Evolución de la librería QuantumSolver para el desarrollo cuántico

RECSI 2022: Santander 19-21 Octubre 2022

Daniel Escánez-Expósito
Pino Caballero-Gil
Francisco Martín-Fernández

Escuela Superior de Ingeniería y Tecnología
Universidad de La Laguna

CryptULL

IBM Research

RECSI 2022

ÍNDICE

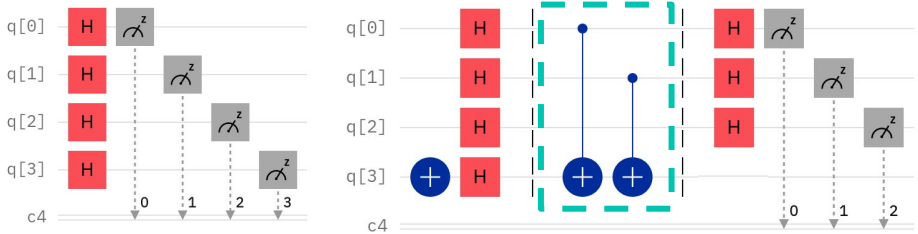
PLANTEAMIENTO INICIAL

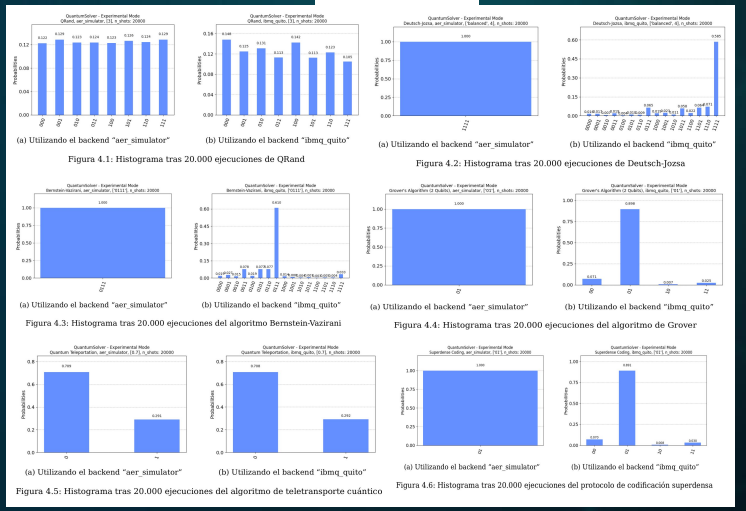
01

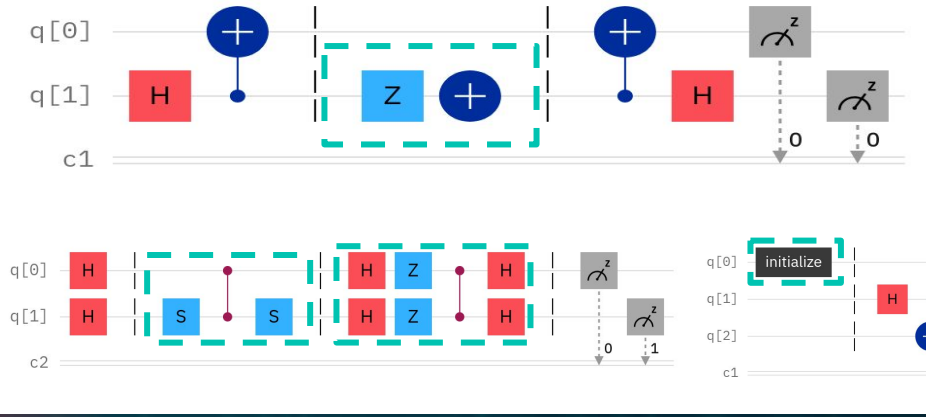
Vista general de la librería

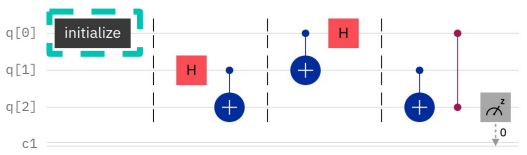
COMPUTACIÓN CUÁNTICA

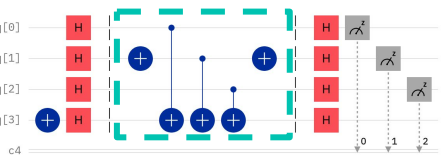
INTERÉS, SEGURIDAD Y DEMOCRATIZACIÓN











RECSI 2022: Santander 19-21 Octubre 2022

OBJETIVOS DE QUANTUMSOLVER

Un módulo de Python para la implementación de algoritmos cuánticos apoyado en Qiskit

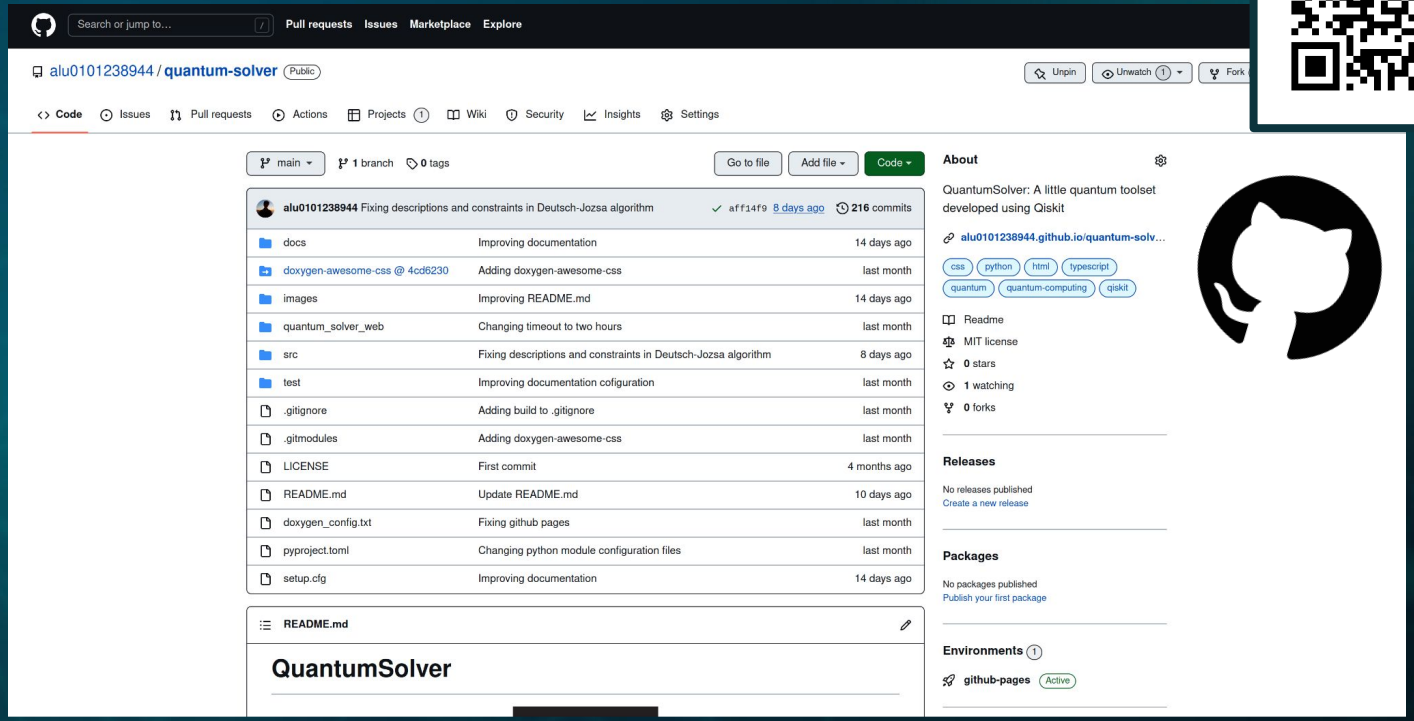
Cuenta con dos modos de ejecución:

