Detectando un Espía con Criptografía Cuántica

Detecting a Spy with Quantum Cryptography

Detectando um Espião com Criptografia Quântica

Mauricio Solar [1,(*)], Jean-Pierre Villacura [2], Felipe Cisternas Alvarez [3], Liuba Dombrovskaia [4]

Recibido: 10/10/2024 Aceptado: 10/10/2024

Resumen. - Este artículo muestra una implementación de la criptografía cuántica. Se introduce los conceptos básicos de la computación cuántica para comprender los términos mencionados en la implementación relacionados con la ciberseguridad y la distribución de llaves cuánticas (QKD). Se muestra una aplicación de QKD, donde se ve cómo se detecta fácilmente un espía cuando se intercepta un mensaje.

Palabras clave: Computación Cuántica; Distribución de Llaves Cuánticas (QKD); Criptografía Cuántica.

(*) Autor de correspondencia

[1] Académico, Universidad Técnica Federico Santa María, mauricio.solar@usm.cl, ORCID iD: https://orcid.org/0000-0002-4433-4622

[2] Estudiante postgrado, Universidad Técnica Federico Santa María, jean-pierre.rojas@sansano.usm.cl, ORCID iD: https://orcid.org/0009-0002-8611-1408

[3] Estudiante postgrado, Universidad Técnica Federico Santa María, felipe.cisternasal@sansano.usm.cl, ORCID iD: https://orcid.org/0009-0000-7029-8736

[4] Académica, Universidad Técnica Federico Santa María, liuba@inf.utfsm.cl, ORCID iD: https://orcid.org/0000-0001-6572-9765

Memoria Investigaciones en Ingeniería, núm. 27 (2024). pp. 200-219
https://doi.org/10.36561/ING.27.13
ISSN 2301-1092 • ISSN (en línea) 2301-1106 – Universidad de Montevideo, Uruguay

Summary. - *This article shows an implementation of quantum cryptography. We introduce the reader to the basic concepts of quantum computing so that they can easily understand the terms mentioned in the implementation related to cybersecurity and quantum key distribution (QKD). We show an application of QKD, where we can see how a spy is easily detected when a message is intercepted.*

Keywords: *Quantum Computing; Quantum Key Distribution (QKD); Quantum Cryptography.*

Resumo. ***-*** *Este artigo mostra uma implementação de criptografia quântica. Apresentamos ao leitor os conceitos básicos da computação quântica para que ele possa entender facilmente os termos mencionados na implementação relacionados à segurança cibernética e distribuição de chaves quânticas (QKD). Mostramos uma aplicação de QKD, onde podemos ver como um espião é facilmente detectado quando uma mensagem é interceptada.*

Palavras-chave: *Computação Quântica; Distribuição de Chaves Quânticas (QKD); Criptografia Quântica.*

1. Introducción. - Un computador cuántico (QC por las siglas en inglés de *Quantum Computing*) es un computador que explota fenómenos de la mecánica cuántica. A pequeña escala, la materia física exhibe propiedades tanto de partículas como de ondas, y la computación cuántica aprovecha este comportamiento utilizando hardware especializado. La física clásica no puede explicar el funcionamiento de estos dispositivos cuánticos, y un QC escalable podría realizar algunos cálculos exponencialmente más rápido que cualquier computador clásico moderno [1].

Nadie ha mostrado la mejor manera de construir un QC tolerante a fallas, y varias empresas y grupos de investigación están investigando diferentes tipos de bits cuánticos (qubits, por las siglas en inglés de *Quantum Bits*) [2]. Algunos ejemplos de estas tecnologías de qubits son procesadores de trampa de iones basados en puertas [3], procesadores superconductores basados en puertas [4], procesadores fotónicos [5], procesadores de átomos neutros [6], procesadores de átomos Rydberg [7], o recocidos cuánticos [5]. Sin embargo, el uso de este último se limita sólo a casos específicos [8].

Los datos cuánticos exhiben dos propiedades propias de la computación cuántica, la superposición y el entrelazamiento, lo que lleva a distribuciones de probabilidad conjuntas que podrían requerir una cantidad exponencial de recursos computacionales clásicos para representar o almacenar.

El qubit sirve como unidad básica de información cuántica. Representa un sistema de dos estados, como un bit clásico, excepto que puede existir en una superposición de sus dos estados. En cierto sentido, una superposición es como una distribución de probabilidad entre dos valores. Sin embargo, un cálculo cuántico puede verse influenciado por ambos valores a la vez, lo que es inexplicable por cualquiera de los estados individualmente. En este sentido, un qubit superpuesto almacena ambos valores simultáneamente. Al medir un qubit, el resultado es una salida probabilística de un bit clásico. Si un QC manipula el qubit de una manera particular, los efectos de interferencia de ondas pueden amplificar los resultados de medición deseados.

Ejemplos de datos cuánticos que pueden generarse o simularse en un dispositivo cuántico son la simulación química [9], la simulación de materia cuántica [10], el control cuántico [11], las redes de comunicación cuántica [12], o metrología cuántica [13].

La teoría de la computación cuántica señala que su velocidad de procesamiento puede ser mucho más rápida que incluso el supercomputador más rápido de la actualidad. Ejemplos como el algoritmo de Shor, con su capacidad potencial para factorizar grandes números primos en cuestión de segundos, a diferencia de los miles de años que podría tardar la computación clásica, se consideran signos de los avances y el desarrollo que vendrán con la computación cuántica [14], especialmente en criptografía.

La criptografía ha tenido un desarrollo importante desde 1975 con el establecimiento del algoritmo *Data Encryption Standard* (DES) para el cifrado de archivos mientras emergía la informática. Desde entonces se han desarrollado varios algoritmos para cumplir esta función criptográfica, siendo los más conocidos RSA (*Rivest-Shamir-Adleman*) y *Advanced Encryption Standard* (AES). La computación cuántica amenaza la seguridad de estos algoritmos al poder realizar cálculos mucho más rápido, pudiendo resolver operaciones que en el paradigma clásico llevarían unos 50 años utilizando supercomputadores en cuestión de segundos.

La Distribución de Llaves Cuánticas (QKD por sus siglas en inglés de *Quantum Key Distribution*) está estrechamente relacionada con la criptografía, ya que la seguridad de los datos depende de la transmisión de la llave previamente generada por un algoritmo criptográfico.

Este artículo muestra una implementación de la criptografía cuántica, destacando su potencial. La estructura de este artículo incluye una breve descripción de la computación cuántica seguida de una sección con el estado del arte que muestra los últimos avances en criptografía cuántica. En la sección IV se muestra una aplicación de la criptografía cuántica, donde se puede ver cómo se detecta fácilmente un espía cuando se intercepta un mensaje. La sección final

[5] https://www.dwavesys.com

resume las conclusiones.

