Implementación de los Algoritmos Cuánticos de Simon y de Shor

Vlatko Marchan-Sekulic
Universidad de La Laguna
Tenerife, Spain
89vlatko@gmail.com

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

Daniel Escanez-Exposito
Universidad de La Laguna
Tenerife, Spain
jdanielescanez@gmail.com

***Resumen*—En el presente trabajo se expone la implementación realizada de los algoritmos de Simon y de Shor así como su inclusión a la librería de código abierto para el desarrollo de software cuántico *QuantumSolver*. Para ello se ha introducido un módulo llamado *QuantumSolver Subroutine*, que abre nuevas posibilidades a futuras incorporaciones de implementaciones en la librería. El principal objetivo de este trabajo es acercar la computación cuántica de una forma accesible y atractiva a todos los públicos, destacando además los avances más importantes que se están logrando en esta tecnología. Por ejemplo, *QuantumSolver* no solo permite poner el foco en cómo la computación cuántica puede resolver problemas más rápido que la computación clásica, sino también en cómo esto puede afectar a las formas actuales de proteger la información. En este sentido, entre los elementos cardinales destacan los algoritmos cuánticos de Simon y Shor por su capacidad de resolver problemas particularmente difíciles para la computación clásica, que implican la ruptura de esquemas criptográficos de gran despliegue en las tecnologías actuales, como *RSA* o *Diffie–Hellman*.**

***Index Terms*—Algoritmo de Shor, Algoritmo de Simon, Transformada cuántica de Fourier, Qiskit, Computación cuántica, RSA, Computación híbrida.**

Tipo de contribución: *investigación en desarrollo.*

I. Introducción.

En el ultimo lustro, la computación cuántica ha adquirido un gran protagonismo como principal amenaza contra los protocolos actuales. El grupo de cifrados más afectados son los de clave pública, como *RSA* y los basados en curvas elípticas [1]. Estos tienen un gran despliegue tecnológico ya que son usados, por ejemplo, para proteger las comunicaciones en Internet y en mensajería instantánea [2]. Por tanto, el desarrollo de ordenadores cuánticos con suficientes cúbits, significaría un gran problema para la sociedad, dado que las comunicaciones realizadas en Internet, así como los datos personales involucrados, quedarían desprotegidos. Sin embargo, para poder realizar un ataque a *RSA* con el algoritmo de Shor, se estima que sería necesario un dispositivo cuántico de, al menos, un millón de cúbits [3].

Según la hoja de ruta de desarrollo de *IBM Quantum* [5], a finales del año 2023 se espera el lanzamiento de su nuevo ordenador cuántico llamado *Condor*, el cual tendrá una capacidad de 1.121 cúbits. Este procesador será el primero en el mundo con más de mil cúbits que podrá realizar cálculos generales de computación universal. Además, se ha establecido el año 2033 [6] como la fecha objetivo para el lanzamiento de un ordenador cuántico con una capacidad superior a 100.000 cúbits.

Por ello, se ha decidido crear un nuevo módulo en la librería de código abierto para el desarrollo de software cuántico *QuantumSolver* [4], La librería se centra en garantizar que la sociedad tenga acceso a las nuevas tecnologías cuánticas sin requerir amplios conocimientos técnicos sobre el tema. Por ello, el objetivo de la inclusión del nuevo módulo, llamado *QuantumSolver Subroutine*, es permitir la implementación de algoritmos como Simon y Shor para que los usuarios puedan entender su funcionamiento y experimentar con ellos. Dichos algoritmos necesitan un post-procesamiento clásico de los resultados de la subrutinas cuántica ejecutadas. De hecho, en la comunidad científica ya se admite el potencial de las técnicas híbridas de computación cuántica-clásica, pues permiten la ejecución de algoritmos que combinan las mejores cualidades de los paradigmas clásicos y cuánticos de la computación.

Para la implementación de los algoritmos cuánticos, que se exponen en el presente artículo, se han seguido las referencias [7]–[9]. En estas se explica en profundidad el funcionamiento de los módulos presentes en el kit de desarrollo cuántico *Qiskit* desarrollado por *IBM* [10] y su utilización para la ejecución del código en hardware cuántico real.

