Detecting a spy with Quantum Cryptography*

1st Mauricio Solar
Dept. of Informatics
Universidad Tecnica Federico Santa Maria (UTFSM)
Santiago, Chile
mauricio.solar@usm.cl or ORCID

2nd Felipe Cisternas Alvarez
Dept. of Informatics
Universidad Tecnica Federico Santa Maria (UTFSM)
Valparaiso, Chile
felipe.cisternasal@sansano.usm.cl

3rd Jean-Pierre Villacura
Dept. of Informatics
Universidad Tecnica Federico Santa Maria (UTFSM)
Valparaiso, Chile
jean-pierre.rojas@sansano.usm.cl

4th Liubov Dombrovskaia
Informatique Dept. (of Aff.)
UTFSM (of Aff.)
Santiago, Chile
liuba@inf.utfsm.cl or ORCID

***Abstract*—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.**

***Index Terms*—quantum computing, quantum key distribution, quantum cryptography**

I. Introduction

A Quantum Computer (QC) is a computer that exploits quantum mechanical phenomena. At small scales, physical matter exhibits properties of both particles and waves, and quantum computing leverages this behavior using specialized hardware. Classical physics cannot explain the operation of these quantum devices, and a scalable QC could perform some calculations exponentially faster than any modern classical [1].

No one has shown the best way to build a fault-tolerant QC, and multiple companies and research groups are investigating different types of qubits [2]. Some examples of these qubit technologies are gate-based ion trap processors [3], gate-based superconducting processors [4], photonic processors [5], neutral atom processors [6], Rydberg atom processors [7], or quantum annealers https://www.dwavesys.com. However, their use is limited to specific cases only [8].

Quantum data exhibits two properties proper of quantum computing, superposition and entanglement, leading to joint probability distributions that could require an exponential amount of classical computational resources to represent or store.

The quantum bit (qubit) serves as the basic unit of quantum information. It represents a two-state system, just like a classical bit, except that it can exist in a superposition of its two states. In one sense, a superposition is like a probability distribution over the two values. However, a quantum computation can be influenced by both values at once, inexplicable by either state individually. In this sense, a superposed qubit stores both values simultaneously. When measuring a qubit, the result is a probabilistic output of a classical bit. If a QC manipulates the qubit in a particular way, wave interference effects can amplify the desired measurement results.

Examples of quantum data that can be generated or simulated on a quantum device are chemical simulation [9], quantum matter simulation [10], quantum control [11], quantum communication networks [12], or quantum metrology [13].

The theory of quantum computing points out that its processing speed can be much faster than even the fastest supercomputer today. Examples such as Shor's algorithm with its potential ability to factor large prime numbers in a matter of seconds, as opposed to the thousands of years that classical computing could take, are considered signs of the advances and development that is to come with quantum computing [14], especially in cryptography.

Cryptography has had an important development since 1975 with the establishment of the Data Encryption Standard (DES) algorithm for file encryption while computing was emerging. Since then, several algorithms have been developed to fulfill this cryptographic function, the best known being RSA (Rivest-Shamir-Adleman) and Advanced Encryption Standard (AES). Quantum computing threatens the security of these algorithms by being able to perform calculations much faster. Being able to solve operations that in the classical paradigm would take about 50 years using supercomputers in a matter of seconds.

Quantum Key Distribution (QKD) is closely related to cryptography, since the security of the data depends on the transmission of the key previously generated by a cryptographic algorithm.

This paper shows an implementation of quantum cryptography, highlighting the immense potential of quantum computing. The structure of this paper includes a brief overview of quantum computing followed by a section with the state of the art showing the latest advances in quantum cryptography. In section IV we show an application of quantum cryptography, where we can see how a spy is easily detected when a message is intercepted. The final section summarizes the conclusions.

II. A Brief overview of Quantum Computing

A. Quantum Computing Properties

Quantum computing relies on properties of quantum mechanics to compute problems that would be out of reach for classical computers. A Quantum Computer (QC) uses qubits. Qubits are like regular bits in a classical computer, but with the added ability to be put into a superposition state and share entanglement with one other [15].