2. Una breve descripción general de la computación cuántica. –

2.1. Propiedades de la computación cuántica. - La computación cuántica se basa en propiedades de la mecánica cuántica para procesar problemas que estarían fuera del alcance de los computadores clásicos. Un QC utiliza qubits. Los qubits son como bits normales en un computador clásico, pero con la capacidad adicional de colocarse en un estado de superposición y compartir entrelazamiento entre sí [15].

La computación cuántica funciona utilizando principios cuánticos. Los principios cuánticos requieren un nuevo diccionario de términos para comprenderse completamente, términos que incluyen superposición, entrelazamiento y decoherencia. A continuación, se explican brevemente estos principios.

- **Superposición:** La superposición establece que, al igual que las ondas en la física clásica, se puede agregar dos o más estados cuánticos y el resultado será otro estado cuántico válido. Por el contrario, también se puede representar cada estado cuántico como una suma de dos o más estados distintos. Esta superposición de qubits da a los QC un paralelismo inherente, permitiéndoles procesar millones de operaciones simultáneamente.
- **Entrelazamiento:** El entrelazamiento cuántico ocurre cuando dos sistemas se vinculan tan estrechamente que el conocimiento sobre uno proporciona conocimiento inmediato sobre el otro, sin importar qué tan lejos estén. Los procesadores cuánticos pueden sacar conclusiones sobre una partícula midiendo otra; el entrelazamiento cuántico permite a los QC resolver problemas complejos más rápidamente. Cuando se mide un estado cuántico, la función de onda colapsa y el estado se mide como cero o uno. En este estado conocido o determinista, el qubit actúa como un bit clásico. El entrelazamiento es la capacidad de los qubits de correlacionar su estado con otros qubits.
- **Decoherencia:** La decoherencia es la pérdida del estado cuántico en un qubit. Los factores ambientales, como la radiación, pueden provocar el colapso del estado cuántico de los qubits. Un gran desafío de ingeniería en la construcción de un QC es diseñar diversas características que intenten retrasar la decoherencia del estado, como la construcción de estructuras especiales que protejan los qubits de campos externos.

El estado actual de la computación cuántica se conoce como la era de la cuántica ruidosa de escala intermedia (NISQ, por las siglas en inglés de Noisy Intermediate-Scale Quantum), caracterizada por procesadores cuánticos que contienen entre 50 y 100 qubits que aún no están lo suficientemente avanzados para la tolerancia a fallas ni lo suficientemente grandes como para alcanzar la supremacía cuántica. El término NISQ fue acuñado por [16]. Estos procesadores, que son sensibles a su entorno (ruidoso) y propensos a la decoherencia cuántica, aún no son capaces de realizar una corrección continua de errores cuánticos. Esta escala intermedia está definida por el volumen cuántico, que se basa en el número moderado de qubits y la fidelidad de la puerta.

Los computadores clásicos realizan operaciones clásicas deterministas o pueden emular procesos probabilísticos utilizando métodos de muestreo. Al aprovechar la superposición y el entrelazamiento, los QC pueden realizar operaciones cuánticas que son difíciles de emular a escala con los computadores clásicos. Las ideas para aprovechar la computación cuántica NISQ incluyen optimización, simulación cuántica, criptografía y aprendizaje automático (ML por sus siglas en inglés de *Machine Learning*).

En particular, se cree que los QC son capaces de resolver rápidamente muchos problemas que ningún computador clásico podría resolver en un período de tiempo factible, una característica conocida como *supremacía cuántica* [17].

2.2. Puertas cuánticas. - El estado de los qubits se puede manipular aplicando puertas lógicas cuánticas, de forma análoga a cómo se pueden manipular los bits clásicos con puertas lógicas clásicas. A diferencia de muchas puertas lógicas clásicas, las puertas lógicas cuánticas son reversibles [18]. Las puertas de lógica cuántica están representadas por matrices unitarias, una puerta que actúa sobre n qubits está representada por una matriz unitaria de $2^n \times 2^n$.

Los estados cuánticos generalmente se representan mediante *kets*, de una notación conocida como *bra-ket*, la representación vectorial de un solo qubit se muestra en la ecuación 1.

$$|a\rangle = v_0|0\rangle + v_1|1\rangle \rightarrow \begin{bmatrix} v_0 \\ v_1 \end{bmatrix} \tag{1}$$

v_0 y v_1 son las amplitudes de probabilidad complejas del qubit, estos valores determinan la probabilidad de medir un 0 o un 1, al medir el estado del qubit. El valor cero está representado por el *ket* en la ecuación 2, y el valor uno está representado por el *ket* en la ecuación 3.

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{2}$$

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{3}$$

El producto tensorial denotado por el símbolo $\otimes$ se utiliza para combinar estados cuánticos. La acción de la puerta sobre un estado cuántico específico se encuentra multiplicando el vector $|\varphi_1 i$ que representa el estado por la matriz U que representa la puerta, por lo que el resultado es un nuevo estado cuántico $|\varphi_2 i$ mostrado en la ecuación 4.

$$U|\phi_1\rangle = |\phi_2\rangle \tag{4}$$

Existen muchas puertas cuánticas, a continuación, se revisan algunas de las más utilizadas en la literatura:

- **Puerta NOT:** Esta puerta es ampliamente conocida como puerta de Pauli-X, ya que esta puerta cuántica en particular transforma el estado existente del qubit para rotarlo alrededor del eje X. Como sugiere el nombre, la puerta NOT convertiría un qubit de su estado inicial a su estado complementario. Esta puerta cuántica está representada por la matriz en la ecuación 5 y opera como se muestra en la ecuación 6.

$$X := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{5}$$

$$X|0\rangle = |1\rangle, \qquad X|1\rangle = |0\rangle \tag{6}$$

- **Puerta Pauli-Y:** Las puertas Pauli-Y son capaces de rotar el qubit de entrada alrededor del eje Y. Esta puerta cuántica está representada por la matriz en la ecuación 7 y opera como se muestra en la ecuación 8.

$$Y := \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{7}$$

$$Y|0\rangle = i|1\rangle, \qquad Y|1\rangle = -i|0\rangle \tag{8}$$

- **Puerta Pauli-Z:** La puerta Pauli-Z o de fase flip es capaz de rotar el qubit de entrada alrededor del eje Z. Esta puerta cuántica está representada por la matriz en la ecuación ??.

$$Z := \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{9}$$

Pauli-Z deja el estado de sesgo $|0i$ sin cambios y asigna $|1i$ a $-|1i$ como se muestra en la ecuación 10.

$$Z|0\rangle = |0\rangle, \qquad Z|1\rangle = -|1\rangle \tag{10}$$

- **Puerta NOT controlada:** la puerta NOT controlada (CNOT) actúa en 2 (o más) qubits y realiza la operación NOT en el segundo (o más) qubit solo cuando el primer qubit es |1i. Esta puerta está representada por la matriz unitaria hermitiana (ecuación 11), y opera según la ecuación 12.