- Ejecución simple
- Modo Experimental

Contiene un programa principal (CLI) y una interfaz web más accesible para el público general

Permite la ejecución en hardware cuántico real de IBM, así como en simuladores

REPOSITORIO *OPEN SOURCE*

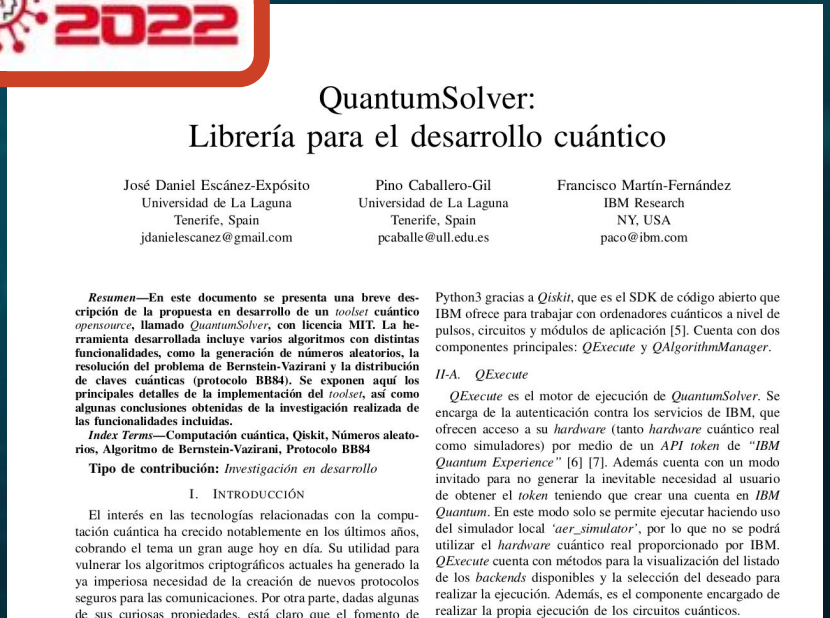


RECSI 2022: Santander 19-21 Octubre 2022

RECSI 2022: Santander 19-21 Octubre 2022

VII Jornadas Nacionales de Investigación en Ciberseguridad (JNIC)

Bilbao, Spain (27-29 de junio de 2022)



QuantumSolver:
Librería para el desarrollo cuántico

José Daniel Escánez-Expósito
Universidad de La Laguna
Tenerife, Spain
jdanielescanez@gmail.com

Pino Caballero-Gil
Universidad de La Laguna
Tenerife, Spain
pcaballe@ull.edu.es

Francisco Martín-Fernández
IBM Research
NY, USA
paco@ibm.com

***Resumen*—En este documento se presenta una breve descripción de la propuesta en desarrollo de un *toolset* cuántico *opensource*, llamado *QuantumSolver*, con licencia MIT. La herramienta desarrollada incluye varios algoritmos con distintas funcionalidades, como la generación de números aleatorios, la resolución del problema de Bernstein-Vazirani y la distribución de claves cuánticas (protocolo BB84). Se exponen aquí los principales detalles de la implementación del *toolset*, así como algunas conclusiones obtenidas de la investigación realizada de las funcionalidades incluidas.**

***Index Terms*—Computación cuántica, Qiskit, Números aleatorios, Algoritmo de Bernstein-Vazirani, Protocolo BB84**

Tipo de contribución: *Investigación en desarrollo*

I. Introducción

El interés en las tecnologías relacionadas con la computación cuántica ha crecido notablemente en los últimos años, cobrando el tema un gran auge hoy en día. Su utilidad para vulnerar los algoritmos criptográficos actuales ha generado la ya imperiosa necesidad de la creación de nuevos protocolos seguros para las comunicaciones. Por otra parte, dadas algunas de sus curiosas propiedades, está claro que el fomento de

Python3 gracias a *Qiskit*, que es el SDK de código abierto que IBM ofrece para trabajar con ordenadores cuánticos a nivel de pulsos, circuitos y módulos de aplicación [5]. Cuenta con dos componentes principales: *QExecute* y *QAlgorithmManager*.

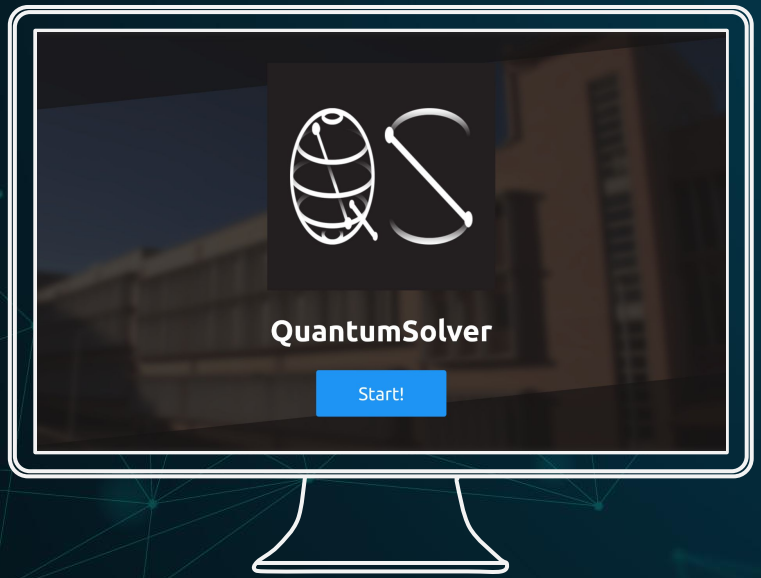
II-A. QExecute

QExecute es el motor de ejecución de *QuantumSolver*. Se encarga de la autenticación contra los servicios de IBM, que ofrecen acceso a su *hardware* (tanto *hardware* cuántico real como simuladores) por medio de un *API token* de *"IBM Quantum Experience"* [6] [7]. Además cuenta con un modo invitado para no generar la inevitable necesidad al usuario de obtener el *token* teniendo que crear una cuenta en *IBM Quantum*. En este modo solo se permite ejecutar haciendo uso del simulador local *'aer_simulator'*, por lo que no se podrá utilizar el *hardware* cuántico real proporcionado por IBM. *QExecute* cuenta con métodos para la visualización del listado de los *backends* disponibles y la selección del deseado para realizar la ejecución. Además, es el componente encargado de realizar la propia ejecución de los circuitos cuánticos.