A QC works using quantum principles. Quantum principles require a new dictionary of terms to be fully understood, terms that include superposition, entanglement, and decoherence. Let's explain these principles below.

- **Superposition:** Superposition states that, much like waves in classical physics, you can add two or more quantum states and the result will be another valid quantum state. Conversely, you can also represent every quantum state as a sum of two or more other distinct states. This superposition of qubits gives QCs their inherent parallelism, allowing them to process millions of operations simultaneously.
- **Entanglement:** Quantum entanglement occurs when two systems link so closely that knowledge about one gives you immediate knowledge about the other, no matter how far away they are. Quantum processors can draw conclusions about one particle by measuring another one, Quantum entanglement allows QCs to solve complex problems faster. When a quantum state is measured, the wave function collapses and you measure the state as either a zero or a one. In this known or deterministic state, the qubit acts as a classical bit. Entanglement is the ability of qubits to correlate their state with other qubits.
- **Decoherence:** Decoherence is the loss of the quantum state in a qubit. Environmental factors, like radiation, can cause the quantum state of the qubits to collapse. A large engineering challenge in constructing a QC is designing the various features that attempt to delay decoherence of the state, such as building specialty structures that shield the qubits from external fields.

The current state of quantum computing is referred to as the noisy intermediate-scale quantum (NISQ) era, characterized by quantum processors containing 50–100 qubits which are not yet advanced enough for fault-tolerance or large enough to achieve quantum supremacy, the term NISQ was coined by [16]. These processors, which are sensitive to their environment (noisy) and prone to quantum decoherence, are not yet capable of continuous quantum error correction. This intermediate-scale is defined by the quantum volume, which is based on the moderate number of qubits and gate fidelity.

Classical computers perform deterministic classical operations or can emulate probabilistic processes using sampling methods. By harnessing superposition and entanglement, QCs can perform quantum operations that are difficult to emulate at scale with classical computers. Ideas for leveraging NISQ quantum computing include optimization, quantum simulation, cryptography, and Machine Learning (ML).

Notably, QCs are believed to be able to solve many problems quickly that no classical computer could solve in any feasible amount of time—a feat known as quantum supremacy [17].

B. Quantum Gates

The state of qubits can be manipulated by applying quantum logic gates, analogous to how classical bits can be manipulated with classical logic gates. Unlike many classical logic gates, quantum logic gates are reversible [18]. Quantum logic gates are represented by unitary matrices, a gate which acts on n qubits is represented by $2^n \times 2^n$ unitary matrix.

Quantum states are typically represented by kets, from a notation know as bra-ket, the vector representation of a single qubit is shown in equation 1.

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

Here v_0 and v_1 are the complex probability amplitudes of the qubit, these values determine the probability of measuring a 0 or a 1, when measuring the state of the qubit. The value zero is represented by the ket in equation 2, and the value one is represented by the ket in equation 3.

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

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

The tensor product denoted by the symbol $\otimes$, is used to combine quantum states. The action of the gate on a specific quantum state is found by multiplying the vector $|\phi_1\rangle$ which represents the state by the matrix U representing the gate, thus the result is a new quantum state $|\phi_2\rangle$ shown in equation 4.

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

There exist many number of quantum gates, below we review some of the most often used in the literature:

- **NOT Gate:** This gate is widely known as X-Pauli gate, as this particular quantum gate transforms the existing state of the qubit to be rotated around the X-axis. As the name suggests, the NOT gate would convert a qubit from its initial state to its complement state. This quantum gate is represented by the matrix in equation 5, and operates as shown in equation 6.

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

$$X|0\rangle = |1\rangle \text{ and } X|1\rangle = |0\rangle \tag{6}$$

- **Y-Pauli Gate:** The Y-Pauli gates are capable of rotating the input qubit around the Y-axis. This quantum gate is

represented by the matrix in equation 7, and operates as shown in equation 8.

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

$$Y|0\rangle = i|1\rangle \text{ and } Y|1\rangle = -i|0\rangle \tag{8}$$

- **Z-Pauli Gate:** The Z-Pauli or phase flip gate are capable of rotating the input qubit around the Z-axis. This quantum gate is represented by the matrix in equation 9.