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{11}$$

$$\begin{aligned} CNOT|00\rangle &= |00\rangle \\ CNOT|01\rangle &= |01\rangle \\ CNOT|10\rangle &= |11\rangle \\ CNOT|11\rangle &= |10\rangle \end{aligned} \tag{12}$$

- **Puerta Hadamard:** La puerta Hadamard actúa sobre un solo qubit y crea estados de superposición iguales dado un estado de sesgo. La puerta de Hadamard realiza una rotación de π sobre el eje $(\hat{x} + \hat{z})/\sqrt{2}$ en la Esfera de Bloch (Figura I). Esta puerta está representada por la matriz de Hadamard (ecuación 13), y opera como en la ecuación 14.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{13}$$

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \qquad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{14}$$

2.3. Ruido. - El ruido está presente en los QC modernos. Los qubits son susceptibles a interferencias del medioambiente, fabricación imperfecta, TLS y, a veces, incluso rayos gamma. Hasta que se alcance la corrección de errores a gran escala, los algoritmos actuales deben poder seguir funcionando en presencia de ruido. Esto hace que probar algoritmos bajo ruido sea un paso importante para validar algoritmos y modelos cuánticos.

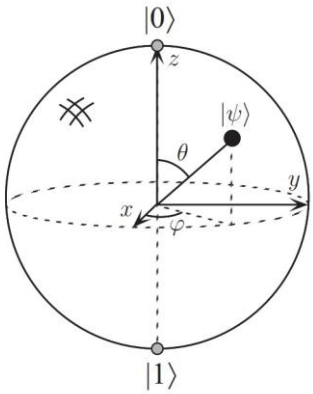

Figura I. Esfera de Bloch para representar un qubit.

2.4. Corrección de errores cuánticos (QEC). - La corrección de errores cuánticos (QEC por sus siglas en inglés de *Quantum Error Correction*) se utiliza en computación cuántica para proteger la información cuántica de errores debidos a la decoherencia y otros ruidos cuánticos. La corrección de errores tradicional emplea repetición excesiva. El código de repetición es la forma más sencilla pero ineficiente. La idea es almacenar la información varias veces y realizar una votación más amplia en caso de que luego se descubra que estas copias difieren. No es posible copiar

información cuántica debido al teorema de no-clonación. Este teorema parece presentar un obstáculo para formular una teoría de QEC; sin embargo, es concebible transferir la información lógica de un solo qubit a un estado altamente entrelazado de varios qubits físicos [16].

La evolución de la computación cuántica ha sido muy rápida. La Tabla I muestra una cronología de los principales avances en la computación cuántica.

3. Estado del Arte. - La Distribución de Llaves Cuánticas (QKD por sus siglas en inglés de *Quantum Key Distribution*) está estrechamente relacionada con la criptografía, ya que la seguridad de los datos depende de la transmisión de la llave previamente generada por un algoritmo criptográfico.

Existen peligros asociados a la transmisión de la llave que se solucionaron en parte con la introducción de la llave asimétrica (cuyo algoritmo más famoso es RSA).

La computación cuántica propone nuevas soluciones en este aspecto al poder transmitir la misma llave a dos destinatarios (Alice y Bob), con una altísima certeza de corroborar que no hay una tercera persona obteniendo esta llave. Esto es posible gracias a propiedades cuánticas como el entrelazamiento y la no-clonación. Con el entrelazamiento es posible conocer el estado de ambos fotones (es decir, dirección de giro) y la no-clonación permite detectar si hay algún intruso en el sistema, ya que arrojaría una llave común diferente a la de Alice y Bob (Figura II). Como Alice y Bob tienen la misma llave común, Alice puede cifrar su mensaje y enviárselo a Bob, quien tiene la llave para descifrarlo y, por lo tanto, leer el mensaje. Se asume un canal libre de ruido para esta situación.

La computación cuántica también puede contribuir al problema del "bloqueo de un solo uso" de la computación clásica al proporcionar llaves aleatorias debido al principio de incertidumbre de Heisenberg. Hay que tener en cuenta que en este tipo de problema la seguridad depende hasta cierto punto de la aleatoriedad de la llave.

En [17], los autores proponen un algoritmo AES modificado y utilizan computación cuántica para cifrar/descifrar archivos de imágenes AES utilizando IBM Qiskit para la evaluación del rendimiento. Ellos muestran que el algoritmo AES se puede implementar utilizando puertas cuánticas y sugieren que AES se implemente con generación de números aleatorios.

Fecha	**Principales avances en computación cuántica**
1970	James Park articula el teorema de no-clonación ?
1973	A. Holevo articula el teorema de Holevo y Ch. H. Bennet muestra que el cálculo se puede hacer de forma reversible ?
1980	Paul Beinoff describe el primer modelo de computador de mecánica cuántica ?, Tomasso Toffoli presenta la Puerta de Toffoli ?
1985	David Deutsch describe el primer QC universal
1992	D. Deutsch y R. Jozsa proponen un problema computacional que puede resolverse eficientemente con su algoritmo en un QC
1993	Dan Simon inventa un problema de oráculo, para el cual un QC sería exponencialmente más rápido que un computador clásico
1994	Peter Shor publica el algoritmo de Shor
1995	Shor propone los primeros esquemas para QEC ?
1996	Lov Grover inventa el algoritmo de búsqueda de BDs cuánticas
2000	Pati y Braunstein demostraron el teorema cuántico de no eliminación
2001	Primera ejecución del algoritmo de Shor
2003	Implementación del algoritmo Deutsch-Jozsa en un QC
2006	Primer QC de 12 qubits
2007	Sistema D-Wave muestra el uso de un QC de recocido de 28 qubits
2009	Creación del primer procesador cuántico electrónico

2010	Desarrollo del qubit de un solo electrón
2014	Los científicos transfieren datos mediante teletransportación cuántica a una distancia de 3 m con una tasa de error del 0% ?
2017	IBM presenta el QC de 17 qubits
2018	Google anuncia la creación de un chip cuántico de 72 qubits
2019	IBM revela su mayor QC hasta ese momento de 53 qubits
2020	Google informa la simulación química más grande en un QC
2021	IBM afirma que ha creado un procesador de 127 qubits
2022	El equipo de Google crea un agujero de gusano transitable con un QC
2023	Investigadores de Innsbruck entrelazaron dos iones a más de 230 m

Tabla I: Cronología de los principales avances de la computación cuántica

AES se combina con el uso de generación de números aleatorios en el proceso. En la implementación tradicional del algoritmo AES, la operación *Shift Row* mueve los datos para alinearlos en ciertos pasos de cifrado. Dado que el proceso de descifrado puede invertir el orden de estos pasos, se vuelve predecible. Para abordar esta vulnerabilidad, el autor sugiere modificar el rendimiento de la operación *Shift Row* para introducir movimientos aleatorios utilizando *Quantum Random Walk* (QRW), dificultando la predicción del orden correcto durante el proceso de descifrado, logrando una mayor seguridad que el enfoque clásico.