INTERFACES 02

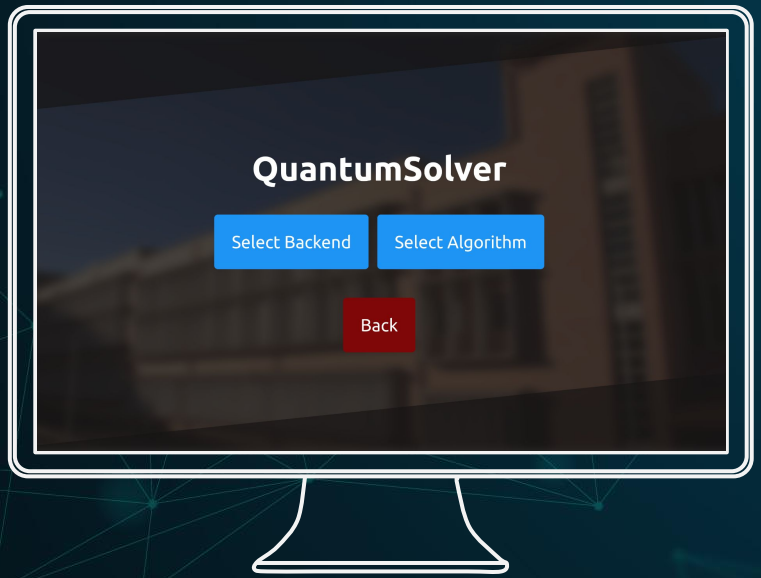
Visualización de QuantumSolver

INICIO



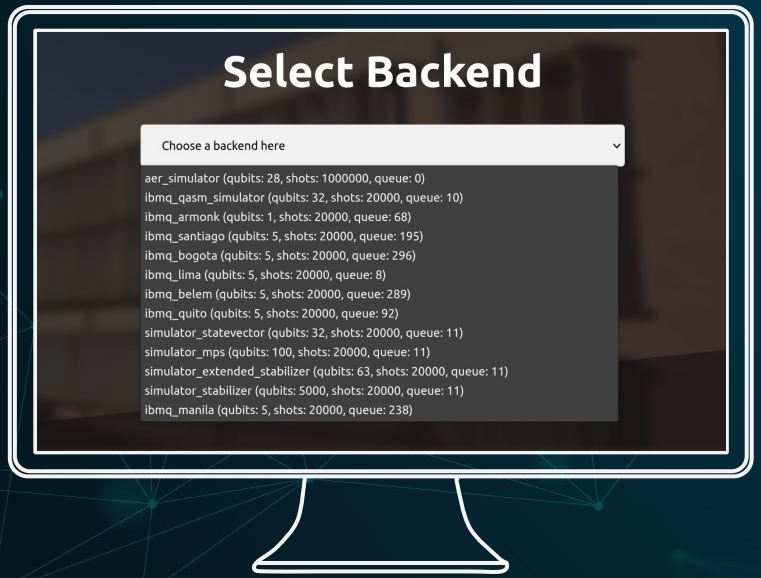

```
QuantumSolver
=============

A little quantum toolset developed using Qiskit
WARNING: The toolset uses your personal IBM Quantum Experience
token to access to the IBM hardware.
You can access to your API token or generate another one here:
https://quantum-computing.ibm.com/account

You can also use the Guest Mode which only allows you to run
quantum circuits in a local simulator ("aer_simulator").

[&] Write your IBM Quantum Experience token (Or Press Enter for Guest Mode):
✓ Loading Guest Mode
```

MENÚ PRINCIPAL



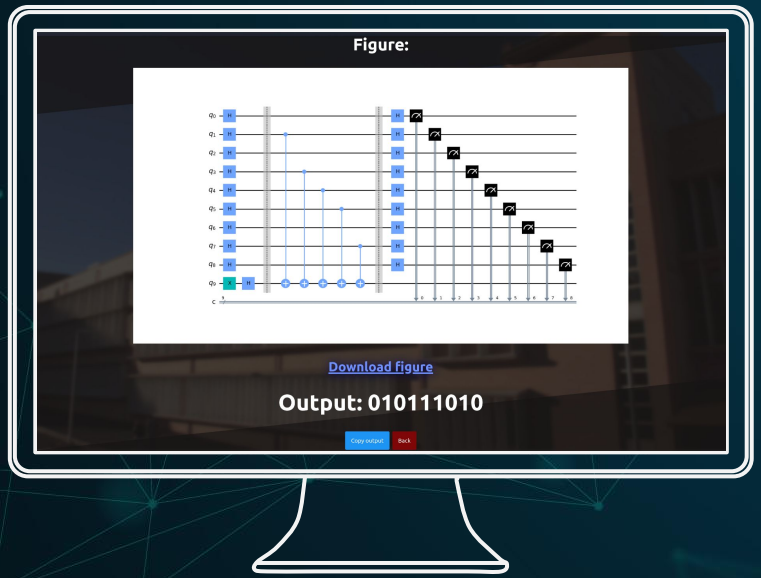

```
QuantumSolver
=============

A little quantum toolset developed using Qiskit
WARNING: The toolset uses your personal IBM Quantum Experience
token to access to the IBM hardware.
You can access to your API token or generate another one here:
https://quantum-computing.ibm.com/account

You can also use the Guest Mode which only allows you to run
quantum circuits in a local simulator ("aer_simulator").

[&] Write your IBM Quantum Experience token (Or Press Enter for Guest Mode):
✔ Loading Guest Mode

QuantumSolver (Guest Mode)
==========================

[1] See available Backends
[2] See available Algorithms
[3] Select Backend
[4] Select Algorithm
[0] Exit
```

MENÚ SELECCIÓN BACKEND




```
QuantumSolver
=============

[1] See available Backends
[2] See available Algorithms
[3] Select Backend
[4] Select Algorithm
[0] Exit

[&] Select an option: 3
[&] Select a backend of the list [1 - 13]: 3
[$] ibmq_armonk selected
```

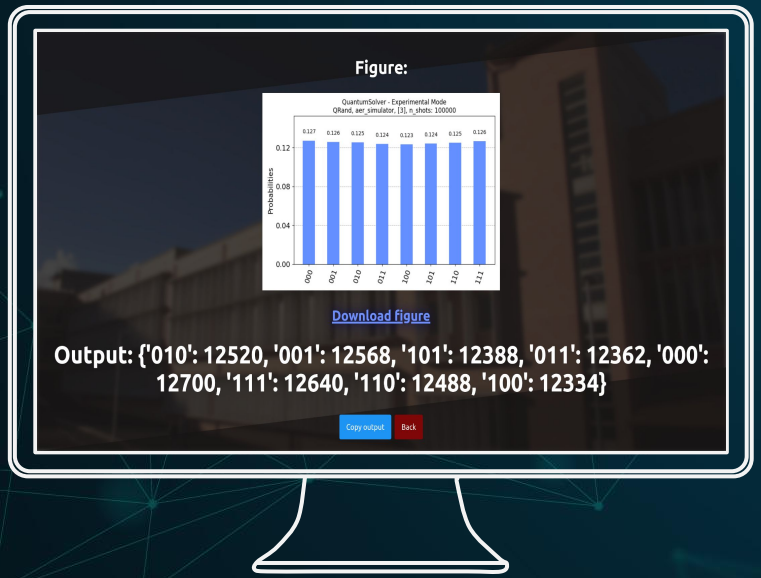
EJECUCIÓN SIMPLE




```
[&] Select an option: 6

✔ Creating circuit
  Circuit created in 2.1665096282958984 ms


Circuit visualization:

q_0: ┤H├┤H├┤M├
q_1: ┤H├┤H├┤M├
q_2: ┤H├┤H├┤M├
q_3: ┤H├┤H├┤M├
q_4: ┤H├┤H├┤M├
q_5: ┤H├┤H├┤M├
q_6: ┤H├┤H├┤M├
q_7: ┤H├┤H├┤M├
q_8: ┤H├┤H├┤M├
q_9: ┤X├┤H├┤X├┤X├┤X├┤X├┤X├
c: 9/                  0 1 2 3 4 5 6 7 8

✔ Executing Bernstein-Vazirani in aer_simulator with parameters: ['010111010']
  Execution done in 57.227134704589844 ms

💡 Output: 010111010
```

