Shor's Algorithm

Ioannis Savvaidis

1 General Analysis of Shor's Algorithm

Shor's algorithm provides a methodology to factor a prime integer in polynomial time. Factoring in classical computers is a Non-Polynomial Problem but has not yet been classified as NP-complete. Shor's algorithm consists of classical and quantum computation steps composed of number theory and quantum phase estimation techniques. Its best-known application is breaking the RSA Cryptosystem, but with a qubit number barrier. As it has been proved, the algorithm needs at least twice the number of qubits required to encode the random big integer prime of the system. This means that for an RSA2048-produced integer prime with 2048 bits, the algorithm needs at least 4096 qubits. The lower computational cost of breaking the RSA system is to find the period of a modular exponent function and apply it to *gcd* functions to construct the two factors of the prime integer. These two factors lead to retrieving the private key.

1.0.1 RSA Components in a nutshell

- p, q: Two large prime numbers.
- $N = p \times q$: The modulus used in both the public and private keys.
- d: Private exponent, random integer prime to $(p-1) \times (q-1)$
- e: Public exponent given by:

$$d \times e \equiv 1 \pmod{(p-1) \times (q-1)}$$

- Public key: (e, N)
- Private key: (d, N)
- Encryption: Given message M, $C = M^e \mod N$
- Decryption: Given ciphertext C,$M = C^d \mod N$

Shor's Algorithm has only one quantum computational part, which is to find the period using the Quantum Phase Estimation by applying the modular exponent function as an Oracle.

1.0.2 Modular exponent function

$$f(x) = a^x mod N$$

a a random chosen number where $2 < a < N$ and $gcd(a, N) = 1$
$x \in \mathbb{N}_0 = \{0, 1, 2, 3, \dots\}$

1.0.3 Period finding

For each x (in decimal), the function f is being evaluated until $f(x) = 1$, then the non-zero iteration x, is the period. The period of f, denoted with r, and it is the first non-zero integer that:

$$f(r) = a^r mod N = 1 \tag{1}$$

1.0.4 Quantum Phase Estimation(QPE)

QPE (Fig. 1), extends the Phase-kickback phenomenon and provides an algorithm to calculate a Unitary Operator in high accuracy. It consists of two registers, usually with different names, but we'll keep *control* and *target* for simplicity. If the Control Register has m-qubits and the target register has n-qubits, the Unitary Operator (U) is a controlled-Unitary operator (CU) controlled by control qubits and acting on all target register qubits. The appliance needs a convention on how the quantum register represents the bits. Qiskit ecosystem has an inverted convention, which we will use here for simplicity. Thus, the most significant bit (MSB) is represented by the least significant register number e.g. for a 5-bit number, x_4 is represented by qubit q_0, and x_0 by qubit q_4.

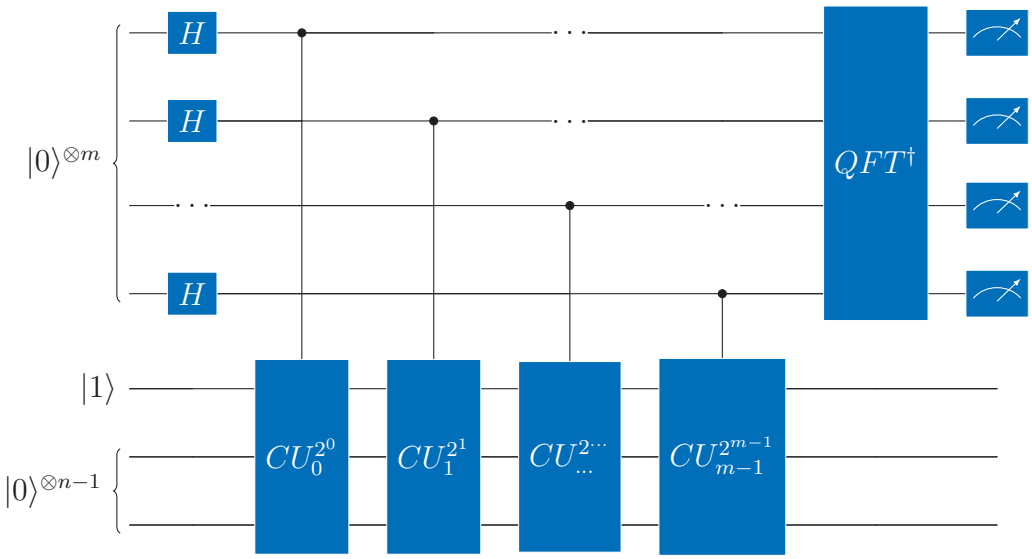


Figure 1: QPE Circuit in Qiskit Convention

If $x_{[10]}$ is a x in decimal state and $x_{[2]}$ the binary, represented by control register with m-bits, then:

$$x_{[10]} = 2^{m-1}x_{m-1} + 2^{m-2}x_{m-2}... + 2^0 x_0$$

The binary expansion of m, relative to the control register state, is:

$$x_{[2]} = x_{m-1}x_{m-2}...x_0$$

The unitary operator is creating an eigenvector of the applied state vector:

$$U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$$

The phase θ controls the eigenvalue and has a value of 0 to 1 range.
The unitary operator U application is proportional to the number of control register qubits, and the target is all target register qubits.
for each $j \in \{0..m-1\}$:

$$CU_j^{2^j}$$

For 1 control qubit $m = 1$, the control version of U:

$$CU_0^{2^0}|+\rangle|\psi\rangle = \frac{1}{\sqrt{2^m}}CU_0^{2^0}\left(|0\rangle|\psi\rangle + |1\rangle|\psi\rangle\right) = \frac{1}{\sqrt{2}}\left(CU_0^{2^0}|0\rangle|\psi\rangle + CU_0^{2^0}|1\rangle|\psi\rangle\right)$$
$$= \frac{1}{\sqrt{2}}\left(|0\rangle|\psi\rangle + |1\rangle U_0^{2^0}|\psi\rangle\right) = \frac{1}{\sqrt{2}}\left(|0\rangle|\psi\rangle + |1\rangle e^{2\pi i\theta 2^0}|\psi\rangle\right)$$
$$= \frac{1}{\sqrt{2}}\left[(|0\rangle + e^{2\pi i\theta 2^0}|1\rangle)|\psi\rangle\right]$$

This is the phase kickback since the global phase of the target qubit, controlled by the U operator, is transferred back to the relative phase of the control qubit.

Generalizing the equation to m-bits summation, the control register's state after CU applications will be:

$$|\psi_1\rangle = H^{\otimes m}|0\rangle \otimes |1\rangle|0\rangle^{\otimes n-1}$$

$$|\psi_2\rangle = \frac{1}{\sqrt{2^m}}\sum_{k\in\{0,2^m-1\}} e^{2\pi i\theta k}|k\rangle$$

Applying the inversed QFT ($QFT\dagger$), will convert the phase into control register's state, and we can measure it:

$$|\psi_3\rangle = QFT^\dagger|\psi_2\rangle = |2^m\theta\rangle \tag{2}$$

1.1 Simulate Shor's Algorithm in Qiskit

Shor circuit uses the QPE circuit in Fig. 1, for period finding upon control register measurement. After the measurement, post-processing is performed to factor the given prime integer using the founded period. To describe the algorithm, we will use $N = 15$, as the prime integer we need to factor. The most critical part is to create the Oracle Circuit, which I found confusing when I tried to apply the QPE-separated CU oracles. I made a tool to perform a few calculations before oracle design, in order to select the number a for the modular exponent function. Since this number dictates the period, it has direct impact on circuit complexity. I found it easier to work with $a = 4$ due to function outputs, which made it easier to construct the circuit. This tool is nothing special, and many qiskit examples having similar tools.