En [18], los autores se centran en analizar las características de la criptografía cuántica y explorar sus ventajas en el futuro de Internet. Analizan el protocolo QKD en el canal libre de ruido realizando mediciones de diferentes variables, como la probabilidad de que se detecte al espía v/s número de fotones medidos en un canal libre de ruido y 30% de ruido. También analiza las probabilidades de errores en el receptor v/s probabilidad de que un espía escuche a escondidas el canal.

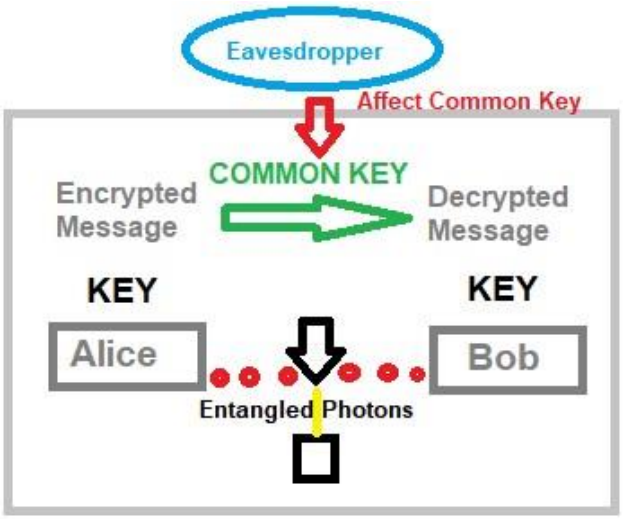

Figura II. Distribución de Llaves Cuánticas (QKD).

En [19], los autores hacen un aporte al estado del arte de la ciberseguridad desde amplias perspectivas. Ofrecen una visión general de la computación cuántica y cómo puede afectar a los problemas de ciberseguridad. También demuestra soluciones en computación cuántica a problemas del paradigma de computación clásica relacionado con la ciberseguridad y muestra cómo la computación cuántica podría usarse en el futuro para mejorar las soluciones de ciberseguridad.

En [20], los autores hacen una contribución proponiendo parámetros y cambios a RSA para hacer factibles la generación de llaves, el cifrado y descifrado, las firmas y la verificación en la computación actual y, al mismo tiempo, protegidos contra ataques de computación cuántica. Proponen un nuevo algoritmo cuántico para generar números de factores, conocido como Método de factorización de curvas elípticas de Grover más el uso de curvas de Edwards

(GEECM por sus siglas en inglés de *Grover plus Elliptic-Curve factorization Method using Edwards curves*) más rápido que Shor y algoritmos del paradigma clásico.

La Tabla II muestra la ventaja (columna "Ventaja comparativa") de la contribución realizada (2da columna) por la referencia en la columna "Ref".

4. Implementación de QKD. - En esta sección se usa la propiedad de los estados de superposición y se muestra su utilidad para compartir llaves cuánticas de forma segura. Esto se hace de una manera diferente al paradigma de computación clásica, donde no se puede estar seguro de si hay otras personas escuchando en el canal.

El principio de QKD consiste en encontrar una llave segura que pueda utilizarse para cifrar la comunicación entre dos interlocutores de forma segura, con una alta probabilidad de detectar un espía en el canal de comunicación.

Utiliza cuatro posibles estados cuánticos para representar los qubits de información, que se pueden representar en la esfera de Bloch (Figura III). Estos cuatro posibles estados cuánticos son:

- |0i y |1i: Estados de polarización lineal de la luz alineados horizontal y verticalmente, respectivamente.
- |+i y |−i: estados de polarización diagonal de la luz de ±45 grados alineados diagonalmente en $+45^o$ y -45^o. Son una superposición de los estados |0i y |1i.

Ref	**Contribución realizada**	**Ventaja comparativa**
[19]	Propuesta de algoritmo AES para Computación Cuántica con Seguridad mejorada usando QRW	Propuesta de versión Quantum del algoritmo AES con mejora contra ataques Quantum
[20]	Explicación de QKD y experimentos con Ruido Cuántico	Estado del arte sobre QKD y experimentos con espías
[21]	Ofrece una descripción general de QC relacionado con la ciberseguridad presentando varias soluciones Quantum y muestra cómo se pueden utilizar en el futuro.	Propone un estado del arte de los ataques Quantum y los enfoques existentes basados en Quantum para la ciberseguridad
[22]	Propone parámetros y cambios a RSA, en QC, para hacerlos factibles en la actualidad	Propone un GEECM, algoritmo más rápido que Shor y experimenta con escuchas de espías

Tabla II. Tabla comparativa.

Para fines prácticos, se supone que Alice quiere enviar el siguiente mensaje a Bob: 1 2 3 2 2 1 usando QKD.

Lo que se desea es enviar una cadena como mensaje a través del protocolo de comunicación cuántica, tomando en consideración los siguientes escenarios:

- Que el mensaje se transmita correctamente entre ambos interlocutores (Alice y Bob) sin la presencia de un espía.
- Detección del espía: Particularidad del paradigma de la computación cuántica basado en el principio de entrelazamiento y superposición.

En ambos casos se determina la 'Llave Compartida', que se utiliza para transmitir el mensaje de forma segura. Además del aspecto cuántico, la biblioteca *crytography.fernet* se utiliza para cifrar el mensaje que Alice envía a Bob utilizando la llave obtenida con el uso de este protocolo de comunicación cuántica. Las bibliotecas requeridas son las siguientes:

```
#Importing Libraries
from qiskit import QuantumCircuit, Aer, transpile
from qiskit.visualization import plot_histogram, plot_bloch_multivector
from numpy.random import randint
import numpy as np
import hashlib
import os
import random
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes from
    cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import padding
np.random.seed(seed=0)
```

4.1. Transmisión entre interlocutores sin presencia de un Espía. - Esta sección presenta la transmisión de un mensaje entre dos interlocutores utilizando el protocolo de comunicación cuántica BB84 sin la presencia de un espía (llamémosla Eve). Se utilizan las siguientes funciones y se explican a medida que se utilizan:

```
def convertir_a_binario(valor, prefijo):
  if isinstance(valor, str):
     binario = ''.join(format(ord(c), '08b') for c in valor)
  elif isinstance(valor, int):
     binario = bin(valor)[2:]
  if len(binario) < prefijo:
     binario = "0" * (prefijo - len(binario)) + binario
  return binario
def convertir_de_binario(cadena, prefijo):
  if len(cadena) % prefijo != 0:
     raise ValueError("La longitud de la cadena no es divisible por el prefijo")
  lista = []
  for i in range(0, len(cadena), prefijo):
     subcadena = cadena[i:i+prefijo]
     numero = int(subcadena, 2)
     lista.append(numero)
  return lista
def encode_message(bits, bases):
  message = []
  n=len(bases)
  for i in range(n):
     qc = QuantumCircuit(1,1)
     if str(bases[i]) == '0':
        if str(bits[i]) == '0':
           pass
        elif str(bits[i]) == '1':
           qc.x(0)
     else:
        if str(bits[i]) == '0':
           qc.h(0)
        elif str(bits[i]) == '1':
           qc.x(0)
     qc.h(0) qc.barrier()
     message.append(qc)
  return message
def measure_message(message, bases):
  backend = Aer.get_backend('aer_simulator')
  measurements = []
  n=len(bases)
  for q in range(n):
     if str(bases[q]) == '0': # base Z
        message[q].measure(0,0)
     if str(bases[q]) == '1': # base X
        message[q].h(0)
        message[q].measure(0,0)
     aer_sim = Aer.get_backend('aer_simulator')
     result = aer_sim.run(message[q], shots=1, memory=True).result()
     measured_bit = int(result.get_memory()[0])
     measurements.append(measured_bit)
  return measurements
def remove_no_coincidences(a_bases, b_bases, bits):
  good_bits = []
  n=len(bits)
  for q in range(n):
     if str(a_bases[q]) == str(b_bases[q]):
        good_bits.append(bits[q])
  return good_bits
def sample_bits(bits, selection):
```

```
    sample = []
    for i in selection:
        i = np.mod(i, len(bits))
        sample.append(bits.pop(i))
    return sample
```