RECSI 2022: Santander 19-21 Octubre 2022

EJECUCIÓN MODO EXPERIMENTAL




```
[&] Select an option: 7
[&] Specify number of shots: 100000

Running Experiment:
✔ Executing QRand in aer_simulator with parameters: [3]

[$] Experiment Finished in 0.2721281051635742 s!

💡 Output: {'010': 12575, '110': 12465, '100': 12425, '101': 12580, '001': 12509, '111

QuantumSolver - Experimental Mode
################################################################################
                                                                  12575  010
                                                                  12465  110
                                                                  12425  100
                                                                  12580  101
                                                                  12509  001
                                                                  12441  111
                                                                  12669  000
                                                                  12336  011
```

RECSI 2022: Santander 19-21 Octubre 2022

ALGORITMOS IMPLEMENTADOS

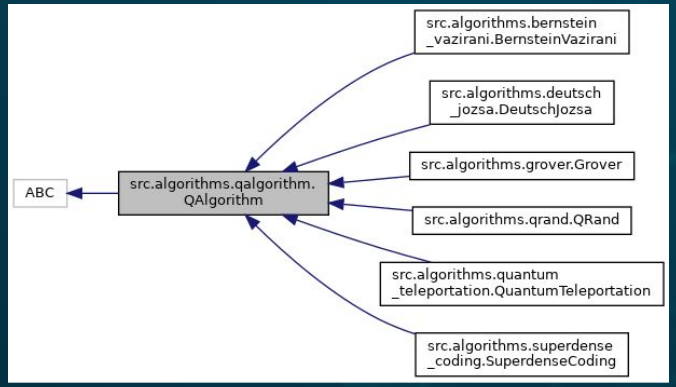
03

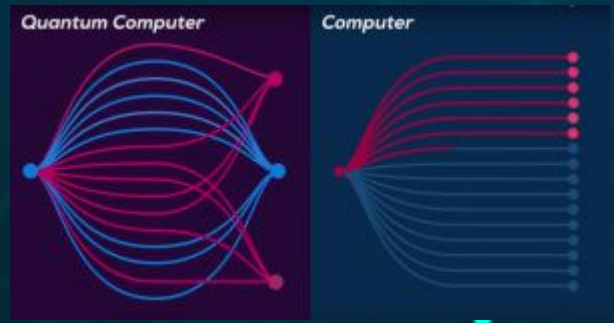
Explicación y circuitos parametrizados desarrollados

ALGORITMOS CUÁNTICOS DESARROLLADOS

- GENERACIÓN DE NÚMEROS ALEATORIOS
- BERNSTEIN-VAZIRANI
- BB84

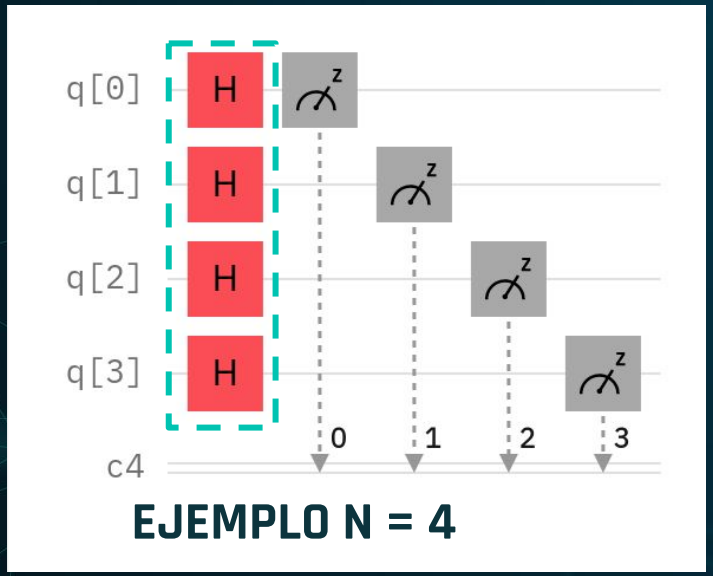
- DEUTSCH-JOZSA
- GROVER
- TELETRANSPORTE CUÁNTICO
- CODIFICACIÓN SUPERDENSA





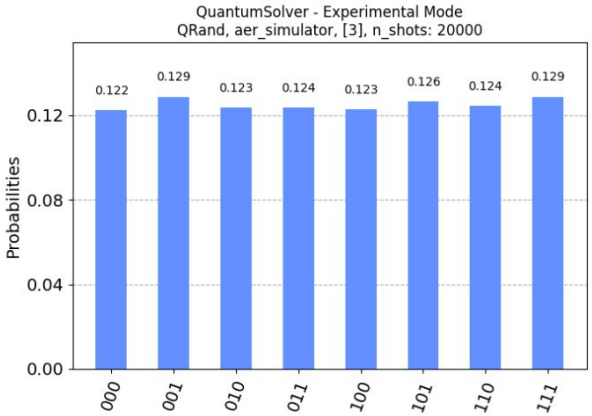
GENERACIÓN DE NÚMEROS ALEATORIOS - QRAND

ENTRADA: NÚMERO DE CÚBITS A UTILIZAR (N)
SALIDA: NÚMERO ALEATORIO ENTRE 0 Y (2 ^ N) - 1

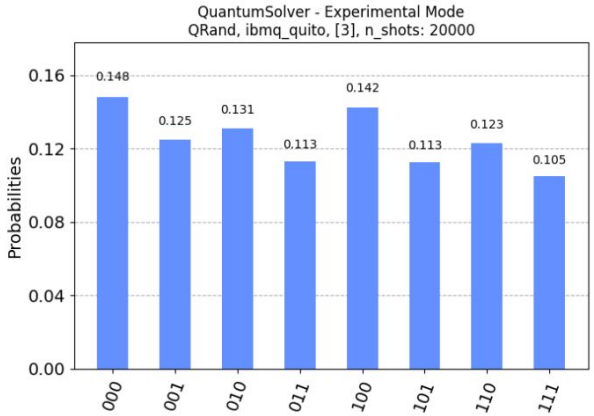



```python
## Create the QRand circuit
def circuit(self, n=1):
  circuit = QuantumCircuit(n, n)
  n_range = list(range(n))

  circuit.h(n_range)
  circuit.measure(n_range, n_range)
  return circuit
```

RESULTADOS QRAND



(a) Utilizando el backend “aer_simulator”