Este trabajo se estructura de la forma siguiente. La sección II introduce el algoritmo de Simon y la implementación que se ha desarrollado del algoritmo. La sección III introduce el algoritmo de Shor, prestando especial atención a la transformada cuántica de Fourier, y describiendo brevemente la implementación desarrollada del algoritmo. Finalmente, la sección IV cierra el trabajo con algunas conclusiones.

II. Algoritmo de Simon

El problema de Simon [11] fue planteado como caso particular del problema del subgrupo oculto abeliano, subgrupo que tiene como característica que al cambiar orden de los elementos que se suman o multiplican, el resultado que se obtiene es el mismo. Por ello, Simon permite encontrar el subgrupo oculto dentro de un grupo dado con la realización de consultas a una función que es periódica con respecto al grupo [12]. Este fue el primer algoritmo cuántico en mostrar una aceleración exponencial en comparación con el mejor algoritmo clásico para resolver un problema específico. Además, inspiró los algoritmos cuánticos basados en la transformada cuántica de Fourier, como el algoritmo de factorización de Shor.

II-A. Definición del problema

El problema de Simon se focaliza en determinar qué condiciones prevalecen para una función f dada. Siendo $f : \{0,1\}^n \rightarrow \{0,1\}^n$ se garantiza sea inyectiva o sobreyectiva :

1. Inyectiva $(1:1)$: La función f asigna exactamente una salida única para cada entrada, como muestra la Ec. (1).

ISBN:978-84-8158-970-2

2. Sobreyectiva $(2:1)$: La función f asigna a una única salida dos entradas. Para conseguir este efecto, existe una cadena b no trivial que cumple la Ec. (2) donde el símbolo $\oplus$ denota la operación binaria *XOR*.

$$\forall x \in \{0,1\}^n, \nexists y \in \{0,1\}^n, f(x) = f(y) \wedge y = x \oplus b \quad (1)$$

$$\forall x, y \in \{0,1\}^n, f(x) = f(y) \iff y = x \oplus b \quad (2)$$

Un ejemplo sencillo de los dos posibles funcionamientos de la función f se puede observar en la Tabla I.

Tabla I
EJEMPLO DE FUNCIONAMIENTO DE LA FUNCIÓN F CON b DE TAMAÑO 2.

(1:1)	(2:1)
$F(0) \to 0$	$F(0) \to 0$
$F(1) \to 1$	$F(1) \to 1$
$F(2) \to 2$	$F(2) \to 0$
$F(3) \to 3$	$F(3) \to 1$

El algoritmo de Simon consta de dos partes diferenciadas: la subrutina cuántica, encargada de buscar el periodo de la función f y el algoritmo clásico especializado en resolver el sistema de ecuaciones resultante de la etapa anterior, para conseguir el valor de b. El oráculo Q_f que codifica el funcionamiento de f en la subrutina cuántica debe tener unas características específicas que se comentan a continuación.

II-A1. Características del oráculo: Al caracterizar una función que cumpla los requisitos antes descritos y que haga el efecto de *caja negra* se deben seguir una serie de pasos que se definirán en este apartado.

El oráculo recibe $|x\rangle |0\rangle^{\otimes n}$ como entrada y en base a un b predeterminado, el oráculo escribe su salida en el segundo registro de modo que transforma la entrada a $|x\rangle |f_b(x)\rangle$, aplicando de esta manera la función f, debido a que se define como $f_b(x) = f(x \oplus b)$ para todo $x \in \{0,1\}^n$. Para conseguir este efecto es necesario realizar los siguientes pasos:

1. Consiste en replicar el contenido del primer registro en el segundo $|x\rangle |0\rangle \to |x\rangle |x\rangle$. Esto se puede hacer debido a que los cúbits de entrada solo pueden estar en los estados base $|0\rangle$ o $|1\rangle$ y aplicando la operación *CX*.
2. Consiste en aplicar el comportamiento de f emulando la función inyectiva (1:1) o sobreyectiva (2:1). Esto se consigue viendo la forma de la cadena b pasada al oráculo definir su comportamiento.
   a) En caso de que b sea todo ceros no se aplica ninguna operación extra.
   b) En caso de que b no sea todo ceros implica que existe un índice mínimo j tal que $b_j = 1$. Conociendo esto se realiza una operación *XOR* con el segundo registro con b Ec. (3).
3. Consiste en aplicar permutaciones aleatorias en el segundo registro. Para generar el efecto de caja negra.

$$\text{Si } x_j \text{ en } b_j = 1 : |x\rangle |x\rangle \to |x\rangle |x \oplus b\rangle \quad (3)$$

II-B. Desarrollo del algoritmo

A continuación se desarrolla una explicación matemática formal sobre la evaluación del estado cuántico en la ejecución del circuito cuántico de Simon descrito en la Fig. 1.

