

# Shor 演算法實作：在特定條件限制下 分解 4096-Bit 整數

Abel C. H. Chen  
Information & Communications Security Laboratory,  
Chunghwa Telecom Laboratories  
Taoyuan, Taiwan  
ORCID: 0000-0003-3628-3033

**摘要**—近幾年隨著量子晶片 Willow 等技術的發展，降低量子計算錯誤率，將有機會帶動量子計算進入實務應用的場域。因此，設計可以實務應用的量子演算法將會是重要的研究方向之一。在本研究中，主要聚焦在 Shor 演算法實作，改進模數計算效率，並且展示可以在特定條件限制下分解 4096-Bit 整數。在實驗結果中與最先進的(State-of-the-Art, SOTA)方法比較，證明可以提升數十倍效率，並且可以做到分解長度更長的整數。

**關鍵字**—Shor 演算法、質因數分解、量子計算、量子演算法、量子傅立葉變換。

## I. 前言

在 2024 年 12 月，Google 研究團隊在《Nature》上發表論文“Quantum error correction below the surface code threshold”，提出量子晶片 Willow，可以有效提供量子糾錯能力，降低錯誤率，從而帶動量子計算的發展[1]。因此，盤點現有的量子演算法，包含量子搜尋演算法 Grover 演算法[2]、量子質子因分解 Shor 演算法[3]等，嘗試把現有的量子演算法落實到真實應用場域將會是現階段的重要研究主軸之一。其中，Shor 演算法主要包含模數計算 (Modular Computation) 和逆量子傅立葉變換 (Inverse Quantum Fourier Transform, IQFT) 計算兩個部分，在質因數分解上可以通過量子計算特性來實現指數級加速，將可能威脅現行主流密碼學 RSA-2048 的安全性[4]。

有鑑於此，本研究在 Shor 演算法上進行改進，提出更有更高計算效率的模數計算量子電路，並且嘗試在特定條件限制下分解長度更長的整數。在符合特定條件限制下，本研究的主要貢獻條列如下。

1. 提出高效率的模數計算量子電路，與最先進的 (State-of-the-Art, SOTA) 方法[5]-[10]比較，證明可以提升二十倍的效率。
2. 本研究方法可以用更少的量子位元數來分解長度更長的整數，包含分解 4096-Bit 整數。
3. 本研究採用 IBM Qiskit 實作量子電路和量子演算法，通過實作來驗證本研究提出方法的可行性及其效率。

本研究主要分為五個小節。第 II 節介紹 Shor 演算法的基本流程，並且第 III 節介紹本研究提出的改進模數計算量子電路，並且搭配實例說明。第 IV 節說明實驗環境，並且與最先進的方法進行比較，提供比較討論。最後，第 V 節總結本研究貢獻，並且討論未來可行的研究方向。

## II. SHOR 演算法

本節將先說明 Shor 演算法的基本流程，再以近年文獻發表的例子來做討論。

### A. 演算法流程

圖 1 為 Shor 演算法的基本流程。假設要分解的整數是  $n$ -bit 整數  $N$  (即  $N \leq 2^n$ )，並且隨機選擇一個整數  $a$  作為底數(在圖 1 中  $a = 2$ )。其中， $p$  和  $q$  各別為質數， $N = pq$ ， $a$  和  $N$  的最大公因數必須為 1。產生兩個註冊器 (register) 各包含  $k$  個量子位元，總共  $2k$  個量子位元。其中， $q_0 \sim q_{k-1}$  為第 1 個註冊器的量子位元， $q_k \sim q_{2k-1}$  為第 2 個註冊器的量子位元，並且在後續計算在第 1 個註冊器和第 2 個註冊器之間存在量子糾纏態。在計算上，主要包含三個部分：模數計算、逆量子傅立葉變換計算、取得週期  $r$  和分解因數，可以用來分解  $n$ -bit 整數(其中  $n \leq k$ )，分述如下。

