rozenman, N. K. Kundu, D. Chandra, D. De et al., "Towards the industrialisation of quantum key distribution in communication networks: A short survey," *IET Quantum Communication*, 2022. DOI: 10.1049/qtc2.12044.

[20] T. Jawaid, "Quantum computing and the future Internet," *arXiv preprint arXiv:2203.06180*, 2022. DOI: 10.48550/arXiv.2203.06180.

[21] IBM Quantum, *IBM Quantum Development Roadmap*, https://research.ibm.com/blog/ibm-quantum-roadmap-2023, Accedido: 11-ago-2025, 2023.

[22] Nature Editorial, "IBM unveils the first quantum computer with more than 1,000 qubits," *Nature*, dic. de 2023. DOI: 10.1038/d41586-023-03854-1.

[23] M. Field, A. Q. Chen, B. Scharmann et al., "Modular superconducting qubit architecture with a multi-chip tunable coupler," *Phys. Rev. Appl.*, 2024. DOI: 10.1103/PhysRevApplied.21.054063.

[24] T. P. Harty, D. T. C. Allcock, C. J. Ballance et al., "High-fidelity preparation, gates, memory, and readout of a trapped-ion quantum bit," *Phys. Rev. Lett.*, 2014. DOI: 10.1103/PhysRevLett.113.220501.

[25] X. Q. Technologies, "First demonstration of error-resistant photonic qubits on a chip using Gottesman–Kitaev–Preskill (GKP) states," *Nature*, vol. 1, 2025. DOI: 10.1038/s41586-025-09044-5.

[26] D. Aasen, M. Aghaee, Z. Alam et al., "Interferometric single-shot parity measurement in InAs–Al hybrid devices," *Nature*, vol. 638, págs. 651-655, 2025. DOI: 10.1038/s41586-025-00527-z.

[27] National Institute of Standards and Technology, *Federal information processing standards (FIPS)*, https://csrc.nist.gov/publications/fips, 2024.

[28] H.-K. Lo, M. Curty y B. Qi, "Measurement-device-independent quantum key distribution," *Phys. Rev. Lett.*, 2012. DOI: 10.1103/PhysRevLett.108.130503.

[29] A. K. Ekert, "Quantum cryptography based on Bell's theorem," *Phys. Rev. Lett.*, ago. de 1991. DOI: 10.1103/PhysRevLett.67.661.

[30] C. H. Bennett, G. Brassard y N. D. Mermin, "Quantum cryptography without Bell's theorem," *Phys. Rev. Lett.*, 1992. DOI: 10.1103/PhysRevLett.68.557.

[31] K. Inoue, E. Waks e Y. Yamamoto, "Differential-phase-shift quantum key distribution," *Phys. Rev. Lett.*, 2002. DOI: 10.1103/PhysRevLett.89.037902.

[32] M. Motaharifar, M. Hasani y H. Kaatuzian, "A survey on continuous variable quantum key distribution for secure data transmission: Toward the future of secured quantum-networks," *Quantum Inf. Comput.*, mayo de 2025. DOI: 10.2478/qic-2025-0009.

[33] C. H. Bennett, "Quantum cryptography using any two non-orthogonal states," *Phys. Rev. Lett.*, 1992.

[34] V. Scarani, A. Acín, G. Ribordy y N. Gisin, "Quantum cryptography protocols robust against photon-number-splitting attacks for weak laser pulse implementations," *Phys. Rev. Lett.*, 2004.

[35] M. Lucamarini, Z. Yuan, J. F. Dynes y A. J. Shields, "Overcoming the rate-distance barrier of quantum key distribution without using quantum repeaters," *arXiv preprint*, 2018.

[36] F. Bouchard et al., "Round-robin differential-phase-shift quantum key distribution," *Phys. Rev. A*, 2018.

[37] Q. Zhuang, Z. Zhang, J. Dove, F. N. C. Wong y J. H. Shapiro, "Floodlight quantum key distribution: A practical route to Gbps secret-key rates," *arXiv preprint*, 2015.

[38] W. Li, L. Zhang, H. Tan et al., "High-rate quantum key distribution exceeding 110 Mb/s," *arXiv*, 2023.

[39] H. Wang, Y. Li, T. Ye et al., "High-rate continuous-variable quantum key distribution over 100 km fiber with composable security," *arXiv*, 2025.

[40] L. Zhang et al., "Experimental Mode-Pairing Quantum Key Distribution ...," *Physical Review X*, 2025.

[41] —, "Researchers Demonstrate the UK's First Long-Distance ...," *The Quantum Insider (reported at OFC 2025)*, 2025.

[42] S.-K. Liao, W.-Q. Cai, W.-Y. Liu et al., "Satellite-to-ground quantum key distribution," *Nature*, 2017. DOI: 10.1038/nature23655.

[43] J.-W. Pan, C.-Z. Peng, Y. Wang et al., "Micius quantum experiments in space," *Rev. Mod. Phys.*, 2022. DOI: 10.1103/RevModPhys.94.035001.

[44] Y. Li, W.-Q. Cai, J.-G. Ren et al., "Microsatellite-based real-time quantum key distribution," *Nature*, 2025. DOI: 10.1038/s41586-025-08739-z.

[45] Qiskit contributors, *Qiskit documentation*, 2024. dirección: https://qiskit.org/documentation/.

[46] Qiskit Development Team, *Qiskit Aer: High-performance quantum circuit simulation*, https://docs.quantum.ibm.com/guides/simulate-with-qiskit-aer, 2024.

[47] Qiskit Development Team, *Qiskit IBM Quantum Provider: Access IBM Quantum systems and simulators*, https://docs.quantum.ibm.com/api/qiskit/providers, 2024.

[48] T. Kluyver, B. Ragan-Kelley, F. Pérez et al., "Jupyter notebooks — a publishing format for reproducible computational workflows," en *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, IOS Press, 2016, págs. 87-90. DOI: 10.3233/978-1-61499-649-1-87.

[49] M. Corporation, *Visual Studio Code*, https://code.visualstudio.com/, 2025.

[50] Conda Documentation Team. "Managing environments — Conda documentation." (2024), dirección: https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#creating-an-environment-with-commands.

[51] Qiskit, *Qiskit: Learn quantum computation using Qiskit*, https://www.youtube.com/@qiskit, 2020.