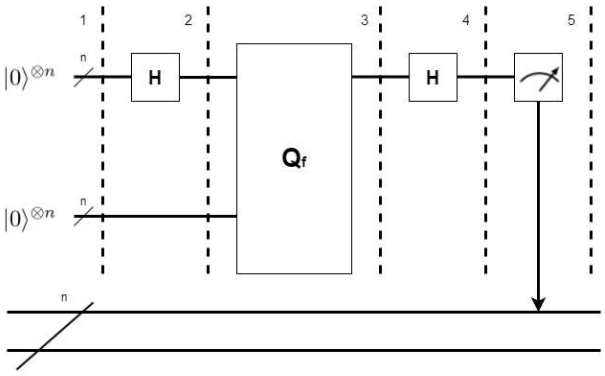

Figura 1. Circuito cuántico de Simon

1. Para una entrada de n-qbits los registros se inicializan a ceros (véase Ec. (4)).

$$|\psi_1\rangle = |0\rangle^{\otimes n} |0\rangle^{\otimes n} \quad (4)$$

2. Se aplica la puerta Hadamard al primer registro, obteniendo Ec. (5).

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |0\rangle^{\otimes n} \quad (5)$$

3. El comportamiento general del oráculo sobre un registro cuántico arbitrario viene dado por la Ec. (6). Además, sobre el estado $|0\rangle^{\otimes n}$ se comporta estableciendo ese registro a la evaluación de la función sobre la entrada, como se describe en la Ec. (7). Tras realizar las transformaciones descritas, el oráculo Q_f opera sobre ambos registros dando como resultado Ec. (8).

$$Q_f(|x\rangle |a\rangle) = |x\rangle |a \oplus f(x)\rangle \quad (6)$$

$$Q_f(|x\rangle |0\rangle^{\otimes n}) = |x\rangle |f(x)\rangle \quad (7)$$

$$|\psi_3\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle \quad (8)$$

Una vez aplicado el oráculo se consigue que el primer registro pueda llegar a tomar dos valores posibles $|x\rangle$ o $|y\rangle$ (véase Ec. (9)) dando como resultado la expresión Ec. (10).

$$\begin{cases} x \\ y \quad = x \oplus b \end{cases} \quad (9)$$

$$|\psi_4\rangle = \frac{1}{\sqrt{2}} (|x\rangle + |y\rangle) \quad (10)$$

4. Se aplica nuevamente una puerta de Hadamard al primer registro, obteniendo con ello Ec. (11).

$$|\psi_5\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{z\in\{0,1\}^n} [(-1)^{x\cdot z} + (-1)^{y\cdot z}] \, |z\rangle \quad (11)$$

5. El último paso del circuito es medir el primer registro dando una salida únicamente si se cumple la expresión Ec. (12).

$$\begin{aligned} (-1)^{x\cdot z} &= (-1)^{y\cdot z} \\ x\cdot z &= y\cdot z \\ x\cdot z &= (x\oplus b)\cdot z \\ x\cdot z &= x\cdot z \oplus b\cdot z \\ b\cdot z &= 0 \pmod 2 \end{aligned} \quad (12)$$

Una vez se extrae el resultado de la subrutina cuántica se procede a realizar el procesado clásico del algoritmo de Simon. En el resultado de la rutina se medirá una cadena z cuyo producto interno con b es igual a cero. Repitiendo el algoritmo aproximadamente n veces, se podrán obtener n valores diferentes de z y se podrá escribir el sistema de ecuaciones definido en la expresión Ec. (13).

$$\begin{cases} b\cdot z_1 = 0 \\ b\cdot z_2 = 0 \\ \quad\vdots \\ b\cdot z_n = 0 \end{cases} \quad (13)$$

A partir de este sistema de ecuaciones es posible despejar y obtener el valor de b. Existen múltiples opciones para resolver el sistema de ecuaciones resultante.

II-C. Resultados

Dentro de los resultados de la ejecución se puede observar claramente la diferencia del comportamiento del oráculo Q_f según si nos encontramos en el caso que la función f sea inyectiva (1:1) o sobreyectiva (2:1).