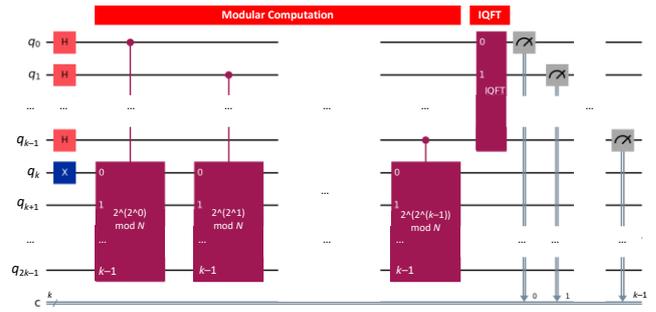


Fig. 1. Shor 演算法的基本流程

#### 1) 模數計算

在模數計算主要計算的值  $f(i) = a^i \pmod N$ ，其中  $0 \leq i \leq 2^k - 1$ ，並且把結果紀錄到第 2 個註冊器。根據文獻[3]，模數計算結果存在子循環，符合  $a^r \pmod N \equiv 1 \pmod N$ ，所以表示也存在  $a^j \pmod N \equiv a^{j+r} \pmod N$ 。

#### 2) 逆量子傅立葉變換計算

由前一個小節可知，模數計算後的值每隔週期  $r$  會出現 1 次，所以每個模數計算後的值會呈現  $\frac{1}{r}$  的頻率。而在模數計算主要取得模數計算後的值，但 Shor 演算法的主要目標在於取得週期  $r$ ，所以需搭配逆量子傅立葉變換計算，可以在量子計算的特性下，用多項式計算時間取得每個計算結果的頻率和週期。

#### 3) 取得週期 $r$ 和分解因數

通過逆量子傅立葉變換計算後進行量測，可以估計模數計算後的值的頻率和週期，從結果中可以取得週期  $r$ 。並且，通過公式(1)和公式(2)可以取得  $p$  值和  $q$  值。其中， $g$  函數表示取最大公因數。需要注意的是，這個步驟在取得週期  $r$  和分解因數的計算上是在經典電腦(classical computer)上執行，而不是量子電腦上執行。

$$p = g(a^r - 1, N). \quad (1)$$

$$q = g(a^r + 1, N). \quad (2)$$

### B. 最先進的方法的模數計算量子電路

在實現 Shor 演算法上，IBM 提供了一個實作範例[5]，並且後續有許多文獻[6]-[10]都跟隨著 IBM 的範例來做嘗試。其中，在 IBM 提供的實作範例中，主要採用循環和 swap 的方式來迭代交換量子位元間的量子態作為模數計算的結果(即 Algorithm 1 的 Line 02~ Line 04)，如表 I 所示。迭代完成後，最後再轉換為控制邏輯閘(即 Algorithm 1 的 Line 05)，產生量子糾纏態。其中，當第 1 個註冊器的第  $b$  個量子位元作為控制位元時，根據其量子態  $|q_b\rangle$  來執行 Algorithm 1 和產生第 2 個註冊器的量子位元量子態，則其迭代次數達  $2^b k$ 。

TABLE I. 最先進的方法的模數計算量子電路[5]-[10]

Algorithm 1. 最先進的方法的模數計算量子電路	
01:	$Q = \text{QuantumCircuit}(k)$
02:	for $\_iteration$ in range( $2^b$ ):
03:	for $i$ in range( $k - 1$ ):
04:	$Q.\text{swap}(k - i - 2, k - i - 1)$
05:	$Q = Q.\text{control}()$
06:	return $Q$

### C. 常見案例一分解整數 15

目前已經有不少研究實作 Shor 演算法，並且多為採用分解整數 15 為例(即  $N = 15$ )來說明。其中，文獻[11]採用  $a = 2$ 、文獻[12]採用  $a = 7$ 、文獻[13]採用  $a = 13$ ，都表示可以順利分解整數 15 為 3 和 5(即  $p = 3$ 、 $q = 5$ )。有鑑於此，本節採用  $a = 2$  為例來說明分解整數 15，其 Shor 演算法的量子電路如圖 2 所示。

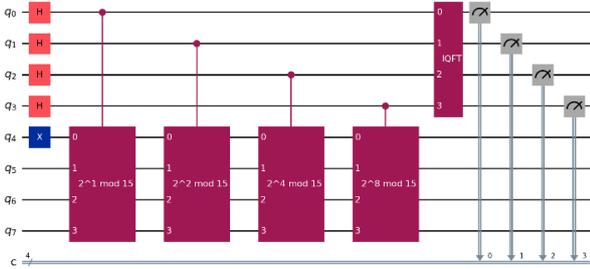


Fig. 2. 分解整數 15 的 Shor 演算法

#### 1) 模數計算

根據第 II.A 節描述的模數計算方法，設定  $k = 4$ ，首先將為每個量子位元操作 Hadamard Gate，讓其產生均勻疊加態，也就是可以同時展開  $0 \sim 2^k - 1$  的排列組合。在量子計算的特性下，可以操作 4 次 Algorithm 1 的量子電路來取得表 II 的模數計算結果，並且從可以觀察到模數計算結果只會 4 種可能(即  $|1\rangle$ 、 $|2\rangle$ 、 $|4\rangle$ 、 $|8\rangle$ )。這個結果也同時反應出週期  $r = 4$ 。