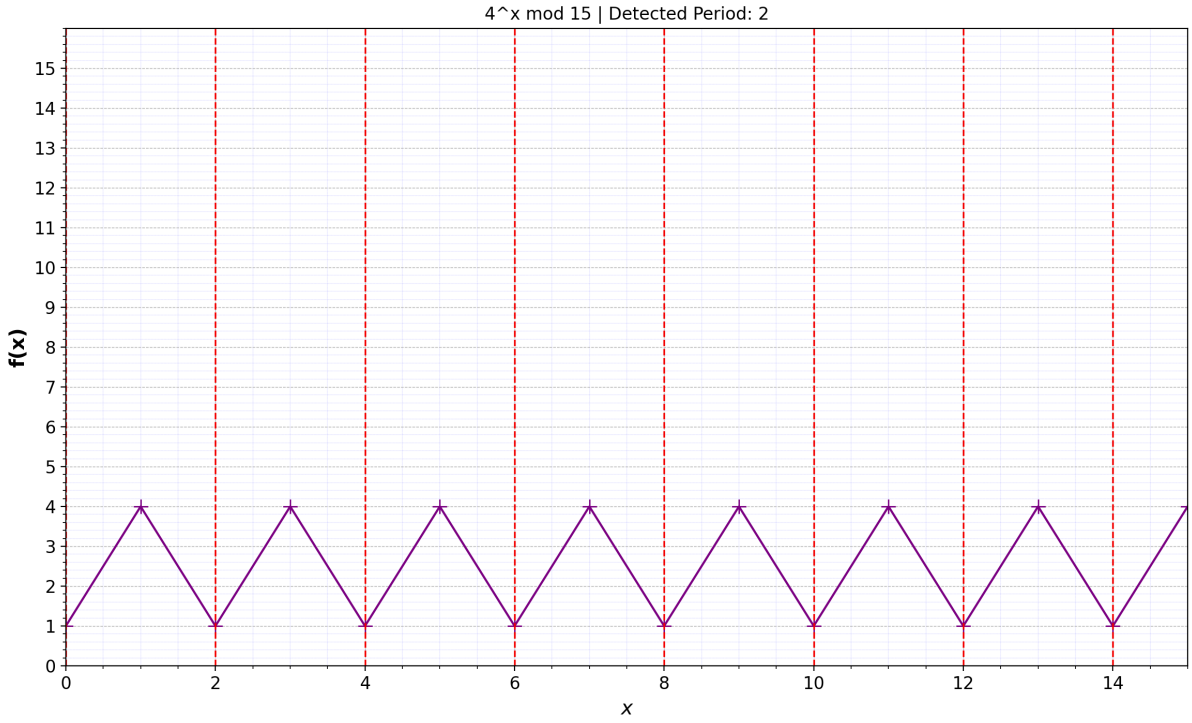


Figure 2

```
➜  modular_exponent python modular_exponent.py
Bits of P Prime : 256
Bits of Precision: 16
Control Register Qubits: 8
Target Register Qubits: 4
f(x)=4^x mod 15
        x                f(x)  |   (Binary)x              f(x)
-------------------------------------------------------------------
        0                   1  |       0000               001
        1                   4  |       0001               100
        2                   1  |       0010               001
        3                   4  |       0011               100
        4                   1  |       0100               001
        5                   4  |       0101               100
        6                   1  |       0110               001
        7                   4  |       0111               100
        8                   1  |       1000               001
        9                   4  |       1001               100
       10                   1  |       1010               001
       11                   4  |       1011               100
       12                   1  |       1100               001
       13                   4  |       1101               100
       14                   1  |       1110               001
       15                   4  |       1111               100
```

Figure 3

Along with the period, we calculate the needed qubits for Control register (m) and for target (n) with:

$$n = \log_2 N, m = 2n$$

To construct the multiple Controlled Oracles, we need to construct the first instance with $j = 0$, and detect the output. In Shor's Algorith, the U operator is :

$$U_{a,N}|y\rangle = |ay \mod N\rangle$$

In our case, $N = 15$ and we chose $a = 4$:

$$U_{4,15}|y\rangle = |4y \mod 15\rangle$$

We will choose fewer qubits for this example. We have m=n=4, U is unitary, and QPE starts the target register in $|1\rangle$, therefore:

$$U_{4,15}^{2^0}|y\rangle = U_{4,15}|y\rangle = |4y \mod 15\rangle$$

$$U_{4,15}^{2^1}|y\rangle = U_{4,15}^{2}|y\rangle = |4^2y \mod 15\rangle$$

$$U_{4,15}^{2^2}|y\rangle = U_{4,15}^{4}|y\rangle = |4^4y \mod 15\rangle$$

$$U_{4,15}^{2^3}|y\rangle = U_{4,15}^{8}|y\rangle = |4^8y \mod 15\rangle$$

As depicted in Fig. 3, the first non-zero x with f=1, is 2, therefor r=2.

$$f(1) = 4$$

$$f(2) = 1$$

Applying the Oracles, we can observe the state changes needed to construct the circuit:

$$U_{4,15}|1\rangle = |4\rangle$$

$$U_{4,15}^2|1\rangle = U_{4,15}(U_{4,15}|1\rangle) = U_{4,15}|4\rangle = |1\rangle$$
$$U_{4,15}^4|1\rangle = U_{4,15}^2(U_{4,15}^2|1\rangle) = U_{4,15}^2|1\rangle = |1\rangle$$
$$U_{4,15}^8|1\rangle = U_{4,15}^4(U_{4,15}^4|1\rangle) = |1\rangle$$

Only the first Oracle application is transforming the following states:

$$U_{4,15} : |1\rangle \rightarrow |4\rangle$$
$$U_{4,15} : |4\rangle \rightarrow |1\rangle$$

The other Oracle appliances do not change state. Hence, the decision of $a = 4$, consulting the Fig. 4