- Inyectiva (1:1): El circuito resultante generado por Q_f se puede observar en la Fig. 2 dando como resultado tras su ejecución el sistema de ecuaciones Ec. (14) que al ser resuelto se obtiene $b = 000$.
- Sobreyectiva (2:1): El circuito resultante generado por Q_f se puede observar en la Fig. 3 definiendo como resultado al ser ejecutado el sistema de ecuaciones Ec. (15) que al ser resuelto brinda un valor de $b = 101$.

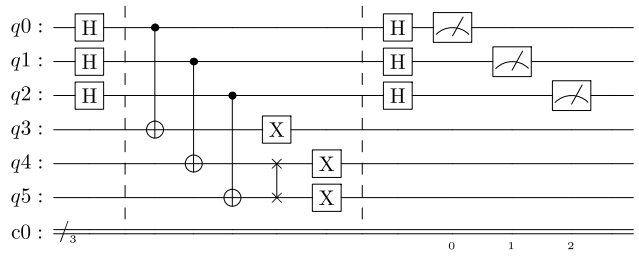

Figura 2. Ejecución con $b = 000$

$$\begin{aligned} b\cdot 101 &= b0\cdot 1 + b1\cdot 0 + b2\cdot 1 = 0 \pmod 2 \\ b\cdot 010 &= b0\cdot 0 + b1\cdot 1 + b2\cdot 0 = 0 \pmod 2 \\ b\cdot 001 &= b0\cdot 0 + b1\cdot 0 + b2\cdot 1 = 0 \pmod 2 \\ b\cdot 111 &= b0\cdot 1 + b1\cdot 1 + b2\cdot 1 = 0 \pmod 2 \\ b\cdot 011 &= b0\cdot 0 + b1\cdot 1 + b2\cdot 1 = 0 \pmod 2 \\ b\cdot 110 &= b0\cdot 1 + b1\cdot 1 + b2\cdot 0 = 0 \pmod 2 \\ b\cdot 100 &= b0\cdot 1 + b1\cdot 0 + b2\cdot 0 = 0 \pmod 2 \end{aligned} \quad (14)$$

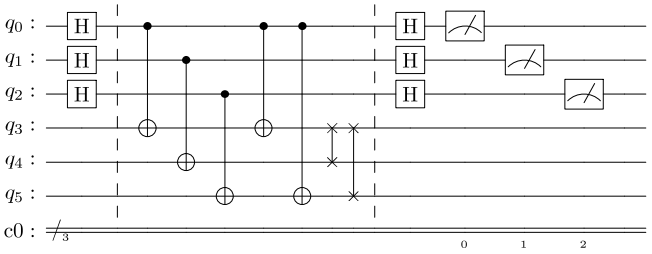

Figura 3. Ejecución con $b = 101$

$$\begin{aligned} b\cdot 101 &= b0\cdot 1 + b1\cdot 0 + b2\cdot 1 = 0 \pmod 2 \\ b\cdot 111 &= b0\cdot 1 + b1\cdot 1 + b2\cdot 1 = 0 \pmod 2 \\ b\cdot 010 &= b0\cdot 0 + b1\cdot 1 + b2\cdot 0 = 0 \pmod 2 \end{aligned} \quad (15)$$

El método elegido para la resolución de los sistemas de ecuaciones en la implementación fue la función $nullspace$ de la librería $sympy$. El método $nullspace$ se encarga de resolver el espacio nulo de una matriz A que es el conjunto de todos los vectores x que satisfacen la ecuación $Ax = 0$. Este problema es equivalente al sistema de ecuaciones Ec. (13) pudiendo interpretar el resultado de la siguiente manera: si el espacio nulo de una matriz es vacío, significa que no hay ningún vector x que satisfaga esta ecuación; por lo tanto, la solución será una b con todo ceros. Es decir, f se encuentra en el caso inyectiva (1:1). En caso contrario, se obtendrá el valor de b que resuelva el problema con una función f del tipo sobreyectiva (2:1).

III. Algoritmo de Shor

El algoritmo de Shor [13] resuelve el problema de la factorización de un número entero de una manera eficiente [14]. Algunos de los algoritmos criptográficos de clave pública más utilizados en la actualidad, como *RSA* [15], basan su funcionamiento en la multiplicación de dos números primos de gran tamaño. A continuación, se procederá a explicar en detalle el funcionamiento de *RSA* y la vulnerabilidad que origina el algoritmo de Shor.