#### 2) 逆量子傅立葉變換計算

逆量子傅立葉變換計算主要用來計算每一個模數計算後的結果出現的頻率和週期。以表 II 結果，可以得到下面的結果：

- $|1\rangle(|0\rangle + |4\rangle + |8\rangle + |12\rangle)$
- $|2\rangle(|1\rangle + |5\rangle + |9\rangle + |13\rangle)$
- $|4\rangle(|2\rangle + |6\rangle + |10\rangle + |14\rangle)$
- $|8\rangle(|3\rangle + |7\rangle + |11\rangle + |15\rangle)$

由結果可知，每個模數計算後的結果出現的頻率為  $\frac{4}{16} = \frac{1}{4}$ ，並且週期為 4。圖 3 為 Shor 演算法的量子電路執行後的量測結果。

TABLE II. 分解整數 15 的模數計算結果

$i$	$f(i)$	量子態 $ q_{2k-1} \dots q_k\rangle  q_{k-1} \dots q_0\rangle =  f(i)\rangle  i\rangle$
0	1	$ 1\rangle 0\rangle$
1	2	$ 2\rangle 1\rangle$
2	4	$ 4\rangle 2\rangle$
3	8	$ 8\rangle 3\rangle$
4	1	$ 1\rangle 4\rangle$
5	2	$ 2\rangle 5\rangle$
6	4	$ 4\rangle 6\rangle$
7	8	$ 8\rangle 7\rangle$
8	1	$ 1\rangle 8\rangle$
9	2	$ 2\rangle 9\rangle$
10	4	$ 4\rangle 10\rangle$
11	8	$ 8\rangle 11\rangle$
12	1	$ 1\rangle 12\rangle$
13	2	$ 2\rangle 13\rangle$
14	4	$ 4\rangle 14\rangle$
15	8	$ 8\rangle 15\rangle$

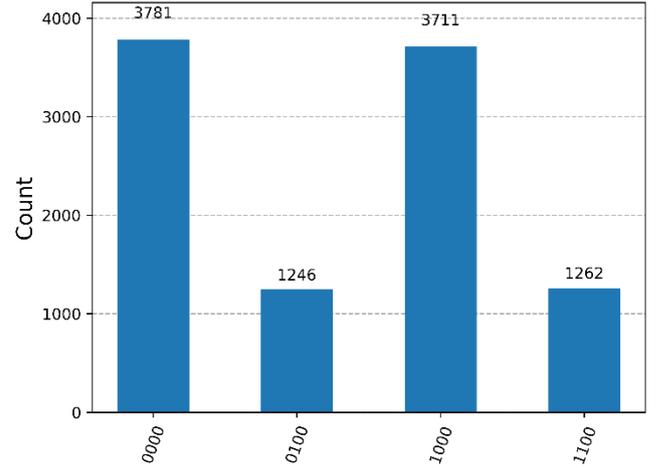


Fig. 3. 分解整數 15 的量子電路執行後的量測結果

#### 3) 取得週期 $r$ 和分解因數

從逆量子傅立葉變換計算後進行量測，可以得到 4 個結果，其代表的意涵分別如下：

- $|0000\rangle \rightarrow 0 \rightarrow \frac{0}{16} \rightarrow 16$
- $|0100\rangle \rightarrow 4 \rightarrow \frac{4}{16} = \frac{1}{4} \rightarrow 4$
- $|1000\rangle \rightarrow 8 \rightarrow \frac{8}{16} = \frac{1}{2} \rightarrow 2$
- $|1100\rangle \rightarrow 12 \rightarrow \frac{12}{16} = \frac{3}{4} \rightarrow 4$

由結果可以觀察到  $r$  有較高的可能性是 4，之後再採用公式(1)和公式(2)可以取得  $p = g(3,15) = 3$  和  $q = g(5,15) = 5$ 。

### III. 本研究提出方法

本研究主要提出具有更高效率的模數計算量子電路，並且相較於最先進的方法的模數計算量子電路，可以用更少的量子位元和量子邏輯閘。然而，需要注意的是，本研究所提方法僅適用於 $p = 3$ ，並且 $r$ 值為2的次方值。