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

Pauli Z leaves the biasis state $|0\rangle$ unchanged and maps $|1\rangle$ to $-|1\rangle$ as shown in equation 10.

$$Z|0\rangle = |0\rangle \text{ and } Z|1\rangle = -|1\rangle \tag{10}$$

- **Controlled NOT Gate:** the Controlled NOT (CNOT) gate acts on 2 (or more) qubits, and performs the NOT operation on the second (or more) qubit only when the first qubit is $|1\rangle$, this gate is represented by the Hermitian unitary matrix (equation 11), and operates as in equation 12.

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

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

- **Hadamard gate:** The Hadamard gate, acts on a single qubit and creates an equal superposition states given a biasis state. The Hadamard gate performs a rotation of π bout the axis $(\hat{x} + \hat{z})/\sqrt{2}$ at the Bloch Sphere (Figure 1). This gate is represented by the Hadamard matrix (equation 13), and operates as in equation 14.

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

$$\text{H—}0\rangle = \tfrac{|0\rangle+|1\rangle}{\sqrt{2}} \text{ and } H|1\rangle = \tfrac{|0\rangle-|1\rangle}{\sqrt{2}} \tag{14}$$

C. Noise

Noise is present in modern day QCs. Qubits are susceptible to interference from the surrounding environment, imperfect fabrication, TLS and sometimes even gamma rays. Until large scale error correction is reached, the algorithms of today must be able to remain functional in the presence of noise. This makes testing algorithms under noise an important step for validating quantum algorithms and quantum models.

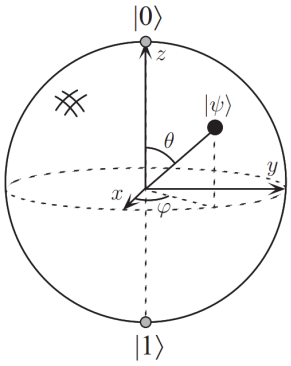

Fig. 1. Bloch sphere to represent a qubit

D. Quantum Error Correction (QEC)

QEC is used in quantum computing to protect quantum information from errors due to decoherence and other quantum noise. Traditional error correction employs over repetition. The repetition code is the simplest but most inefficient way. The idea is to store the information multiple times and take a larger vote in the event that these copies are later found to differ. Copying quantum information is not possible due to the no-cloning theorem. This theorem seems to present an obstacle to formulating a theory of QEC, nevertheless, it is conceivable to transfer the logical information of a single qubit to a highly entangled state of several physical qubits [19].

The evolution of quantum computing has been very fast. Table I shows a timeline of quantum computing major advances.

III. State of the Art

Quantum Key Distribution (QKD) is closely related to cryptography, since the security of the data depends on the transmission of the key previously generated by a cryptographic algorithm.

There are dangers associated with the transmission of the key that were partly solved with the introduction of the asymmetric key (whose most famous algorithm is RSA).

Quantum computing proposes new solutions in this aspect by being able to transmit the same key to two recipients (Alice and Bob), with a very high certainty of corroborating that there is no third person obtaining this key. This is possible due to quantum properties such as entanglement and non-cloning. With entanglement it is possible to know the state of both photons (i.e. direction of spin) and non-cloning allows to detect if there is any intruder in the system, since it would yield a common key different from Alice and Bob (Figure 2).

Since Alice and Bob have the same common key, Alice can encrypt her message and send it to Bob, who has the key to decrypt and therefore read the message. We assume noise-free channel for this situation.

Quantum computing can also contribute to the 'one-time-pad' problem of classical computing by providing random keys due to the Heisenberg uncertainty principle. Note that