1. Información privada
   - *a*) Dos números primos de gran tamaño que se llamarán p y q
   - *b*) $\Phi(N) = (p-1)\cdot(q-1)$
   - *c*) d es entero primo con $\Phi(N)$
2. Información pública
   - *a*) $N = p\cdot q$
   - *b*) e es el inverso de d módulo $\Phi(N)$
3. Operación de cifrado

   a) Siendo M_i el texto original
   b) $C_i = M_i^e \pmod N$
4. Operación de descifrado
   a) Siendo C_i el texto cifrado
   b) $M_i = C_i^d \pmod N$

Durante el proceso, los datos públicos serán utilizados por el emisor y el receptor para llevar a cabo el cifrado *RSA*. Entre ellos el valor de N, que al ser factorizado se logra obtener los datos privados p y q. La seguridad de este algoritmo de cifrado se basa, por tanto, en la alta complejidad para llevar a cabo esta factorización. Por este motivo, en el momento que el algoritmo de Shor sea ejecutado en un computador cuántico con los recursos necesarios, *RSA* y otros algoritmos de clave pública similares se verían obsoletos y vulnerados.

III-A. Definición del problema

El problema de la factorización de un entero N, consiste en encontrar los enteros entre 1 y N que sean divisores enteros de N. La aproximación al problema de la factorización seguida por el algoritmo de Shor se describe brevemente a continuación. Si r es el orden de un entero a módulo N, es porque es el menor entero positivo tal que $a^r = 1 \pmod N$. En ese caso, $a^{r+1} = a \pmod N$. Esto a su vez se corresponde con que r es el período de la función $f(x) = a^x \pmod N$, es decir, con el menor entero r tal que $f(x + r) = f(x)$. Por otra parte, si el orden r es par, se tiene que $a^r - 1 = (a^{r/2} - 1)(a^{r/2} + 1) = 0 \pmod N$, de donde se deduce que como N no puede dividir a $(a^{r/2}-1)$ ni a $(a^{r/2}+1)$, entonces N debe tener un factor común no trivial con $(a^{r/2} - 1)$ y $(a^{r/2} + 1)$, es decir, se logra factorizar N.

Así, la búsqueda de factores se realiza en el algoritmo de Shor mediante de dos partes diferenciadas:

1. Subrutina cuántica para resolver el problema de encontrar el periodo de una función.
2. Subrutina clásica para resolver el problema de encontrar el orden de un entero módulo otro, es decir, el menor exponente positivo tal que al elevar el primer entero a ese exponente se obtiene 1 en módulo el segundo entero.

III-B. Desarrollo del algoritmo

La subrutina cuántica del algoritmo de Shor permite encontrar el periodo de la función que define una exponenciación modular. Dicha subrutina consta de tres etapas:

1. A los cúbits de entrada se les aplica la puerta Hadamard para ponerlos en estado de superposición.
2. Se aplica el circuito encargado de la exponenciación modular que de manera general se describe como $a^r \pmod N$.
3. Se aplica la inversa de la transformada cuántica de Fourier $QFT^\dagger$ definida en la Fig. 5.

La función encargada de la exponenciación modular es la parte más compleja de la implementación del algoritmo de Shor. Esto se debe a que actualmente no se cuenta con una manera general factible para calcularla, aunque existen aproximaciones [16], las implementaciones se realizan mediante prueba y error. Uno de los factores principales que añaden dificultad es la cantidad de cúbits requeridos para el computo de la misma con respecto a la potencia de los ordenadores cuánticos actuales y el tiempo de coherencia de los mismos. La función de exponenciación modular utilizada ha sido la $a^r \pmod{21}$ [17]. El circuito se muestra en la Fig. 4.