#### A. 本研究提出的模數計算量子電路

本研究提出的模數計算量子電路是採用經典電腦和量子電腦混合的作法，先在經典電腦計算 $m = 2^b \pmod{N}$  (即 **Algorithm 2** 的 **Line 01**)，之後再根據 $m$ 值來產生對應的模數值的量子電路(即 **Algorithm 2** 的 **Line 04~ Line 08**)，如表 III 所示。這個方法相較於最先進的方法的模數計算量子電路，可以少掉一層 $2^b$ 循環，將可以大幅度減少計算量。

TABLE III. 本研究提出的模數計算量子電路

Algorithm 2. 本研究提出的模數計算量子電路	
01:	$m = 2^b \pmod{N}$
02:	$Q = \text{QuantumCircuit}(k)$
03:	$isTwoPower = \text{false}$
04:	for $i$ in range( $k - 1$ ):
05:	if $m == 2^i$ :
06:	$Q.swap(0, i)$
07:	$isTwoPower = \text{true}$
08:	break
09:	if $isTwoPower == \text{true}$ :
10:	$Q = Q.control()$
11:	return $Q$
12:	else:
13:	return <b>Algorithm 1</b>

#### B. 以分解整數 771 為例

在分解整數 771 時，如果採用最先進的方法的模數計算量子電路來計算，則 $k = 16$ 。因此，由於最先進的方法的模數計算量子電路需要用到 32 個量子位元，但如果以 IBM Qiskit 實作時會需要執行 `transpile` 函數來建立量子電路，但 `transpile` 函數僅支援最多 31 個量子位元，所以將導致無法計算。

本研究提出的模數計算量子電路可以在 $k = 12$ 情況下分解整數 771，結合本研究提出的模數計算量子電路的 Shor 演算法量子電路如圖 4 所示。

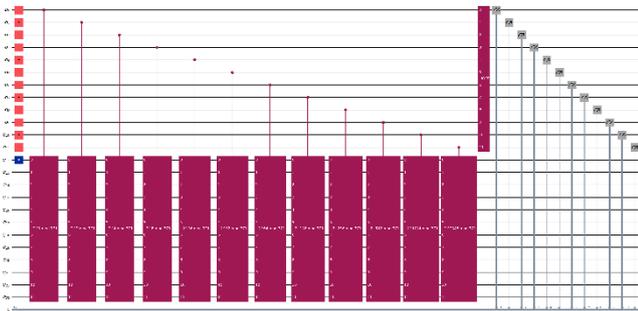


Fig. 4. 結合本研究提出的模數計算量子電路的 Shor 演算法

##### 1) 模數計算

根據第 III.A 節描述的模數計算方法，設定 $k = 12$ ，首先將為每個量子位元操作 Hadamard Gate，讓其產生均勻疊加態，也就是可以同時展開 $0 \sim 2^k - 1$ 的排列組合。在量子計算的特性下，可以操作 12 次 **Algorithm 2** 的量子電路來取得表 IV 的模數計算結果，並且從可以觀察到

模數計算結果只會有 16 種可能(即 $|1\rangle$ 、 $|2\rangle$ 、 $|4\rangle$ 、 $|8\rangle$ 、 $|16\rangle$ 、 $|32\rangle$ 、 $|64\rangle$ 、 $|128\rangle$ 、 $|256\rangle$ 、 $|512\rangle$ 、 $|253\rangle$ 、 $|506\rangle$ 、 $|241\rangle$ 、 $|482\rangle$ 、 $|193\rangle$ 、 $|386\rangle$ )。這個結果也同時反應出週期 $r = 16$ 。

TABLE IV. 分解整數 771 的模數計算結果

$i$	$f(i)$	量子態 $ q_{2k-1} \dots q_k\rangle q_{k-1} \dots q_0\rangle =  f(i)\rangle i\rangle$
0	1	$ 1\rangle 0\rangle$
1	2	$ 2\rangle 1\rangle$
2	4	$ 4\rangle 2\rangle$
3	8	$ 8\rangle 3\rangle$
...		
4095	8	$ 386\rangle 4095\rangle$

##### 2) 逆量子傅立葉變換計算

逆量子傅立葉變換計算主要用來計算每一個模數計算後的結果出現的頻率和週期。以表 IV 結果，可以得到下面的結果：

- $|1\rangle(|0\rangle + |16\rangle + |32\rangle + \dots + |4080\rangle)$
- $|2\rangle(|1\rangle + |17\rangle + |33\rangle + \dots + |4081\rangle)$
- $|4\rangle(|2\rangle + |18\rangle + |34\rangle + \dots + |4082\rangle)$
- $|8\rangle(|3\rangle + |19\rangle + |35\rangle + \dots + |4083\rangle)$
- ...
- $|386\rangle(|15\rangle + |31\rangle + |47\rangle + \dots + |4095\rangle)$