La situación es la siguiente:

- Alice quiere enviar como mensaje la cadena: 1 2 3 2 2 1
- Alice quiere enviarle la cadena utilizando el protocolo de comunicación cuántica a Bob:
    - El cifrado binario que utiliza 2 bits para cada cifrado es: 0*b*011011101001
    - La 'Llave Común' se determina mediante el uso del protocolo cuántico para cifrar el mensaje y transmitirlo de forma segura.

Lo anterior se obtiene de la siguiente manera:

```
Mensaje_sin_encriptar=[1,2,3,2,2,1]
prefix=2
Mensaje_binario_sin_encriptar=''.join([convertir_a_binario(x,prefix) for x in
    Mensaje_si #print('Cadena en Binario que quiere enviar Alice a Bob: '+
    Mensaje_binario_sin_encriptar
n=30
Bits_a_mandar_testeo=randint(2, size=n)
```

Alice envía el mensaje (bits que se utilizarán para establecer la Llave Compartida) a Bob usando una base aleatoria que contiene información correspondiente a la base en la que se codificará cada qubit (Base-*Z* para 0 o Base-*X* para 1) de acuerdo a lo siguiente:

```
Base_Alice=randint(2,size=len(Bits_a_mandar_testeo))
print('Base Alice: '+str(Base_Alice))

Base Alice: [1 0 1 0 1 1 0 1 1 0 0 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 1 0 0 1]
```

Para codificar el mensaje con esta base se utiliza la función *encode_message(bits, bases)*, que funciona de la siguiente manera:

- Cada elemento de la base corresponde a la base de codificación:
    - 0: Preparar la codificación del qubit en la base *Z*
    - 1: Preparar la codificación del qubit en la base *X*
- Por ejemplo, si se tiene un bit 1 en la base 0, implica que la codificación es "1" en la base *Z*.
- Esto es importante ya que dependiendo de la base sobre la que se realice la codificación se utilizan diferentes puertas cuánticas. Para codificar un 1 en la base *X*, se aplica la puerta Hadamard *H* para integrar la superposición de estados, junto con la puerta *X* para cambiar el estado del qubit de 0 a 1. En las simulaciones de Qiskit, todos los estados se inicializan en 0 y, por lo tanto, a veces se debe usar la negación con la puerta *X* para tener el estado 1.
- Para cada bit que se va a codificar, utilizando las reglas anteriores, se utiliza *qc.barrier()* para separar los estados.
- Aplicando la función *encode_message(bits, bases)* sobre el mensaje que Alice quiere enviar a Bob, con la base de Alice, se tiene: *mensaje=codificar_mensaje(Bits_para_enviar_prueba, Base_Alice)*

Se realiza una rápida verificación de que el primer elemento del mensaje codificado sea consistente en su circuito cuántico según su qubit y base, produciendo la salida de la Figura III:

```
print('Bit: '+str(Bits_a_mandar_testeo[0])+', Base: '+str(Base_Alice[0]))
# Se presenta el circuito cuantico del primer qubit.
mensaje[0].draw()
```

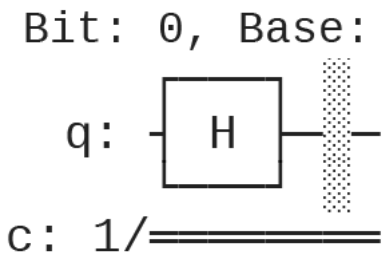

Figura III. Un circuito cuántico como salida.

Cuando la base es *X* y el bit es 1, se debe aplicar una puerta *X* y una puerta Hadamard (superposición de estados). Si la base es *Z* y el bit es 1, se debe aplicar una puerta *X*.

Determinando aleatoriamente la base de Bob y sumando la base obtenida por Alice anteriormente, se tiene:

```
print('Base Alice: '+str(Base_Alice))
Base_Bob=randint(2,size=len(Bits_a_mandar_testeo))
print('Base Bob: '+str(Base_Bob))

Base Alice: [1 0 1 0 1 1 0 1 1 0 0 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 1 0 0 1]
Base Bob:  [1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 1 1 0 1 0 0 1 0 1 1 1 1 1 1 0]
```

Se puede notar que las bases de ambos, generadas aleatoriamente, son diferentes entre sí. Bob codifica el mensaje enviado por Alice (relacionado con la configuración de la llave compartida, no del contenido) usando la función *medida_mensaje(mensaje, Base_Bob)* con su propia base:

```
print('Bob codifica mensaje(Shared Key) measure_message(mensaje, Base_Bob)')
Resultados_Bob=measure_message(mensaje,Base_Bob)
print('Resultados_Bob: '+str(Resultados_Bob))

Resultados_Bob: [0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1,
     0, 0, 0, 1, 0, 1, 1]
```

Con ambas bases se utiliza la función *remove_no_coincidences(Base_Alice, Base_Bob, Bits_ a_mandar_testeo)* para determinar el arreglo de bits cuya coincidencia en las bases es mutua, para lograr una llave mutua:

```
Alice_Key=list(map(int,remove_no_coincidences(Base_Alice, Base_Bob,
    Bits_a_mandar_testeo
Alice_Key_str=''.join(str(bit) for bit in Alice_Key) print('Alice_Key: '+
    Alice_Key_str)
#print(Alice_Key_str)
Bob_Key=remove_no_coincidences(Base_Alice, Base_Bob, Resultados_Bob)
Bob_Key_str=''.join(str(bit) for bit in Bob_Key)
print('Bob_Key: '+Bob_Key_str)

Alice_Key: 0110111000001001
Bob_Key:   0110111000001001
```

Se ha podido obtener una llave para Alice y Bob mediante el protocolo de comunicación cuántica. Ambas llaves son idénticas y, por lo tanto, se puede crear una llave compartida que Alice y Bob pueden usar para enviar mensajes de forma segura mediante un algoritmo de cifrado, como AES.