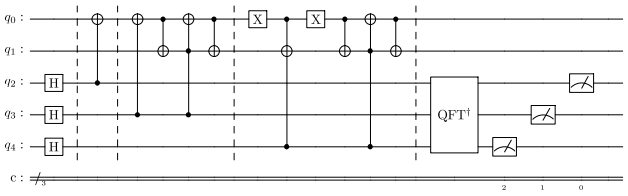

Figura 4. Circuito de Shor implementado para N=21

El algoritmo requiere de 5 cúbits: un registro de trabajo conformado por dos cúbits (q_0, q_1), y un registro de control con tres cúbits (q_2, q_3 y q_4). El registro de trabajo se representa mediante tres estados base de un sistema de dos cúbits, descartándose el cuarto estado base como estado nulo. Los estados que se codifican son $|1\rangle$, $|4\rangle$ y $|16\rangle$ y se mapean siguiendo la Ec. (16).

$$\begin{aligned} |1\rangle &\mapsto |log_4 1\rangle = |00\rangle \\ |4\rangle &\mapsto |log_4 4\rangle = |01\rangle \\ |16\rangle &\mapsto |log_4 16\rangle = |10\rangle \end{aligned} \tag{16}$$

Para calcular la exponenciación se emplean las puertas de control unitario $\hat{U}^x$ siguiendo la expresión Ec. (17) que reduce al intercambiar los estados $|1\rangle$, $|4\rangle$ y $|16\rangle$ en el registro de trabajo controlado por el correspondiente bit del entero x en el registro de control, que viene dado por $x = q_4 2^0 + q_3 2^1 + q_2 2^2$. Es decir, $\hat{U}^x = \hat{U}^{q_2 2^2} \hat{U}^{q_3 2^1} \hat{U}^{q_4 2^0}$ así dependiendo del cúbit de control q_i, se aplica una de las codificaciones definidas en Ec. (18). Las operaciones del registro de trabajo no necesitan ser puertas $SWAP(Fredkin)$ controladas.

$$|x\rangle |y\rangle \rightarrow |x\rangle \hat{U}^x |y\rangle = |x\rangle |a^x y \pmod N\rangle \tag{17}$$

$$\begin{aligned} \hat{U}^1 &: \{\ |1\rangle \mapsto |4\rangle, |4\rangle \mapsto |16\rangle, |16\rangle \mapsto |1\rangle\ \} \\ \hat{U}^2 &: \{\ |1\rangle \mapsto |16\rangle, |4\rangle \mapsto |1\rangle, |16\rangle \mapsto |4\rangle\ \} \\ \hat{U}^4 &: \{\ |1\rangle \mapsto |4\rangle, |4\rangle \mapsto |16\rangle, |16\rangle \mapsto |1\rangle\ \} \end{aligned} \tag{18}$$

Se procede a explicar cada una de las puertas $\hat{U}$ implementadas en la Fig. 4:

- $\hat{U}^1$ se simplifica mediante la observación que los estados $|4\rangle$ y $|16\rangle$ tienen inicialmente amplitud cero y por lo tanto la operación $|1\rangle \mapsto |4\rangle$ se puede realizar mediante el uso de la puerta CX controlada por $|q_4\rangle$ y dirigida al cúbit $|q_1\rangle$.
- $\hat{U}^2$ se puede simplificar observando los estados $|1\rangle$ y $|4\rangle$ ya que son los únicos estados de amplitud no nula en el registro de trabajo después que se haya aplicado $\hat{U}^1$, lo que conlleva considerar únicamente $|1\rangle \mapsto |16\rangle$ y $|4\rangle \mapsto |1\rangle$. Una puerta *CX* controlada por $|q_3\rangle$ dirigida a $|q_1\rangle$ seguida de una puerta *Swap*, intercambiando $|q_0\rangle$. y $|q_1\rangle$.
- $\hat{U}^4$ se realiza descomponiendo la puerta $Fredkin$ en una puerta *CCX (Toffoli)* y dos puertas *CX*. Esta puerta no admite simplificaciones ya que todos los posibles estados en el registro de trabajo pueden tener una amplitud distinta de cero en este punto. Esta operación se implementa

con una puerta *CCX* y una puerta *Swap* con puertas *X* de un cúbit.

III-C. Transformada cuántica de Fourier

La transformada cuántica de Fourier (QFT, Quantum Fourier Transform) [18] es la analogía cuántica de la transformada discreta de Fourier clásica. A continuación se compara el funcionamiento de ambas.

La transformada discreta de Fourier unitaria actúa sobre un vector en $\mathbb{C}^n$, $(x_0, \ldots, x_{n-1})$ y lo mapea al vector $(y_0, \ldots, y_{n-1})$ siguiendo la expresión Ec. (19), donde la expresión Ec. (20) es una raíz n-ésimas de la unidad primitiva.