由結果可知，每個模數計算後的結果出現的頻率為 $\frac{256}{4096} = \frac{1}{16}$ ，並且週期為 16。圖 5 為 Shor 演算法的量子電路執行後的量測結果。

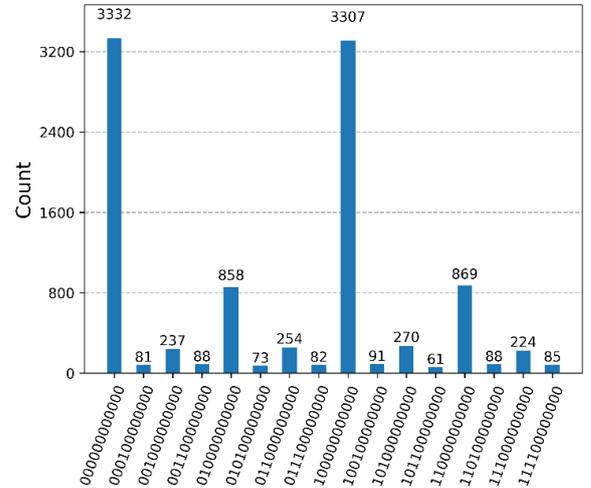


Fig. 5. 分解整數 771 的量子電路執行後的量測結果

##### 3) 取得週期 $r$ 和分解因數

從逆量子傅立葉變換計算後進行量測，可以得到 16 個結果，其代表的意涵分別如下：

- $|000000000000\rangle \rightarrow 0 = \frac{0}{4096} \rightarrow 4096$
- $|000100000000\rangle \rightarrow 256 \rightarrow \frac{256}{4096} = \frac{1}{16} \rightarrow 16$
- $|001000000000\rangle \rightarrow 512 \rightarrow \frac{512}{4096} = \frac{1}{8} \rightarrow 8$
- $|001100000000\rangle \rightarrow 768 \rightarrow \frac{768}{4096} = \frac{3}{16} \rightarrow 16$

- ...
- $|11110000000\rangle \rightarrow 3840 \rightarrow \frac{3840}{4096} = \frac{15}{16} \rightarrow 16$

由結果可以觀察到  $r$  有較高的可能性是 16，之後再採用公式(1)和公式(2)可以取得  $p = g(3,771) = 3$  和  $q = g(257,771) = 257$ 。

#### IV. 實驗結果與討論

在 IV.A 節將介紹本研究的實驗環境，並且在 IV.B 節討論實驗結果，比較本研究提出的方法和最先進的方法的結果。

##### A. 實驗環境

本研究實驗時所採用的硬體和軟體規格描述如下：CPU Intel(R) Core(TM) i7-10510U、RAM 16 GB、Python 3.11.9、NumPy 1.23.5、以及 Qiskit 1.1.1。其中，在量子計算的部分，主要採用 Qiskit 的 AER Simulator 進行模擬。為驗證本研究提出的方法，總共採用 12 個案例來進行評估與比較，每個案例在執行 Shor 演算法時皆進行 1 萬次 shot 量測，案例的詳細參數值描述於附錄 A。

##### B. 結果比較與討論

在本研究中主要採用附錄 A 描述的 12 個案例來比較本研究提出的方法和最先進的方法[5]-[10]，並且分別從使用的量子位元數、量子電路產製時間長度、以及 Shor 演算法執行時間長度來進行比較。

表 V 為 Shor 演算法使用的量子位元數及其產製時間長度，其時間單位為秒。其中，由於 IBM Qiskit 1.1.1 的 transpile 函數僅支援最多 31 個量子位元，所以如果量子電路超過該限制將無法被執行。因此，採用最先進的方法僅能實作 Case 1 和 Case 2，而其他 10 個案例將標記為“Not applicable”。從實驗結果可以發現，本研究提出的模數計算量子電路可以用更少的量子位元來分解出 4096-bit 整數。