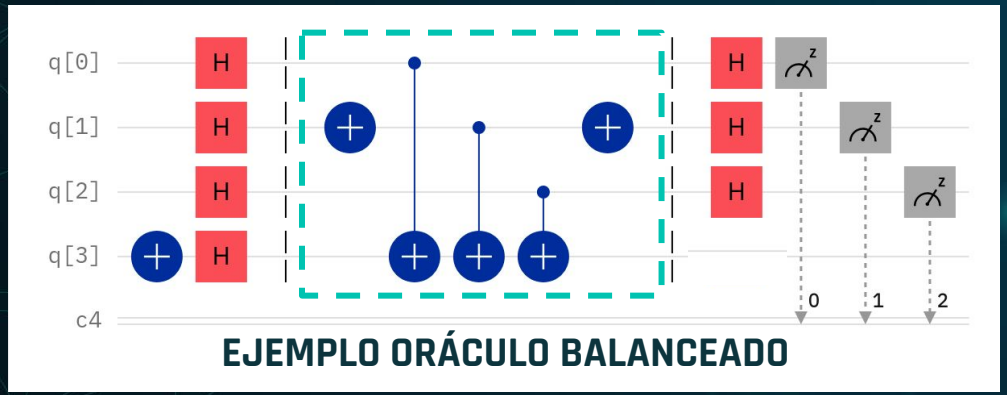


(b) Utilizando el backend “ibmq_quito”

Figura 4.1: Histograma tras 20.000 ejecuciones de QRand

ALGORITMO DEUTSCH-JOZSA

ENTRADA: ELECCIÓN ORÁCULO CONSTANTE O BALANCEADO
SALIDA: DETERMINAR EL TIPO DEL ORÁCULO
(00...0 -> CONSTANTE; EOC -> BALANCEADO)

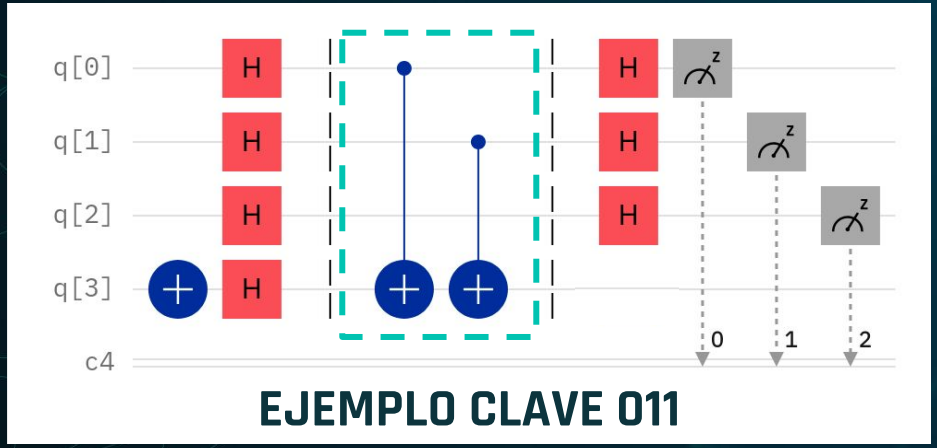


EJEMPLO ORÁCULO BALANCEADO

```python
if oracle_type == 'balanced':
    # Generate a random number that indicates which CNOTs to wrap in X-gates
    b = np.random.randint(1, 2 ** n)
    # Format 'b' as a binary string of length 'n', padded with zeros
    b_str = format(b, '0' + str(n) + 'b')

    # Each digit in the binary string corresponds to a qubit,
    # if it is 1 apply an X-gate to that qubit
    for i, current_char in enumerate(b_str):
        if current_char == '1':
            oracle_qc.x(i)

    # Do the controlled-NOT gates for each qubit,
    # using the output qubit as the target
    for qubit in range(n):
        oracle_qc.cx(qubit, n)

    # Place the final X-gates
    for i, current_char in enumerate(b_str):
        if current_char == '1':
            oracle_qc.x(i)
```

ALGORITMO BERNSTEIN-VAZIRANI

ENTRADA: CLAVE A CODIFICAR EN EL ORÁCULO
SALIDA: CLAVE ADIVINADA

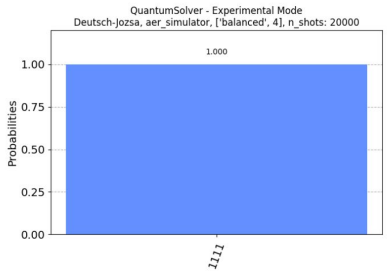


EJEMPLO CLAVE 011

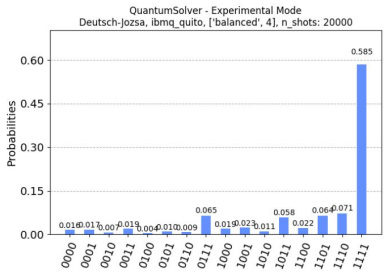
```python
def create_oracle(self, circuit, n):
  for i, char in enumerate(reversed(secret_number)):
    if char == '1':
      circuit.cx(i, n)
  return circuit
```

RESULTADOS DEUTSCH-JOZSA Y BERNSTEIN-VAZIRANI

100.0% 58.5% 100.0% 61.0%

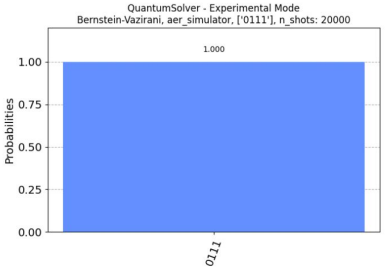


(a) Utilizando el backend “aer_simulator”

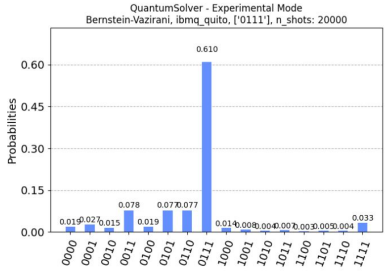


(b) Utilizando el backend “ibmq_quito”

Figura 4.2: Histograma tras 20.000 ejecuciones de Deutsch-Jozsa



(a) Utilizando el backend “aer_simulator”



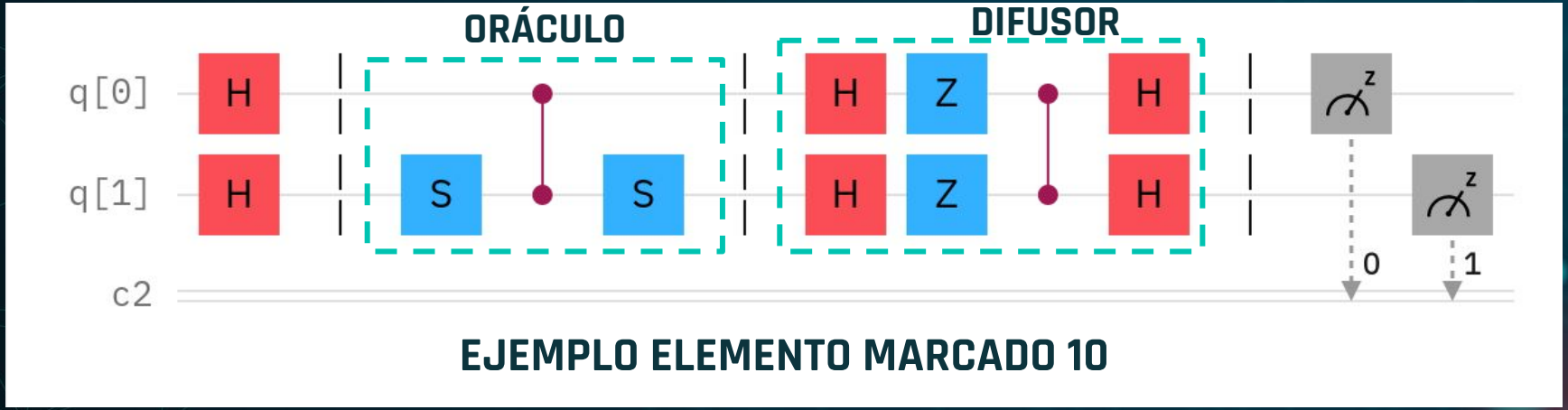
(b) Utilizando el backend “ibmq_quito”