TABLE I
TIMELINE OF QUANTUM COMPUTING MAJOR ADVANCES

Date	Quantum Computing Major Advances
1970	James Park articulates the no-cloning theorem [20]
1973	A. Holevo articulates the Holevo's Theorem and Ch. H. Bennet shows that computation can be done reversibly [18]
1980	Paul Beinoff describes the 1st quantum mechanical computer model [21], Tomasso Toffoli introduces the Toffoli Gate [22]
1985	David Deutsch describes the 1st universal QC
1992	D. Deutsch and R. Jozsa propose a computational problem that can be solved efficiently with their algorithm on a QC
1993	Dan Simon invents an oracle problem, for which a QC would be exponentially faster than a conventional computer
1994	Peter Shor publishes the Shor's Algorithm
1995	Shor proposes the 1st schemes for QEC [23]
1996	Lov Grover invents the quantum DB search algorithm
2000	Pati and Braunstein proved the quantum no-deleting theorem
2001	First execution of Shor's algorithm
2003	Implementation of the Deutsch–Jozsa algorithm on a QC
2006	First 12 qubit QC benchmarked
2007	D-Wave System demonstrates use of a 28-qubit annealing QC
2009	First electronic quantum processor created
2010	Single-electron qubit developed
2014	Scientists transfer data by quantum teleportation over a distance of 3 m with 0% error rate [24]
2017	IBM unveils 17-qubit QC
2018	Google announces the creation of a 72-qubit quantum chip
2019	IBM reveals its biggest QC yet, consisting of 53 qubits
2020	Google reports the largest chemical simulation on a QC
2021	IBM claims that it has created a 127 quantum bit processor
2022	Google team make traversable wormhole with a QC
2023	Researchers of Innsbruck entangled two ions over 230 m

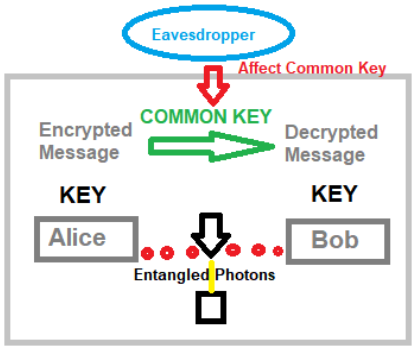

Fig. 2. Quantum Key Distribution (QKD)

in this type of problem security depends to some degree on the randomness of the key.

In [25], the authors propose a modified AES algorithm and use quantum computing to encrypt/decrypt AES image files using IBM Qiskit for performance evaluation. They show that AES algorithm can be implemented using quantum gates and suggest that AES be implemented with random number generation.

AES is combined with the use of random number generation in the process. In the traditional implementation of the AES algorithm, the Shift Row operation moves the data to align them at certain encryption steps. Since the decryption process can reverse the order of these steps, it becomes predictable. To address this vulnerability, the author suggests modifying the performance of the Shift Row operation to introduce random movements using Quantum Random Walk (QRW), making it difficult to predict the correct order during the decryption process, achieving greater security than classical approach.

In [26], the authors focus on analyzing characteristics of the quantum cryptography and exploring of the advantages of it in the future internet. They analyze the QKD protocol in the noise-free channel by making measurements of different variables. Probability of the eavesdropper being detected v/s number of photons measured in a noise-free channel and 30% noise. Also analyzes the probabilities of errores in the receiver v/s probability of eavesdropper to eavesdrop on the channel.

In [27], the authors make a contribution in the state of the art of cybersecurity from wide perspectives. They give an overview of quantum computing and how it can affect cybersecurity issues. Also demonstrate solutions in quantum computing to problems in classical computing paradigm related to cybersecurity, and relates how quantum computing could be used in the future to make cybersecurity solutions better.

In [28], the authors make a contribution proposing parameters and changes to RSA to make Key-Generation, encrypt and decryption, signatures and verification feasible in actual computing and, at the same time, protected against quantum computing attacks. They propose a new quantum algorithm to generate factor numbers, GEECM (Grover plus Elliptic-Curve factorization Method using Edwards curves) faster than Shor and algorithms of classic paradigm.

Table II shows the advantage (column "Comparative advantage") of the contribution made (2nd column) by the reference in column "Ref".