TABLE V. SHOR 演算法使用的量子位元數及其產製時間長度

Case	The Number of Qubits		Quantum Circuit Generation Time (s)	
	The Proposed Method	The SOTA Methods [5]-[10]	The Proposed Method	The SOTA Methods [5]-[10]
Case 1	8	8	0.011	0.029
Case 2	16	16	0.020	0.576
Case 3	24	32	0.048	Not applicable
Case 4	24	64	0.120	Not applicable
Case 5	24	128	0.250	Not applicable
Case 6	24	256	0.508	Not applicable
Case 7	24	512	0.873	Not applicable
Case 8	24	1024	2.048	Not applicable
Case 9	24	2048	3.970	Not applicable
Case 10	24	4096	14.696	Not applicable
Case 11	24	8192	33.467	Not applicable
Case 12	26	16384	68.334	Not applicable

除此之外，在 Case 1 採用相同量子位元數的情況，最先進的方法的產製量子電路時間長度是本研究提出方法的產製量子電路時間長度 2.7 倍；相似的，Case 2 採用相同量子位元數的情況，最先進的方法的產製量子電路時間長度是本研究提出方法的產製量子電路時間長度 28.7 倍。有鑑於此，即使 transpile 函數可支援最先進的方法的量子位元數，但最先進的方法的量子電路產製時間也將呈指數增加。

表 VI 為 Shor 演算法使用的量子位元數及其執行時間長度，其單位時間為秒。由於本研究執行 Shor 演算法將進行 1 萬次 shot 量測，所以將需要花費較多的時間在執行和量測。隨著分解的整數值越大，則需要的執行時間長度則會越大。以 Case 11 來觀察，可以發現採用本研究提出的方法可以在 1 小時內分解 4096-bit 整數。

TABLE VI. SHOR 演算法使用的量子位元數及其執行時間長度

Case	The Number of Qubits		Quantum Circuit Execution Time (s)	
	The Proposed Method	The SOTA Methods [5]-[10]	The Proposed Method	The SOTA Methods [5]-[10]
Case 1	8	8	0.050	0.112
Case 2	16	16	0.077	2.132
Case 3	24	32	0.152	Not applicable
Case 4	24	64	5.469	Not applicable
Case 5	24	128	14.628	Not applicable
Case 6	24	256	34.696	Not applicable
Case 7	24	512	78.052	Not applicable
Case 8	24	1024	171.513	Not applicable
Case 9	24	2048	360.163	Not applicable
Case 10	24	4096	733.837	Not applicable
Case 11	24	8192	2229.684	Not applicable
Case 12	26	16384	11308.347	Not applicable

#### V. 結論與未來研究

本研究提出高效率的模數計算量子電路，與最先進的方法[5]-[10]比較，證明可以提升二十倍的效率以上。除此之外，本研究提出的方法可以用更少的量子位元數來分解長度更長的整數，包含分解 4096-Bit 整數。在實驗環境中，本研究採用 IBM Qiskit 實作量子電路和量子演算法，通過實作來驗證本研究提出方法可以在 1 小時內分解 4096-Bit 整數。然而，需要注意的是，本研究所提方法僅適用於  $p = 3$ ，並且  $r$  值為 2 的次方值之案例。

由於本研究方法僅限於特定條件限制下分解 4096-Bit 整數，距離通用型模數計算量子電路仍有段距離。在未來研究中可以嘗試設計高效率的通用型模數計算量子電路，讓量子計算更能解決各種應用問題。

##### 致謝

本研究承蒙 IBM 協助免費提供 IBM Quantum Platform 和 IBM Qiskit，讓本研究得以實作 Shor 演算法進行實驗和分析，特此感謝 IBM 提供 Open Plan。