Figura 4.3: Histograma tras 20.000 ejecuciones del algoritmo Bernstein-Vazirani

DEUTSCH-JOZSA BERNSTEIN-VAZIRANI

ALGORITMO DE GROVER (PARA DOS CÚBITS)

ENTRADA: ELEMENTO MARCADO EN LA BASE DE DATOS
SALIDA: ELEMENTO ENCONTRADO

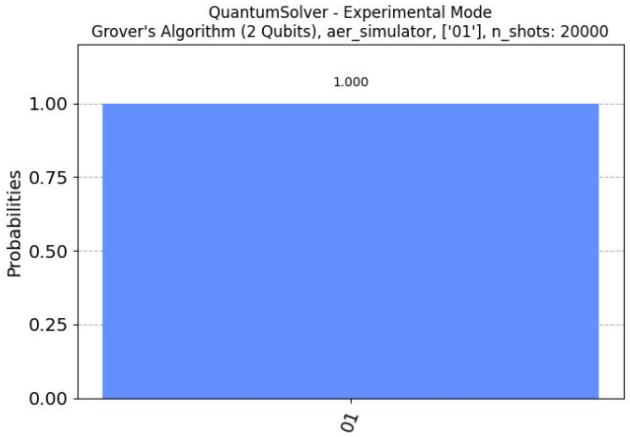


EJEMPLO ELEMENTO MARCADO 10

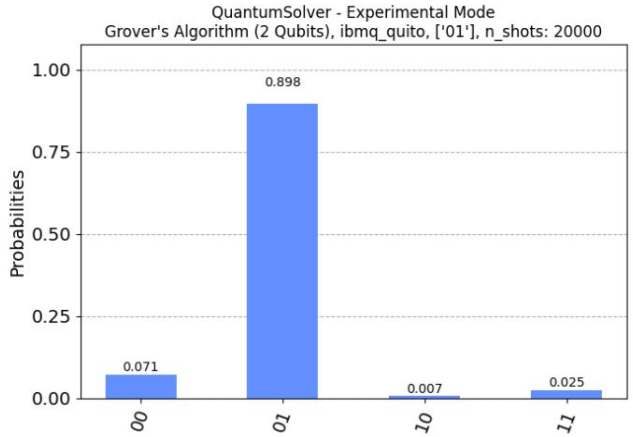
RESULTADOS GROVER

100.0%

89.8%



(a) Utilizando el backend “aer_simulator”

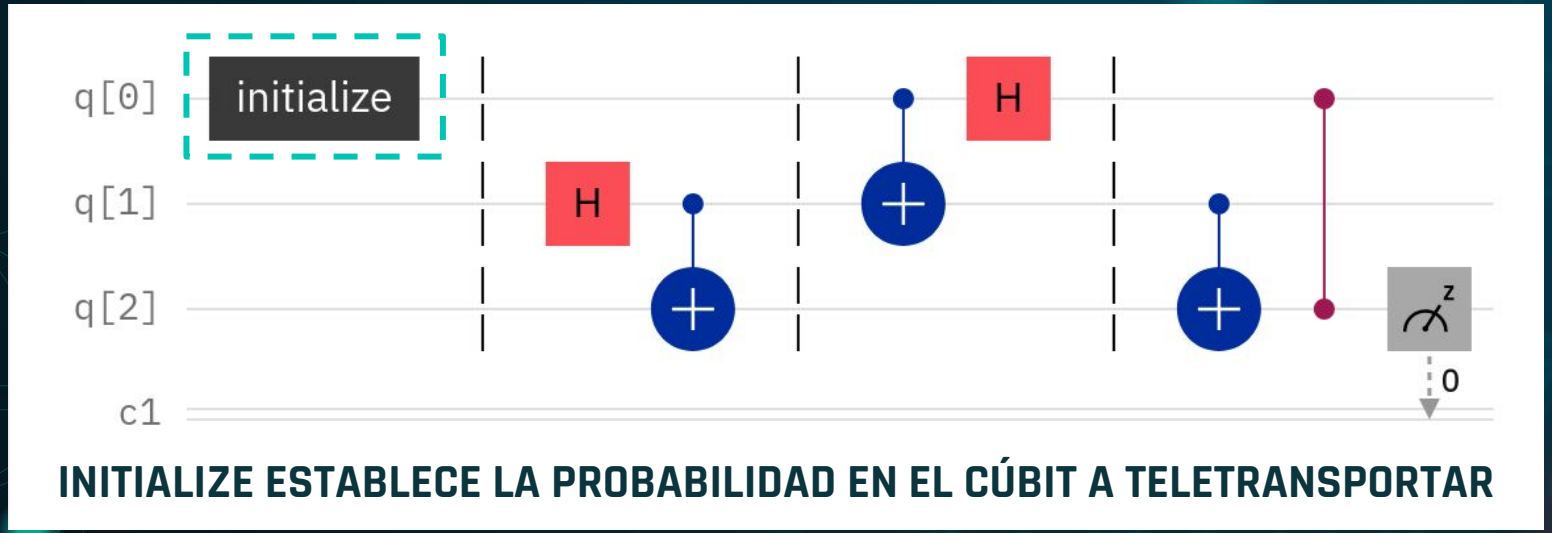


(b) Utilizando el backend “ibmq_quito”