TABLE II
COMPARATIVE TABLE

Ref	Contribution made	Comparative advantage
[25]	Propose of AES algorithm for Quantum Computing with improved Security using QRW	Propose of Quantum version of AES algorithm with improvement against Quantum attacks
[26]	Explication of QKD and experiments with Quantum Noise	State of art about QKD and experiments with eavesdropper
[27]	Give an overview of QC related to Cybersecurity presenting several Quantum solutions and show how can be used in future	Proposes a state of art of Quantum attacks, and existing Quantum-based approaches for Cybersecurity
[28]	Propose parameters and changes to RSA,on QC, to make feasible in actual	Proposes a GEECM, faster algorithm than Shor and experiments with eavesdropper

IV. IMPLEMENTATION OF QKD

In this section we will use the property of superposition states and we will display its utility to share quantum keys safely and securely. This is done in a different way than the classical paradigm computing, where we can not be sure if there is other people listening in the channel.

The principle of QKD consists of finding a secure key so that it can be used to encrypt the communication between two interlocutors safely, with a high probability of detecting a eavesdropper in the communication channel.

It uses four possible quantum states to represent the qubits of information, which can be represented in the Bloch sphere (Figure 1). These four possible quantum states are:

- $|0\rangle$ and $|1\rangle$: Linear polarization states of light aligned horizontally and vertically, respectively.
- $|+\rangle$ and $|-\rangle$: $+-45$ degree diagonal polarization states of light aligned diagonally at $+45^o$ and -45^o. They are a superposition of the states $|0\rangle$ and $|1\rangle$.

For practical purposes, let's assume that Alice wants to send the following message to Bob: 1 2 3 2 2 1 using QKD.

What we want is to send a string as a message through the quantum communication protocol, taking into consideration the following scenarios:

- That the message is transmitted correctly between both interlocutors (Alice and Bob) without the presence of a spy (Eavesdropper).
- Detection of the Eavesdropper: Particularity of the quantum computing paradigm based on the principle of entanglement and superposition.

In both cases, the 'Shared Key' is determined, which is used to transmit the message securely. In addition to the quantum aspect, the $crytography.fernet$ library is used to encrypt the message that Alice sends to Bob using the key obtained with the use of this quantum communication protocol. The required libraries are the following:

```
#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)
```

A. Transmission between interlocutors without the presence of Eavesdropper

This section presents the transmission of a message between two interlocutors using the BB84 quantum communication protocol without the presence of an eavesdropper (Eve). The following functions are used and are explained as they are used:

```
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
```

The situation is the following:

- Alice wants to send as a message the string: 1 2 3 2 2 1
- Alice wants to send the chain using the quantum communication protocol to Bob:
  - Binary encryption using 2 bits for each cipher is: $0b011011101001$
  - The 'Common Key' is determined with the use of the quantum protocol to encrypt the message and transmit it securely.

The above is obtained in the following way:

```
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 sends the message (bits that will be used to establish the Shared Key) to Bob using a random base that contains information corresponding to the base in which each qubit will be encoded (Z-Basis 0 or X-Basis 1) according to the following:

```
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]
```

To encode the message with this base, the function $encode\_message(bits, bases)$ is used, which works as follows:

- Each element of the base corresponds to the coding base:
  - 0: Prepare qubit encoding in base Z.
  - 1: Prepare qubit encoding in base X.
- For example, if you have a bit 1 in the base 0, it implies that the encoding is "1" in the base Z.
- This is important since depending on the basis on which the coding is carried out, different quantum gates are used. To encode a 1 into the base X, the Hadamard gate H is applied to integrate the superposition of states, along with the gate X to change the state of the qubit from 0 to 1. In Qiskit simulations all states are initialized to 0, and therefore sometimes negation must be used with the gate X to have state 1.
- For each bit to be encoded, using the above rules, $qc.barrier()$ is used to separate the states.

By applying the function $encode\_message(bits, bases)$ on the message that Alice wants to send to Bob, with Alice's base, we have: $message = encode\_message(Bits\_to\_send\_test, Base\_Alice)$
A quick verification is carried out that the first element of the coded message is consistent in its quantum circuit according to its qubit and base, producing the output of Figure 3:

```
print('Bit: '+str(Bits_a_mandar_testeo[0])+',
    Base: '+str(Base_Alice[0]))
# The quantum circuit of the first qubit is
    drawn.
mensaje[0].draw()
```

When the base is X and the bit is 1, a gate X and a Hadamard gate (state superposition) should be applied. If the base is Z and the bit is 1, a gate X should be applied.

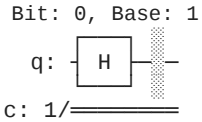

Fig. 3. A quantum circuit as output

By randomly determining Bob's base and adding the base obtained by Alice previously, we have:

```
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]
```

It can be noted that the bases of both, generated randomly, are different from each other. Bob encodes the message sent by Alice (related to setting the shared key, not content) using the function $measure\_message(message, Base\_Bob)$ with his own 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]
```

With both bases, the function $remove\_no\_coincidences(Base\_Alice, Base\_Bob, Bits\_a\_mandar\_testeo)$ is used to determine the array of bits whose coincidence in the bases is mutual, to achieve a mutual key:

```
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
```

It has been possible to obtain a key for Alice and Bob using the quantum communication protocol. Both keys are identical and therefore a shared key can be created that Alice and Bob can use to send messages securely using an encryption algorithm, such as AES.
Now Alice sends the encrypted message using the key obtained using the AES algorithm for encryption and the key determined previously with the quantum communication protocol. If this key has less than 16 bits, an

extension is incorporated using the first 16 characters of the SHA256 Hash:

```
def expandir_clave_aes128(clave):
  # Extension using HASH SHA256
  clave_hash = hashlib.sha256(clave.encode()).
    digest()
  return clave_hash[:16]
def encriptar_mensaje(clave, mensaje):
  # Convert the key to bytes using a hash
    function
  clave_bytes = hashlib.sha256(clave.encode()).
    digest()[:16]
  iv = os.urandom(16)
  cipher = Cipher(algorithms.AES(clave_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(clave,
    mensaje_encriptado):
  iv = mensaje_encriptado[:16]
  mensaje_encriptado_sin_iv = mensaje_encriptado
    [16:]
  clave_bytes = hashlib.sha256(clave.encode()).
    digest()[:16]
  cipher = Cipher(algorithms.AES(clave_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
```

Alice then sends the array of numbers 1 2 3 2 2 1, attempting to encrypt the message using the key obtained, in this case identical to Bob's. This is not always the case since the presence of a spy often implies that the key obtained for both is different, and therefore the presence of a spy is suspected, which is presented in the following section.

```
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)
```

Therefore, the encrypted message that Alice sends through the channel to Bob is the following:

```
print(Mensaje_cifrado.hex())

8dca5bd861f9658f928cc5e421414441aa2c6788e2df5476
    f0e7992e2c68df27
```

Bob receives this message and uses the key obtained with the quantum communication protocol to decrypt this message. It is assumed that there is no interference with noise in the channel:

```
mensaje_cifrado_recibido_por_Bob=Mensaje_cifrado
try:
  Mensaje_Descifrado_por_Bob=
    desencriptar_mensaje(Bob_Key_str,
    mensaje_cifrado_recibido
  print('Message decrypted by Bob with his key:
    '+Mensaje_Descifrado_por_Bob)
  mensaje_final=convertir_de_binario(
    Mensaje_Descifrado_por_Bob,2)
  print('')
  print('Equivalente a: '+str(mensaje_final))
except:
  print('ERROR. Used key is not correct.')
  print('POSSIBLE PRESENCE OF EAVESDROPPER')
  print(' Key: '+Bob_Llave_str)
```

Message decrypted by Bob with his key: 011011101001 is equivalent to [1, 2, 3, 2, 2, 1].
The result is that Alice and Bob have been able to establish a common key using the quantum communication protocol. Alice uses this key to encrypt her message and send it over the channel to Bob.
Bob receives this message, uses its key (generated by the quantum communication protocol) and decrypts it, obtaining with complete certainty and without errors the message that Alice sent him before encrypting.

B. Transmission between interlocutors with Eavesdropper Detection

What happens if Alice sends the message to set the encryption key to Bob and on the way Eve intercepts this message? There are possibilities that the keys generated with Alice and Bob's QKD protocol are different. This occurs because Eve (Spy) intercepts the communication and performs measurements of the qubits. In certain cases Eve does not know with certainty what the polarization of the photon she measured was and she could send Bob the wrong one.
The repetition of this error is what causes Alice and Bob to obtain different keys, and thus they can realize that it was because there was possibly a spy along the way.
The following figure 4 exemplifies how it can happen that a certain photon preparation sent by Alice can be mistakenly received by Bob, with Eve intercepting the message.