## 參考文獻

- [1] Google Quantum AI and Collaborators, "Quantum Error Correction Below the Surface Code Threshold," in *Nature*, 2024, doi: 10.1038/s41586-024-08449-y.
- [2] L. K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," in *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing*, Philadelphia Pennsylvania, USA, 1996, pp. 212-219, doi: 10.1145/237814.237866.
- [3] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," in *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484--1509, 1997, doi: 10.1137/S0097539795293172.
- [4] C. Wang, J. Yu, Z. Pei, Q. Wang and C. Hong, "A First Successful Factorization of RSA-2048 Integer by D-Wave Quantum Computer," in *Tsinghua Science and Technology*, vol. 30, no. 3, pp. 1270-1282, June 2025, doi: 10.26599/TST.2024.9010028.
- [5] Frank Harkins, "Shor's Algorithm," in *GitHub*, Nov. 2023, URL: <https://github.com/Qiskit/textbook/blob/main/notebooks/ch-algorithms/shor.ipynb>.
- [6] A. N. Korotkov, "Shor's Algorithm," in *GitHub*, Jan. 2025, URL: [https://github.com/CS203UCR/ee214\\_25wi/blob/main/shor.ipynb](https://github.com/CS203UCR/ee214_25wi/blob/main/shor.ipynb).
- [7] H. T. Larasati and H. Kim, "Simulation of Modular Exponentiation Circuit for Shor's Algorithm in Qiskit," *Proceedings of 2020 IEEE 14th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, Bandung, Indonesia, 2020, pp. 1-7, doi: 10.1109/TSSA51342.2020.9310794.
- [8] G. R. Mounica, G. Manimaran, L. B. Jerome and P. Bhattacharjee, "Implementation of 5-Qubit approach-based Shor's Algorithm in IBM Qiskit," *Proceedings of 2021 IEEE Pune Section International Conference (PuneCon)*, Pune, India, 2021, pp. 1-6, doi: 10.1109/PuneCon52575.2021.9686492.
- [9] C. DeCusatis and E. Mcgettrick, "Near term implementation of Shor's Algorithm using Qiskit," *Proceedings of 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, NV, USA, 2021, pp. 1564-1568, doi: 10.1109/CCWC51732.2021.9376169.
- [10] D. Sun, N. Zhang and F. Franchetti, "Optimization and Performance Analysis of Shor's Algorithm in Qiskit," *Proceedings of 2023 IEEE High Performance Extreme Computing Conference (HPEC)*, Boston, MA, USA, 2023, pp. 1-7, doi: 10.1109/HPEC58863.2023.10363522.
- [11] V. Bhatia and K. R. Ramkumar, "An Efficient Quantum Computing technique for cracking RSA using Shor's Algorithm," 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India, 2020, pp. 89-94, doi: 10.1109/ICCCA49541.2020.9250806.
- [12] K. K. Soni and A. Rasool, "Cryptographic Attack Possibilities over RSA Algorithm through Classical and Quantum Computation," *Proceedings of 2018 IEEE International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India, 2018, pp. 11-15, doi: 10.1109/ICSSIT.2018.8748675.
- [13] S. Mishra, O. Agarwal and S. K. Patel, "Quantum Decryption using Shor's Algorithm," *Proceedings of 2024 IEEE Pune Section International Conference (PuneCon)*, Pune, India, 2024, pp. 1-6, doi: 10.1109/PuneCon63413.2024.10895777.

## APPENDIX A

為驗證本研究提出的方法，設計 12 個案例的  $p$  值和  $q$  值，其中  $N = pq$ 。Case 1 為常見的案例，分解 15，得到其質因數分別有 3 和 5。另外 11 個 Case 為長度更長的整數，並且被本研究方法被分解出來。

### 1) Case 1 for Factoring a 4-Bit Integer

$p: 3$

$q: 5$

$N: 15$

### 2) Case 2 for Factoring a 8-Bit Integer

$p: 3$

$q: 17$

$N: 51$

### 3) Case 3 for Factoring a 16-Bit Integer

$p: 3$

$q: 257$

$N: 771$

### 4) Case 4 for Factoring a 32-Bit Integer

$p: 3$

$q: 65537$

$N: 196611$

### 5) Case 5 for Factoring a 64-Bit Integer

$p: 3$

$q: 4294967297$

$N: 12884901891$

### 6) Case 6 for Factoring a 128-Bit Integer

$p: 3$

$q: 18446744073709551617$

$N: 55340232221128654851$

### 7) Case 7 for Factoring a 256-Bit Integer

$p: 3$

$q: 340282366920938463463374607431768211457$

$N: 1020847100762815390390123822295304634371$

### 8) Case 8 for Factoring a 512-Bit Integer

$p: 3$

$q: 11579208923731619542357098500868790785326998$   
 $4665640564039457584007913129639937$

$N: 3473762677119485862707129550260637235598099$   
 $53996921692118372752023739388919811$

### 9) Case 9 for Factoring a 1024-Bit Integer

$p: 3$

$q: 1340780792994259709957402499820584612747936$   
 $5820592393377723561443721764030073546976801$   
 $8742981669034276900318581864860508537538828$   
 $11946569946433649006084097$

$N: 4022342378982779129872207499461753838243809$   
 $7461777180133170684331165292090220640930405$   
 $6228945007102830700955745594581525612616484$   
 $35839709839300947018252291$