Figura 4.4: Histograma tras 20.000 ejecuciones del algoritmo de Grover

TELETRANSPORTE CUÁNTICO

ENTRADA: PROBABILIDAD DE MEDIR CERO PARA EL CÚBIT A TELETRANSPORTAR
SALIDA: MEDICIÓN DEL CÚBIT TELETRANSPORTADO

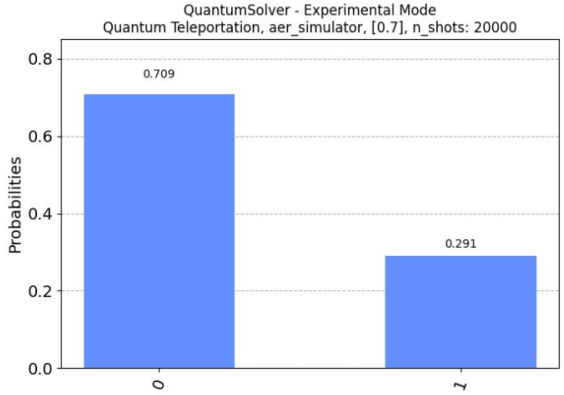


INITIALIZE ESTABLECE LA PROBABILIDAD EN EL CÚBIT A TELETRANSPORTAR

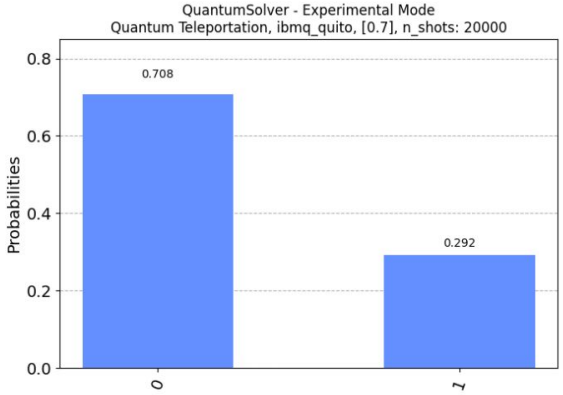
RESULTADOS TELETRANSPORTE CUÁNTICO

~100.0%

~100.0%



(a) Utilizando el backend “aer_simulator”

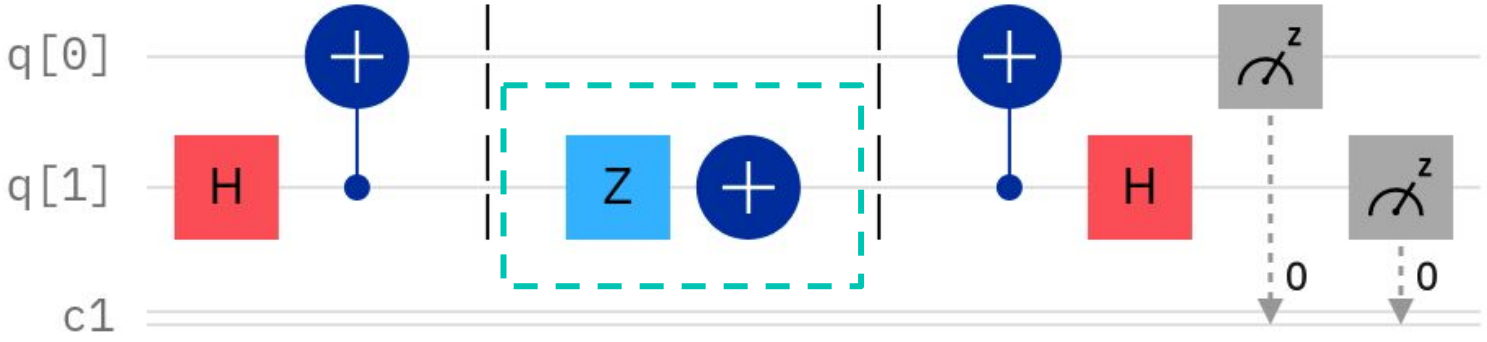


(b) Utilizando el backend “ibmq_quito”

Figura 4.5: Histograma tras 20.000 ejecuciones del algoritmo de teletransporte cuántico

PROTOCOLO DE CODIFICACIÓN SUPERDENSA

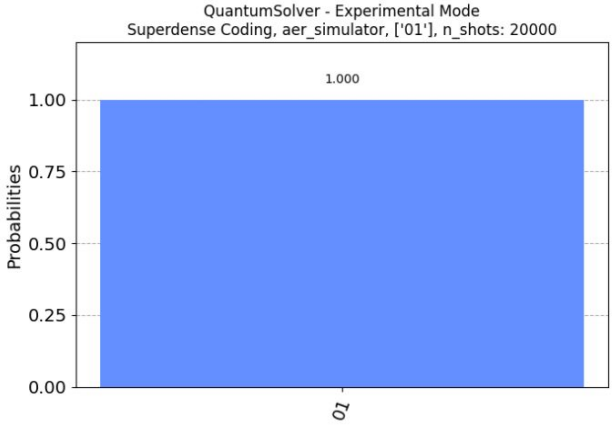
ENTRADA: 2 BITS CLÁSICOS A TRANSMITIR
SALIDA: 2 BITS CLÁSICOS TRANSMITIDOS



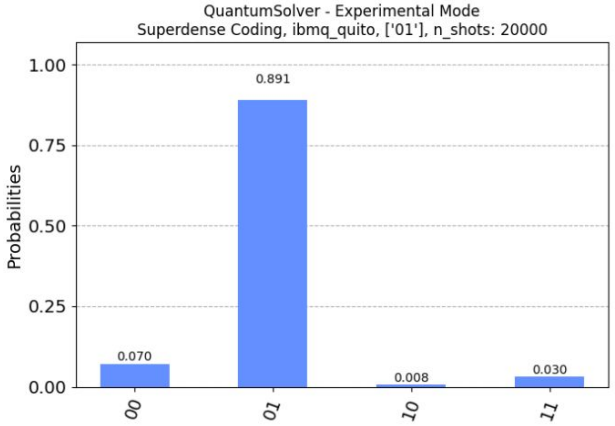
EJEMPLO PARA EL MENSAJE 11

RESULTADOS CODIFICACIÓN SUPERDENSA

100.0% **89.1%**



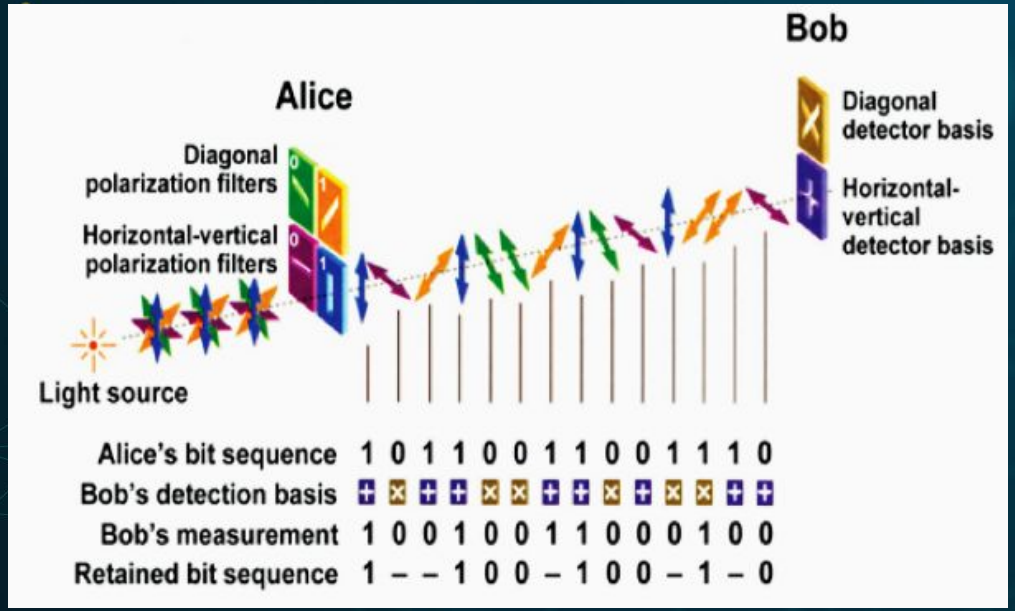
(a) Utilizando el backend “aer_simulator”