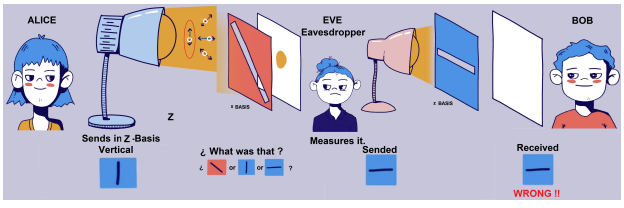

Fig. 4. Message intercepted by a spy

By recreating this situation you can check whether the generated keys match or not.

Alice, when preparing the qubits to be sent, defines their polarization randomly:

```
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)
```

Eva defines her own base, intercepts $Initial\_Message\_for\_Key$ sent by Alice to Bob and sends it again. When she makes a measurement on the message, due to quantum properties the content when Bob reads it will change.

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

Bob defines his own base and measures the message:

```
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]
```

With both bases (Alice and Bob) the function $remove\_no\_coincidences(Base\_Alice, Base\_Bob, Message)$ is used to determine the array of bits whose coincidence in the bases is mutual:

```
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('Alice_Key_with_interception: '+
    Alice_Llave_str)
print('Bob_Key_with_interception: '+
    Bob_Llave_str)

Alice_Key_with_interception:
    11101001010001101110
Bob_Key_with_interception:
    01111011100111110110
```

It can notice that the keys do not match, because Eve interferes with communication when making a measurement. This has the consequence that when encrypting the message, because they do not have the same key, it cannot be successfully decrypted.
Let's verify how this situation occurs:

```
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 sends the following encrypted message 1 2 3 2 2 1 in binary form ($0b011011101001$) using the obtained key (intercepted by Eve):

```
print(Mensaje\_cifrado.hex())
#generating the output

c00d63ad44c4df7e0668b28f554060337b5afdf592f01cd6
    ff73dc8597139496
```

Bob receives this message and uses the Key obtained with the quantum communication protocol to decrypt this message. His key is different from Alice's due to Eve's interference:

```
mensaje_cifrado_recibido_por_Bob=Mensaje_cifrado
try:
   Mensaje_Descifrado_por_Bob=
    desencriptar_mensaje(Bob_Llave_str,
    mensaje_cifrado_recibi
   print('Mensaje desencriptado 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('Equivalente a: '+str(mensaje_final))
except:
  print('ERROR. Used key is not correct.')
  print('POSSIBLE PRESENCE OF EAVESDROPPER')
  print(' Key: '+Bob_Llave_str)

ERROR. Used key is not correct.
POSSIBLE PRESENCE OF EAVESDROPPER
  Key: 01111011100111110110
```

It can be noted that it is not possible to decrypt the message since the key obtained by Bob is not correct, it is different from Alice's due to Eve's interference. For this reason, it is valid to suspect the presence of an eavesdropper since the distributed keys do not match.
Let's check what happens when using Alice's key:

```
Bob_Llave_str=Alice_Llave_str
try:
  Mensaje_Descifrado_por_Bob=
    desencriptar_mensaje(Bob_Llave_str,
    mensaje_cifrado_recibi
  print('Mensaje desencriptado por Bob si
    hubiese coincidido con la llave de Alice: '
    mensaje_final=convertir_de_binario(
    Mensaje_Descifrado_por_Bob,2)
  print('')
  print('Equivalente a: '+str(mensaje_final))
except:
  print('ERROR. Used key is not correct.')
  print(' Key: '+Bob_Llave_str)
```

The message decrypted by Bob if it had matched Alice's key: 011011101001 is equivalent to: $[1, 2, 3, 2, 2, 1]$.
It is then concluded that there truly is the presence of a spy in the channel that affects the distribution of quantum keys and causes Alice and Bob to obtain different keys. In this situation, Alice and Bob must suspect the presence of a spy and take action based on this suspicion. It is advisable not to communicate under these conditions.

It has been verified that the presence of eavesdropper affects the distribution of keys between Alice and Bob. If they are not identical, there is a possibility that there is an eavesdropper in the communication. This feature of being able to verify if there is a spy in the channel is one of the advantages of using QKD, compared to the classic paradigm, which is much more complex to detect a spy.

V. Conclusions

Quantum Computing is still in its early stages, and building a functional and efficient QC with enough qubits will take years.

The development of post-quantum cryptography is crucial to mitigate the cybersecurity risks posed by quantum computing. Post-quantum cryptography refers to algorithms that are resistant to attacks from QCs. It not only improves database search capabilities but also addresses optimization problems in various business domains such as data analytics, logistics, and medical research.

We have been able to implement QKD using quantum circuits with the Qiskit Python library. We verify the occurrence of two situations:

- Generation of a quantum key between two interlocutors without the presence of an eavesdrepper: A shared key was successfully obtained for Bob and Alice, the message could be encrypted with AES using the quantum key obtained. Subsequently, the decryption was successfully carried out using the key obtained by Bob. The communication objective is met in this scenario.
- Generation of quantum key between two interlocutors with the presence of an eavesdrepper: It is verified that it is possible to detect spies in the channel using QKD since the action of spying involves taking a measurement, which modifies the content of the message received by receiver Bob. This occurs mainly due to the principle of superposition of states. This situation could be verified since Bob obtained a different key than Alice due to Eve's interference. Because the keys obtained are different, Alice and Bob may then suspect that there is a spy on the channel.

The longer the key generated, the greater the chances of effectively detecting a spy on the network.

Discovering better algorithms to work with quantum computing is still an open area of research. Finally, this study gives an overview of quantum cybersecurity and studies in quantum computations with its possible applications.

References

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

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

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

[4] Sangil Kwon, Akiyoshi Tomonaga, Gopika Lakshmi Bhai, Simon J. Devitt, Jaw-Shen Tsai (2021): Gate-based superconducting quantum computing. J. Appl. Phys. 129(4): 041102. https://doi.org/10.1063/5.0029735

[5] June Sang Lee, Nikolaos Farmakidis, C. David Wright and Harish Bhaskaran (2022): Polarization-selective reconfigurability in hybridized-active-dielectric nanowires. Science Advances, 8eabn9459. DOI:10.1126/sciadv.abn9459

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

[7] Kornjača, M., Samajdar, R., Macrì, T. et al. (2023): 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] Steven H. Adachi, Maxwell P. Henderson (2015): Application of Quantum Annealing to Training of Deep Neural Networks. https://arxiv.org/abs/1510.06356

[9] Cao, Yudong, Romero, Jonathan, Olson, Jonathan P., Degroote, Matthias, Johnson, Peter D., Kieferová, Mária, Kivlichan, Ian D., Menke, Tim, Peropadre, Borja, Sawaya, Nicolas P.D., Sim, Sukin, Veis, Libor, Aspuru-Guzik, Alán (2019): Quantum Chemistry in the Age of Quantum Computing. Chemical Reviews, Vol. 119, No. 19, pp. 10856–10915, https://doi.org/10.1021/acs.chemrev.8b00803

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

[11] Sivarajah, Ilamaran. (2022): What is Quantum Control Theory?. AZoQuantum. Retrieved on March 06, 2024 from https://www.azoquantum.com/Article.aspx?ArticleID=335

[12] Riccardo Bassoli, Holger Boche, Christian Deppe, Roberto Ferrara, Frank H. P. Fitzek, Gisbert Janssen, Sajad Saeedinaeeni (2021): Quantum Communication Networks. Foundations in Signal Processing, Communications and Networking. Springer. https://doi.org/10.1007/978-3-030-62938-0

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

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

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

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

[17] Brooks, M. (2019). Beyond quantum supremacy: the hunt for useful quantum computers. Nature, 574(7776), 19-21. doi:10.1038/d41586-019-02936-3

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

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

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

[21] Benioff, P. (1980). 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. doi:10.1007/BF01011339

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

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

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

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

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

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

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