Ahora Alice envía el mensaje cifrado utilizando la llave obtenida mediante el algoritmo de cifrado AES y la llave determinada previamente con el protocolo de comunicación cuántica. Si esta llave tiene menos de 16 bits, se incorpora una extensión utilizando los primeros 16 caracteres del Hash SHA256:

```
def expandir_llave_aes128(llave):
  # Extension using HASH SHA256
  llave_hash = hashlib.sha256(llave.encode()).digest()
  return llave_hash[:16]
def encriptar_mensaje(llave, mensaje):
  # Convert the key to bytes using a hash function
  llave_bytes = hashlib.sha256(llave.encode()).digest()[:16]
  iv = os.urandom(16)
  cipher = Cipher(algorithms.AES(llave_bytes), modes.CBC(iv), backend=default_backend
    ( encriptador = cipher.encryptor()
  padder = padding.PKCS7(128).padder()
  mensaje_rellenado = padder.update(mensaje.encode()) + padder.finalize()
  mensaje_encriptado = encriptador.update(mensaje_rellenado) + encriptador.finalize()
  mensaje_encriptado_con_iv = iv + mensaje_encriptado
  return mensaje_encriptado_con_iv
def desencriptar_mensaje(llave, mensaje_encriptado):
  iv = mensaje_encriptado[:16]
  mensaje_encriptado_sin_iv = mensaje_encriptado[16:]
  llave_bytes = hashlib.sha256(llave.encode()).digest()[:16]
  cipher = Cipher(algorithms.AES(llave_bytes), modes.CBC(iv), backend=default_backend
    ( desencriptador = cipher.decryptor()
  mensaje_desencriptado = desencriptador.update(mensaje_encriptado_sin_iv) +
    desencrip
  unpadder = padding.PKCS7(128).unpadder()
  mensaje_original = unpadder.update(mensaje_desencriptado) + unpadder.finalize()
  return mensaje_original.decode()
def is_equal_Keys(Key1,Key2):
   if Key1==Key2:
    print('Llaves coinciden.')
    return True
   else:
    print('Llaves no coinciden.')
    return False
```

Luego, Alice envía la matriz de números 1 2 3 2 2 1, intentando cifrar el mensaje usando la llave obtenida, en este caso idéntica a la de Bob. Esto no siempre es así ya que la presencia de un espía muchas veces implica que la llave obtenida para ambos es diferente, por lo que se sospecha de la presencia de un espía, lo cual se presenta en la siguiente sección.

```
Mensaje_sin_encriptar=[1,2,3,2,2,1]
prefix=2
Mensaje_binario_sin_encriptar=''.join([convertir_a_binario(x,prefix) for x in
    Mensaje_si
Mensaje_cifrado=encriptar_mensaje(Alice_Key_str,Mensaje_binario_sin_encriptar)
```

Por lo tanto, el mensaje cifrado que Alice envía a través del canal a Bob es el siguiente:

```
print(Mensaje_cifrado.hex())

8dca5bd861f9658f928cc5e421414441aa2c6788e2df5476 f0e7992e2c68df27
```

Bob recibe este mensaje y utiliza la llave obtenida con el protocolo de comunicación cuántica para descifrarlo. Se supone que no hay interferencia de ruido en el canal:

```
mensaje_cifrado_recibido_por_Bob=Mensaje_cifrado
try:
  Mensaje_Descifrado_por_Bob=desencriptar_mensaje(Bob_Key_str,
    mensaje_cifrado_recibido
  print('Mensaje descifrado por Bob con su llave: '+Mensaje_Descifrado_por_Bob)
  mensaje_final=convertir_de_binario(Mensaje_Descifrado_por_Bob,2)
  print('')
  print(  equivale   a: '+str(mensaje_final))
except:
  print('ERROR. Llave usada no es correcta.')
  print('POSIBLE PRESENCIA DE UN ESPIA)
  print(' Key: '+Bob_Llave_str)
```

El resultado es que Alice y Bob han podido establecer una llave común utilizando el protocolo de comunicación cuántica. Alice usa esta llave para cifrar su mensaje y enviárselo a Bob por el canal.

Bob recibe este mensaje, utiliza su llave (generada por el protocolo de comunicación cuántica) y lo descifra, obteniendo con total seguridad y sin errores el mensaje que Alice le envió antes de cifrar, obteniendo como resultado: Mensaje descifrado por Bob con su llave: 011011101001 equivale a [1, 2, 3, 2, 2, 1].

4.2. Transmisión entre interlocutores con Detección de Espías. - ¿Qué sucede si Alice envía el mensaje para establecer la llave de cifrado a Bob y en el camino Eve intercepta este mensaje?

Hay posibilidades de que las llaves generadas con el protocolo QKD de Alice y Bob sean diferentes. Esto ocurre porque Eve (Espía) intercepta la comunicación y realiza mediciones de los qubits. En ciertos casos, Eve no sabe con certeza cuál era la polarización del fotón que midió y podría enviarle a Bob la polarización equivocada.

La repetición de este error es lo que hace que Alice y Bob obtengan llaves diferentes, y así puedan darse cuenta de que fue porque posiblemente había un espía en el camino.

La Figura IV ejemplifica cómo puede suceder que Bob reciba por error cierta preparación de fotones enviada por Alice, y Eve intercepte el mensaje.

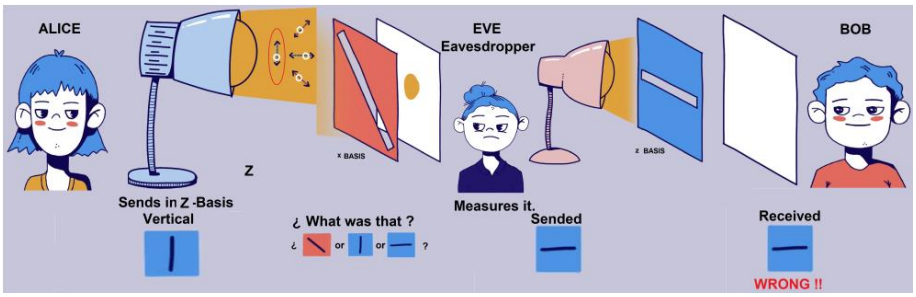

Figura IV: Mensaje interceptado por un espía (Eve)

Al recrear esta situación, se puede comprobar si las llaves generadas coinciden o no. Alice, al preparar los qubits a enviar, define su polarización de forma aleatoria:

```
np.random.seed(seed=5)
n=30
Alice_Bits=randint(2, size=n) Alice_Base=randint(2, size=n)
Mensaje_inicial_para_Key=encode_message(Alice_Bits,Alice_Base)
```

Eve define su propia base, intercepta *Initial_Message_for_Key* enviado por Alice a Bob y lo envía nuevamente. Cuando ella realiza una medición en el mensaje, debido a las propiedades cuánticas, el contenido cuando Bob lo lea

cambiará.

```
Eva_Base=randint(2, size=n)
mensaje_interceptado_por_Eva=measure_message(Mensaje_inicial_para_Key,Eva_Base)
```