### 10) Case 10 for Factoring a 2048-Bit Integer

$p: 3$

$q: 1797693134862315907729305190789024733617976$   
 $9789423065727343008115773267580550096313270$   
 $8477322407536021120113879871393357658789768$   
 $814416622492847430639474124377678934248654$   
 $8527630221960124609411945308295208500576883$   
 $8150682342462881473913110540827237163350510$

6845862982399472459384797163048353563296242  
24137217

N: 5393079404586947723187915572367074200853930  
9368269197182029024347319802741650288939812  
5431967222608063360341639614180072976369306  
4432498674785422919184223731333036802745964  
5582890665880373828235835924885625501730651  
4452047027388644421739331622481711490051532  
0537588947198417378154391489145060689888726  
72411651

11) Case 11 for Factoring a 4096-Bit Integer

$p: 3$

$q:$  3231700607131100730071487668866995196044410  
2669715484032130345427524655138867890893197  
2014115229134636887179609218980194941195591  
5049092109508815238644828312063087736730099  
6091750197750389652106796057638384067568276  
7922186426197561618380943384761704705816458  
5203630504288757589154106580860755239912393  
0385521914333389668342420684974786564569494  
8561760353263220580778056593310261927084603  
1415025859286417711672594360371846185735759  
8351152301645904403697613233287231227125684  
7108202097251571017269313234696785425806566  
9793504599726835299863821552516638943733554  
3602135433229604645318478604952148193555853  
611059596230657

N: 9695101821393302190214463006600985588133230  
8009146452096391036282573965416603672679591  
6042345687403910661538827656940584823586774  
5147276328526445715934484936189263210190298  
8275250593251168956320388172915152202704830  
3766559278592684855142830154285114117449375  
5610891512866272767462319742582265719737179  
1156565743000169005027262054924359693708484  
5685281059789661742334169779930785781253809  
4245077577859253135017783081115538557207279  
5053456904937713211092839699861693681377054  
1324606291754713051807939704090356277419700  
9380513799180505899591464657549916831200663  
0806406299688813935955435814856444580667560  
833178788691971

12) Case 12 for Factoring a 8192-Bit Integer

$p: 3$

$q:$  1044388881413152506691752710716624382579964  
2490473837803842334832839539079715574568488  
2681193499755834089010671443926283798757343  
8185793607263236087851365277945956976543709  
9983403615901343837183144280700118559462263  
7631883939771274567233468434458661749680790

8705803704071284048740118609114467977783598  
0290066869389768817877859469056301902609405  
9957945343282346930302669644305902501597239  
9867714215541693835559885291486318237914434  
4967340878118726394964751001890413490084170  
6167509366833385055103297208826955076998361  
6369411933015213796825837188091833656751221  
3184928463681255502259983004123447848625956  
7449219461702380650591324561082573183538008  
7608622102834270197698202313169017678006675  
1954850799216364193702853751247840149071591  
3545998279051339961155179427110683113409058  
4272884279791554849782954323534517065223269  
0613949059876930021229633956877828789484406  
1600741294567491982305057164237715481632138  
0631045902916136926708342856440730447899971  
9017814657634732238502672530598997959960907  
9946920177462481771844986745565925017832907  
0473119433165550807568221846571746373296884  
9128195203174570024409266169108741483850784  
1192980452298185733897764810312608590300130  
2413467189726673216491511131602920781738033  
436090243804708340403154190337

N: 3133166644239457520075258132149873147739892  
7471421513411527004498518617239146723705464  
8043580499267502267032014331778851396272031  
4557380821789708263554095833837870929631129  
9950210847704031511549432842100355678386791  
2895651819313823701700405303375985249042372  
61174111122138521462203558273434039333502794  
0870200608169306453633578407168905707828217  
9873836029847040790908008932917707504791719  
9603142646625081506679655874458954713743303  
4902022634356179184894253005671240470252511  
8502528100500155165309891626480865230995084  
9108235799045641390477511564275500970253663  
9554785391043766506779949012370343545877870  
2347658385107141951773973683247719550614026  
2825866308502810593094606939507053034020025  
5864552397649092581108561253743520447214774  
0637994837154019883465538281332049340227175  
2818652839374664549348862970603551195669807  
1841847179630790063688901870633486368453218  
4802223883702475946915171492713146444896414  
1893137708748410780125028569322191343699915  
7053443972904196715508017591796993879882723  
9840760532387445315534960236697775053498721  
1419358299496652422704665539715239119890654  
7384585609523710073227798507326224451552352  
3578941356894557201693294430937825770900390  
7240401569180019649474533394808762345214100  
308270731414125021209462571011