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega_N^{jk} \tag{19}$$

$$\omega_N^{jk} = e^{2\pi i \frac{jk}{N}} \tag{20}$$

De manera similar, la transformada cuántica de Fourier actúa sobre un estado cuántico $|X\rangle = \sum_{j=0}^{N-1} x_j |j\rangle$ y lo mapea al estado cuántico $|Y\rangle = \sum_{k=0}^{N-1} y_k |k\rangle$ de acuerdo con la expresión Ec. (21).

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \omega_N^{jk} |k\rangle \tag{21}$$

Por otro lado, de forma equivalente, la transformada cuántica de Fourier puede verse como una matriz unitaria actuando sobre vectores estado cuántico, de acuerdo con la expresión Ec. (22).

$$U_{QFT} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} \omega_N^{jk} |k\rangle |x\rangle \tag{22}$$

Una de las propiedades más interesantes de la transformada cuántica de Fourier reside en el hecho que se trata de una transformación unitaria. Esto se puede verificar realizando una multiplicación de matrices y comprobando la relación $F \cdot F^\dagger = F^\dagger \cdot F = I$ donde F es la hermítica adjunta de F. Por ese motivo, la inversa de la QFT es igual a la $QFT^\dagger$.

En el caso del algoritmo de Shor se utiliza la implementación de la inversa de la transformada cuántica de Fourier ($QFT^\dagger$), cuya implementación se puede observar en la Fig. 5.

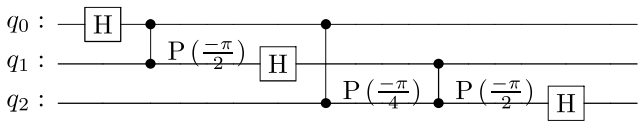

Figura 5. Ejemplo de circuito $QFT\dagger$

Una vez se concluye la subrutina cuántica se procede a realizar el post-procesado clásico del resultado, para lo cual se realiza lo siguiente:

1. Se convierten los números binarios en decimal d y se obtiene la fase realizando la operación $phase = d/2^{n\_count}$.
2. Se busca la fracción más cercana al valor de la $phase$ con un denominador menor a N. De esta fracción se utiliza el valor del denominador dando con ello el valor de r.
3. Por cada valor de r se comprueba que el valor no sea impar.
4. Se realiza el cálculo $mcd(a^{r/2} - 1, N)$ y $mcd(a^{r/2} + 1, N)$ dando con ello los dos factores que multiplicados entre sí dan 21, que en este caso son 7 y 3.

III-D. Resultados

La ejecución del circuito cuántico representado en la Fig. 4 da como resultados los mostrados en la Fig. 6. Dichos datos concuerdan con los resultados esperados pues al ser procesados devuelven los factores 7 y 3.

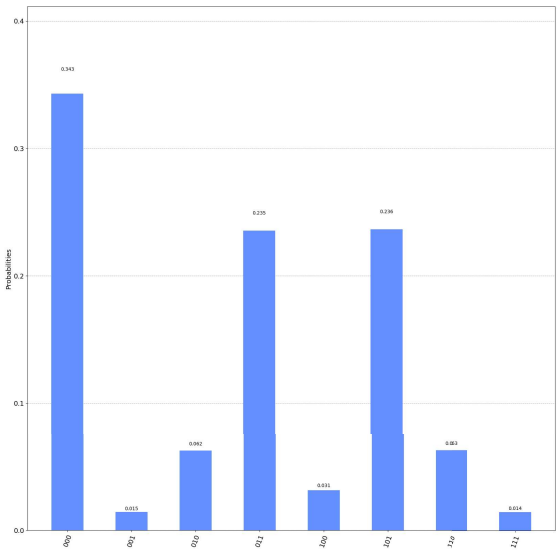

Figura 6. Resultados de la ejecución del algoritmo de Shor para N=21

IV. Conclusiones

En este trabajo se han presentado una introducción a los algoritmos de Simon y de Shor y la implementación de que de ellos se ha desarrollado para su inclusión en la librería de software cuántico *QuantumSolver* dentro del nuevo módulo *QuantumSolver Subroutine*. Los resultados obtenidos con dichas implementaciones se corresponden con los esperados, según las referencias bibliográficas consultadas. Como trabajo en proceso que es, son varias las líneas que se están desarrollando actualmente, tales como la adición de nuevos casos de factorización con el algoritmo de Shor, comenzando por los enteros $N = 33$ y $N = 55$, que ya están en proceso.