Bob define su propia base y mide el mensaje:

```
Bob_Base=randint(2, size=n) resultado_Bob=measure_message(Mensaje_inicial_para_Key,
    Bob_Base) print('Bases')
print('Alice_Base: '+str(Alice_Base))
print('Eva_Base: '+str(Eva_Base))
print('Bob_Base: '+str(Bob_Base))

Bases
Alice_Base: [0 0 1 0 0 1 1 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0]
Eva_Base:    [1 1 1 0 1 1 1 0 0 1 0 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 0 0 0]
Bob_Base:    [0 1 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 0 0 1 0 1 0 0 0 1 1 0 1 0]
```

Con ambas bases (Alice y Bob) la función *remove_ no_ coincidencias (Base_Alice, Base_Bob, Message)* se utiliza para determinar el arreglo de bits cuya coincidencia en las bases es mutua:

```
Bob_Llave=list(map(int,remove_no_coincidences(Alice_Base,Bob_Base,resultado_Bob)))
Bob_Llave_str=''.join(str(bit) for bit in Bob_Llave) Alice_Llave=list(map(int,
    remove_no_coincidences(Alice_Base,Bob_Base,Alice_Bits)))
Alice_Llave_str=''.join(str(bit) for bit in Alice_Llave)
print(    Llave_Alice_con_intercepcion    : '+Alice_Llave_str)
print(    Llave_Bob_con_intercepcion    : '+Bob_Llave_str)

Llave_Alice_con_intercepcion: 11101001010001101110
Llave_Bob_con_intercepcion:    01111011100111110110
```

Se puede notar que las llaves no coinciden, porque Eve interfiere con la comunicación al realizar una medición. Esto tiene como consecuencia que, al cifrar el mensaje, al no tener la misma llave, no se pueda descifrar con éxito.
Para comprobar cómo se da esta situación:

```
Mensaje_sin_encriptar=[1,2,3,2,2,1]
prefix=2
Mensaje_binario_sin_encriptar=''.join([convertir_a_binario(x,prefix) for x in
    Mensaje_si
Mensaje_cifrado=encriptar_mensaje(Alice_Llave_str,Mensaje_binario_sin_encriptar)
```

Alice envía el siguiente mensaje cifrado 1 2 3 2 2 1 en forma binaria (0*b*011011101001) usando la llave obtenida (interceptada por Eve):

```
print(Mensaje\_cifrado.hex())
#generando la salida

c00d63ad44c4df7e0668b28f554060337b5afdf592f01cd6 ff73dc8597139496
```

Bob recibe este mensaje y utiliza la llave obtenida con el protocolo de comunicación cuántica para descifrar este mensaje. Su llave es diferente a la de Alice debido a la interferencia de Eve:

```
mensaje_cifrado_recibido_por_Bob=Mensaje_cifrado
try:
   Mensaje_Descifrado_por_Bob=desencriptar_mensaje(Bob_Llave_str,
    mensaje_cifrado_recibi
   print('Mensaje descifrado por Bob con su llave: '+Mensaje_Descifrado_por_Bob)
   print(Mensaje_Descifrado_por_Bob)
   mensaje_final=convertir_de_binario(Mensaje_Descifrado_por_Bob,2)
   print('')
   print('equivale a: '+str(mensaje_final))
except:
  print('ERROR. Llave usada no es correcta.')
```

```
  print('POSIBLE PRESENCIA DE UN ESPIA')
  print(' Llave: '+Bob_Llave_str)

ERROR. Llave usada no es correcta.
POSIBLE PRESENCIA DE UN ESPIA
  Llave: 01111011100111110110
```

Se puede notar que no es posible descifrar el mensaje ya que la llave obtenida por Bob no es correcta, y es diferente a la de Alice debido a la interferencia de Eve. Por este motivo, es válido sospechar la presencia de un espía ya que las llaves distribuidas no coinciden.

Veamos qué sucede al usar la llave de Alice:

```
Bob_Llave_str=Alice_Llave_str
try:
  Mensaje_Descifrado_por_Bob=desencriptar_mensaje(Bob_Llave_str,
    mensaje_cifrado_recibi
  print('Mensaje descifrado por Bob si hubiera coincidido con la llave de Alice: '
    mensaje_final=convertir_de_binario(Mensaje_Descifrado_por_Bob,2)
  print('')
  print('equivale a: '+str(mensaje_final))
except:
  print('ERROR. Llave usada no es correcta.   )
  print(' Llave: '+Bob_Llave_str)
```

Mensaje descifrado por Bob si hubiera coincidido con la llave de Alice: 011011101001 equivale a: [1,2,3,2,2,1].

Se concluye entonces que realmente existe la presencia de un espía en el canal que afecta la distribución de llaves cuánticas y hace que Alice y Bob obtengan llaves diferentes.

En esta situación, Alice y Bob deben sospechar de la presencia de un espía y tomar medidas en base a esa sospecha. Es aconsejable no comunicar en estas condiciones.

Se ha comprobado que la presencia de espías afecta la distribución de llaves entre Alice y Bob. Si no son idénticos, existe la posibilidad de que haya un espía en la comunicación. Esta característica de poder verificar si hay un espía en el canal es una de las ventajas de utilizar QKD, frente al paradigma clásico, que es mucho más complejo para detectar un espía.

5. Conclusiones. - La computación cuántica aún se encuentra en sus primeras etapas y construir un QC funcional y eficiente con suficientes qubits llevará años.

El desarrollo de la criptografía poscuántica es crucial para mitigar los riesgos de ciberseguridad que plantea la computación cuántica. La criptografía poscuántica se refiere a algoritmos que son resistentes a los ataques de los QC. No sólo mejora las capacidades de búsqueda en BDs, sino que también aborda problemas de optimización en diversos dominios comerciales, como análisis de datos, logística e investigación médica.

Se ha podido implementar QKD utilizando circuitos cuánticos con la biblioteca Qiskit Python. Se verifica la ocurrencia de dos situaciones:

- Generación de una llave cuántica entre dos interlocutores sin la presencia de un espía: Se obtuvo con éxito una llave compartida para Bob y Alice, el mensaje pudo cifrarse con AES utilizando la llave cuántica obtenida. Posteriormente, el descifrado se llevó a cabo con éxito utilizando la llave obtenida por Bob. El objetivo de comunicación se cumple en este escenario.
- Generación de llave cuántica entre dos interlocutores con presencia de un espía: Se comprueba que es posible detectar espías en el canal mediante QKD ya que la acción de espiar implica tomar una medición, que modifica el contenido del mensaje recibido por el receptor Bob. Esto ocurre principalmente por el principio

de superposición de estados. Esta situación se pudo verificar ya que Bob obtuvo una llave diferente a la de Alice debido a la interferencia de Eve. Debido a que las llaves obtenidas son diferentes, Alice y Bob pueden sospechar que hay un espía en el canal.

Mientras más larga sea la llave generada, mayores serán las posibilidades de detectar eficazmente un espía en la red.