(b) Utilizando el backend “ibmq_quito”

Figura 4.6: Histograma tras 20.000 ejecuciones del protocolo de codificación superdensa

IMPLEMENTACIÓN DEL PROTOCOLO BB84 (QKD)



PROGRAMA PRINCIPAL BB84

```
BB84 SIMULATOR (Guest Mode)
===========================

[1] See available Backends
[2] Select Backend
        Current Backend: aer_simulator
[3] Run Algorithm
[4] Experimental mode
[0] Exit

[&] Select an option: 3
[&] Message (string): hola
[&] Interception Density (float between 0 and 1): 0
Running BB84 simulation in aer_simulator with message "hola" and density "0.0"
Alice Values:
[1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0
, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0,
0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0]

Alice Axes:
[0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1
, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0,
0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]

Eve Values:
```

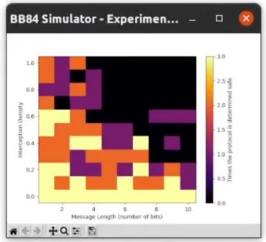
RECSI 2022: Santander 19-21 Octubre 2022

ACEPTACIÓN BB84

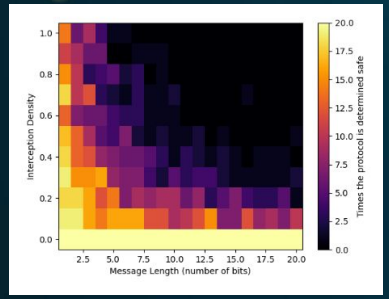
```
Secure Communication!

Alice OTP:
[32, 11, 8, 35, 1]

Bob OTP:
[32, 11, 8, 35, 1]

Initial Message:
hola
Encoded Message:
HddB
💡 Decoded Message:
hola

✅ The initial message and the decoded message are identical
✔ Running BB84 simulation in aer_simulator with message "hola" and density "0.0"
  BB84 simulation runned in 94.07424926757812 ms
```

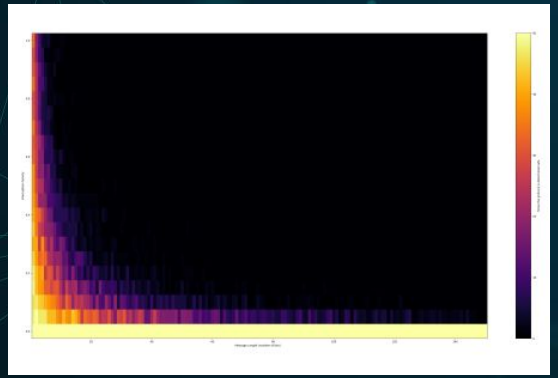
MODO EXPERIMENTAL BB84

```
[&] Select an option: 4
[&] Specify maximum message length (number of bits): 10
[&] Specify density step: 0.1
[&] Specify number of repetitions for each instance: 3

Running BB84 Simulator Experiment (in aer_simulator):
|                                                  | 330/330 [100%] in 9.9s (33.48/s)

[$] Experiment Finished in 9.89268159866333 s!

💡 Output:

x: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

y: [0.0, 0.1, 0.2, 0.30000000000000004, 0.4, 0.5, 0.6000000000000001, 0.7000000000000001, 0.8, 0.9, 1.0]

Image:
[[3. 3. 3. 3. 3. 3. 3. 3. 3. 3.]
 [3. 2. 3. 2. 2. 3. 2. 2. 3. 1.]
 [3. 3. 1. 1. 2. 1. 3. 1. 2. 1.]
 [2. 2. 1. 2. 2. 2. 2. 0. 0. 0.]
 [2. 2. 3. 3. 1. 1. 3. 0. 1. 0.]
 [3. 2. 2. 2. 1. 1. 0. 0. 0. 0.]
 [3. 3. 2. 0. 0. 0. 0. 1. 1. 1.]
 [2. 0. 1. 1. 0. 0. 0. 0. 0. 0.]
 [1. 1. 2. 1. 0. 0. 0. 0. 0. 0.]
 [2. 1. 0. 1. 0. 0. 0. 0. 0. 0.]
 [2. 1. 2. 0. 0. 0. 0. 0. 0. 0.]]
```



MODO EXPERIMENTAL BB84





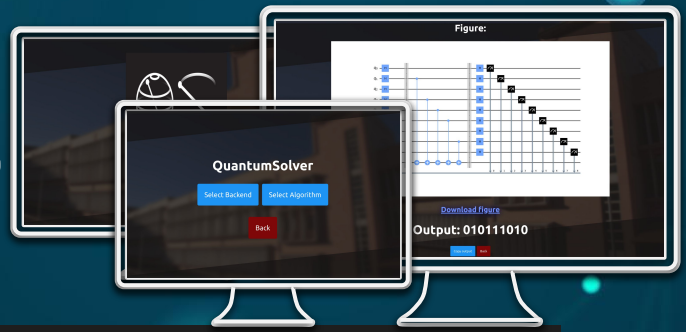
PARÁMETROS RELEVANTES DE LOS MAPAS DE CALOR

Longitud máxima del mensaje	*Step* de densidad de intercepción	Repeticiones de cada instancia	Tiempo de ejecución (aprox.)
20 bits	0.1	20	5 minutos
150 bits	0.05	50	16 horas

04 CONCLUSIONES

Resultados obtenidos

CONCLUSIONES



- LIBRERÍA CUÁNTICA USANDO QISKIT
- DOS INTERFACES: LÍNEA DE COMANDOS Y WEB
- ALGORITMOS PREDEFINIDOS
- DESARROLLO CUÁNTICO
- DE FÁCIL CONTRIBUCIÓN
- GRÁFICAS E INTERFACES INTUITIVAS

REFERENCIAS

05

Bibliografía consultada

REFERENCIAS

[1] IBM, “Learn Quantum Computation using Qiskit” [Online]. Available: https://qiskit.org/textbook/preface.html. [Accessed: 20-Oct-2022].
[2] IBM, “Qiskit”. [Online]. Available: https://qiskit.org/. [Accessed: 20-Oct-2022].
[3] D. Escánez-Expósito, “QuantumSolver”. [Online]. Available: https://github.com/alu0101238944/quantum-solver/. [Accessed: 20-Oct-2022].
[4] D. Escánez-Expósito, “QuantumSolver - Documentation”. [Online]. Available: https://alu0101238944.github.io/quantum-solver/. [Accessed: 20-Oct-2022].
[5] IBM, “Single qubit Gates” [Online]. Available: https://qiskit.org/textbook/ch-states/single-qubit-gates.html. [Accessed: 20-Oct-2022].
[6] IBM, “Quantum Key Distribution” [Online]. Available: https://qiskit.org/textbook/ch-algorithms/quantum-key-distribution.html. [Accessed: 20-Oct-2022].

RECSI 2022: Santander 19-21 Octubre 2022

¡GRACIAS!

¿Alguna pregunta?

Daniel Escánez-Expósito
Universidad de La Laguna
Tenerife, Spain
jdanielescanez@gmail.com

Pino Caballero-Gil
Universidad de La Laguna
Tenerife, Spain
pcaballe@ull.edu.es

Francisco Martín-Fernández
IBM Research
NY, USA
paco@ibm.com