A pesar de anteriormente expuesto, existen una serie de algoritmos criptográficos resistentes a ataques cuánticos que podrían sustituir a *RSA* y al uso de las curvas elípticas. Dichas alternativas vienen descritas en el artículo publicado por el *National Institute of Standards and Technology (NIST)* [19] donde se anunciaron los primeros cuatro algoritmos criptográficos resistentes a ataques cuánticos. Tres de ellos son para firmas digitales, mientras que el restante es para cifrado

general. Este último es el algoritmo criptográfico *CRYSTALS-Kyber* [20] que se propone como el encargado de cifrar las comunicaciones de internet en el futuro.

AGRADECIMIENTOS

Esta investigación ha sido posible gracias a la Cátedra de Ciberseguridad de la Universidad de La Laguna.

REFERENCIAS

[1] Stewart I., Ilie D., Zamyatin A., Werner S., Torshizi M. F. and Knottenbelt W. J. 2018 Committing to quantum resistance: a slow defence for Bitcoin against a fast quantum computing attack R. Soc. open sci.5180410180410

[2] "The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446)". [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8446 [Accessed: 21-May-2023]

[3] "Are quantum computers about to break online privacy¿'. [Online]. Available: https://doi.org/10.1038/d41586-023-00017-0 [Accessed: 25-May-2023]

[4] Escanez-Exposito, D.: "QuantumSolver". [Online]. Available: https://github.com/jdanielescanez/quantum-solver.git [Accessed: 13-Apr-2023].

[5] "The IBM Quantum Development Roadmap". [Online]. Available: https://www.ibm.com/quantum/roadmap [Accessed: 25-May-2023].

[6] "Charting the course to 100,000 qubits". [Online]. Available: https://research.ibm.com/blog/100k-qubit-supercomputer. [Accessed: 1-June-2023]

[7] Norlén, H. Quantum Computing in Practice with Qiskit and IBM Quantum Experience: Practical recipes for quantum computer coding at the gate and algorithm level with Python. Packt Publishing Ltd. 2020.

[8] Wille, R., Van Meter, R., Naveh, Y. IBM's Qiskit tool chain: Working with and developing for real quantum computers. IEEE Design, Automation & Test in Europe Conference & Exhibition, pp. 1234-1240. 2019.

[9] Fortunato, D., Campos, J., Abreu, R. QMutPy: A mutation testing tool for quantum algorithms and applications in Qiskit. ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 797-800. 2022.

[10] "Qiskit Open-Source Quantum Development" . [Online]. Available: https://qiskit.org/ [Accessed: 05-Jan-2023].

[11] Simon, D.R: "On the power of quantum computation". *SIAM journal on computing* 26(5), pp. 1474-1483, 1997.

[12] Cai, G., Qiu, D.: "Optimal separation in exact query complexities for Simon's problem". *Journal of Computer and System Sciences* 97, pp. 83-93, 2018.

[13] Shor, P.W.: "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". *SIAM J. Comput* 26, pp. 1484–1509, 1997.

[14] Nielsen, M., Chuang, I.: "Quantum Computation and Quantum Information". *Cambridge Series on Information and the Natural Sciences*, Cambridge University Press, pp. 633, 2000.

[15] Rivest, R., Shamir, A., Adleman, L.: "RSA A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". *Communications of the ACM* 21(2), pp. 120-126, 1978.

[16] Ravuri, C.: "A General Implementation of Shor's Algorithm". [Online] Available: https://medium.com/mit-6-s089-intro-to-quantum-computing/a-general-implementation-of-shors-algorithm-da1595694430 [Accessed: 14-Apr-2023].

[17] Skosana, U., Tame, M.: "Demonstration of Shor's factoring algorithm for N =21 on IBM quantum processors". *Scientific reports* 11, 16599, 2021.

[18] Nielsen, M.A., Chuang, I.L.: "Quantum Computation and Quantum Information". *10th Anniversary Edition, Cambridge University Press*, pp. 217-220, 2011.

[19] NIST Announces First Four Quantum-Resistant Cryptographic Algorithms . [Online]. Available: https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms [Accessed: 13-Apr-2023].

[20] "CRYSTALS-Kyber". [Online]. Available: https://pq-crystals.org/kyber/index.shtml [Accessed: 13-Apr-2023].