Descubrir mejores algoritmos para trabajar con computación cuántica sigue siendo un área de investigación abierta. Finalmente, este estudio ofrece una visión general de la ciberseguridad cuántica y estudios en computación cuántica con sus posibles aplicaciones.

Referencias

[1] R. P. Feynman (1982). Simulating physics with computers. Int. J. Theor. Phys. 21, 467–488

[2] J.D. Whitfield, J. Yang, W. Wang, J.T. Heath and B. Harrison. Quantum computing 2022.

[3] I. Pogorelov, T. Feldker, Ch.D. Marciniak, L. Postler, G. Jacob, O. Krieglsteiner, V. Podlesnic, M. Meth, V. Negnevitsky, M. Stadler, B. Höfer, C. Wächter, K. Lakhmanskiy, R. Blatt, P. Schindler and T. Monz. Compact Ion-Trap Quantum Computing Demonstrator. PRX Quantum, vol. 2, 2, pp. 020343, 2021. https://link.aps.org/doi/10.1103/PRXQuantum.2.020343

[4] S. Kwon, A. Tomonaga, G. L. Bhai, S. J. Devitt and J.-S. Tsai. Gate-based superconducting quantum computing. J. Appl. Phys. 129(4): 041102. 2021. https://doi.org/10.1063/5.0029735

[5] J. S. Lee, N. Farmakidis, C. D. Wright and H. Bhaskaran. Polarization-selective reconfigurability in hybridized-active-dielectric nanowires. Science Advances, 8eabn9459. 2022. DOI:10.1126/sciadv.abn9459

[6] J. Wurtz et al. Aquila: Quera's 256-qubit neutral-atom quantum computer. 2023. https://arxiv.org/abs/2306.11727.

[7] M. Kornjača, R. Samajdar, T. Macrì et al. Trimer quantum spin liquid in a honeycomb array of Rydberg atoms. Commun Phys 6, 358 (2023). https://doi.org/10.1038/s42005-023-01470-z

[8] S. H. Adachi and M. P. Henderson. Application of Quantum Annealing to Training of Deep Neural Networks. 2015. https://arxiv.org/abs/1510.06356

[9] Y. Cao, J. Romero, J.P. Olson, M. Degroote, P.D. Johnson, M. Mária, I. D. Kivlichan, T. Menke, B. Peropadre, N.P.D. Sawaya, S. Sim, L. Veis and A. Aspuru-Guzik. Quantum Chemistry in the Age of Quantum Computing. Chemical Reviews, Vol. 119, No. 19, pp. 10856–10915, 2019. https://doi.org/10.1021/acs.chemrev.8b00803

[10] Ma, M. Govoni and G. Galli. Quantum simulations of materials on near-term quantum computers.npj Comput Mater 6, 85, 2020. https://doi.org/10.1038/s41524-020-00353-z

[11] Sivarajah. What is Quantum Control Theory? AZoQuantum. 2022. https://www.azoquantum.com/Article.aspx?ArticleID=335

[12] R. Bassoli, H. Boche, C. Deppe, R. Ferrara, F. H. P. Fitzek, G. Janssen and S. Saeedinaeeni. Quantum Communication Networks. Foundations in Signal Processing, Communications and Networking. 2021. Springer. https://doi.org/10.1007/978-3-030-62938-0

[13] Y.L. Len, T. Gefen, A. Retzker et al. Quantum metrology with imperfect measurements. Nat Commun 13, 6971. 2022. https://doi.org/10.1038/s41467-022-33563-8

[14] A. Díaz, M. Rodriguez and M. Piattini. Towards a set of metrics for hybrid (quantum/classical) systems maintainability. Journal of Universal Computer Science, vol. 30, no. 1, pp. 25-48, 2024.

[15] S. N., Singh, H. and N.A.U. An extensive review on quantum computers. Advances in Engineering Software, 174, 103337. 2022. https://doi.org/10.1016/j.advengsoft.2022.103337

[16] J. Preskill, J. Quantum Computing in the NISQ era and beyond. Quantum, 2, 79. 2018. doi:10.22331/q-2018-08-06-79

[17] M. Brooks. Beyond quantum supremacy: the hunt for useful quantum computers. Nature, 574(7776), 19-21. 2020. doi:10.1038/d41586-019-02936-3

[18] C.H. Bennett. Logical reversibility of computation. IBM Journal of Research and Development, 17(6), 525-532. 1973. doi:10.1147/rd.176.0525

[19] R. Raussendorf. Key ideas in quantum error correction. Philo- sophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences , 370 (1975), 4541-4565. 2012. doi:10.1098/rsta.2011.0494

[20] J.L. Park. The concept of transition in quantum mechanics. Foundations of Physics, 1, 23-33. 1970.

[21] P. Benioff. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. Journal of Statistical Physics, 22(5), 563-591. 1980. doi:10.1007/BF01011339

[22] T. Toffoli. Reversible computing. In J. de Bakker J. van Leeuwen (Eds.), Automata, languages and programming (pp. 632–644). 1980. Berlin, Heidelberg: Springer Berlin Heidelberg.

[23] P.W. Shor. Scheme for reducing decoherence in quantum computer memory, 52(4), R2493-R2496. 1995. doi:10.1103/PhysRevA.52.R2493

[24] W. Pfaff, B.J. Hensen, H. Bernien, S.B. van Dam, M.S. Blok, T.H. Taminiau, R. Hanson. Unconditional quantum teleportation between distant solid-state quantum bits. Science, 345(6196), 532–535. 2014. doi:10.1126/science.1253512

[25] K.K. Ko and E.S. Jung. Development of cybersecurity technology and algorithm based on quantum computing. Applied Sciences, 11(19). 2021. doi:10.3390/app11199085

[26] X.L. Tianqi Zhou and J. Shen. Quantum cryptography for the future internet and the security analysis. Security and Communication Networks. 2018. https://doi.org/10.1155/2018/8214619

[27] P.C. Uttam Ghosh and D. Das. A comprenhensive tutorial on cybersecurity in quantum computing paradigm. TechRxiv. 2023. https://doi.org/10.36227/techrxiv.22277251.v1

[28] D.J. Bernstein, N. Heninger, P. Lou and L. Valenta. Post-quantum rsa. Cryptology ePrint Archive, Paper 2017/351. 2017. https://eprint.iacr.org/2017/351

Nota contribución de los autores:

1. Concepción y diseño del estudio
2. Adquisición de datos
3. Análisis de datos
4. Discusión de los resultados
5. Redacción del manuscrito
6. Aprobación de la versión final del manuscrito

MS ha contribuido en: 1, 2, 3, 4, 5 y 6.
JPV ha contribuido en: 1, 2, 3, 4, 5 y 6.
FCA ha contribuido en: 1, 2, 3, 4, 5 y 6.
LD ha contribuido en: 1, 2, 3, 4, 5 y 6.

Nota de aceptación: Este artículo fue aprobado por los editores de la revista Dr. Rafael Sotelo y Mag. Ing. Fernando A. Hernández Gobertti.