Modular Exponentiation Operator U^1:

|y⟩	U^p|y⟩	(Binary)x	f(x)
0	0	0000	000
1	4	0001	100
2	8	0010	1000
3	12	0011	1100
4	1	0100	001
5	5	0101	101
6	9	0110	1001
7	13	0111	1101
8	2	1000	010
9	6	1001	110
10	10	1010	1010
11	14	1011	1110
12	3	1100	011
13	7	1101	111
14	11	1110	1011
15	0	1111	000

Modular Exponentiation Operator U^2:

|y⟩	U^p|y⟩	(Binary)x	f(x)
0	0	0000	000
1	1	0001	001
2	2	0010	010
3	3	0011	011
4	4	0100	100
5	5	0101	101
6	6	0110	110
7	7	0111	111
8	8	1000	1000
9	9	1001	1001
10	10	1010	1010
11	11	1011	1011
12	12	1100	1100
13	13	1101	1101
14	14	1110	1110
15	0	1111	000

Modular Exponentiation Operator U^4:

|y⟩	U^p|y⟩	(Binary)x	f(x)
0	0	0000	000

Figure 4

The circuit was simulated with Qiskit and executed using the ideal simulator. Fig. 5 depicts the circuit, and Fig. 6 shows the 4 Oracle appliances. The SWAP of target_0 and target_2 qubits of the target register covers the state transitions, and for all other applications, the multiplied execution of SWAP reverts the state to the beginning.

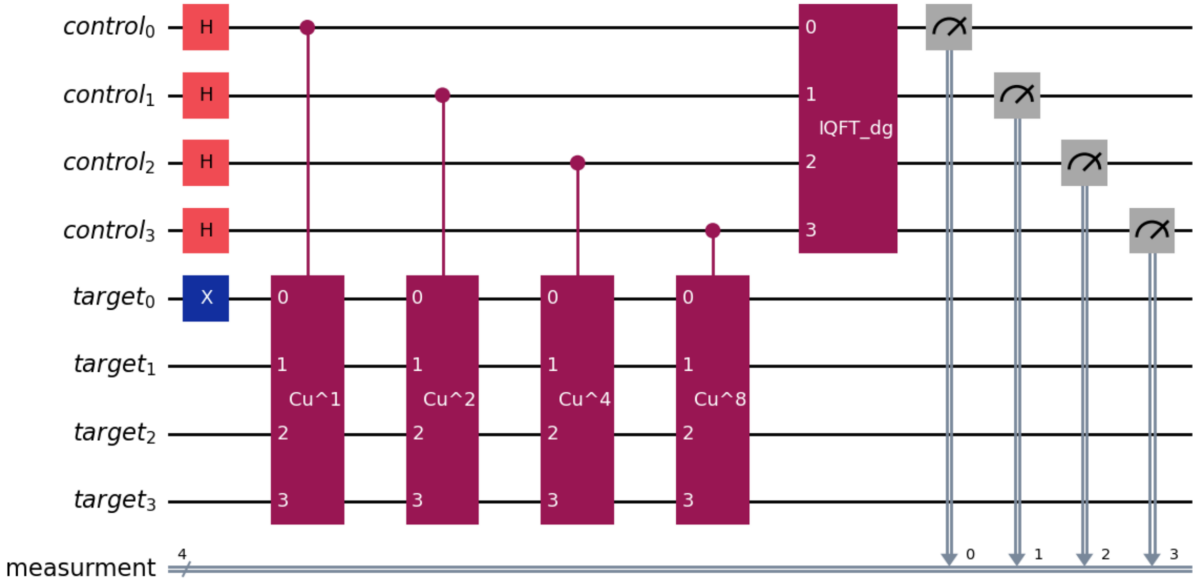
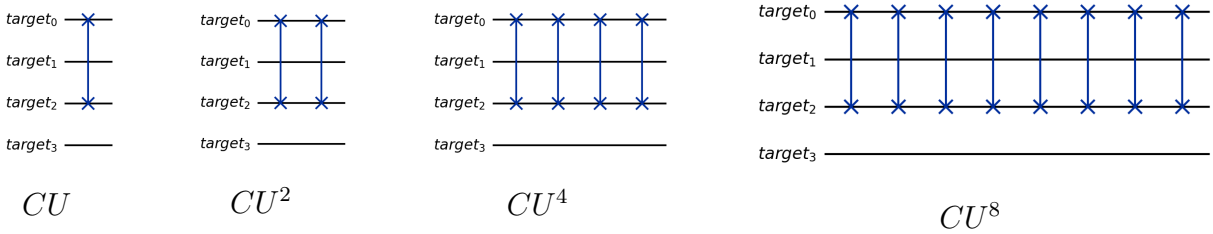


Figure 5: Shor Circuit in Qiskit

Figure 6: Oracles



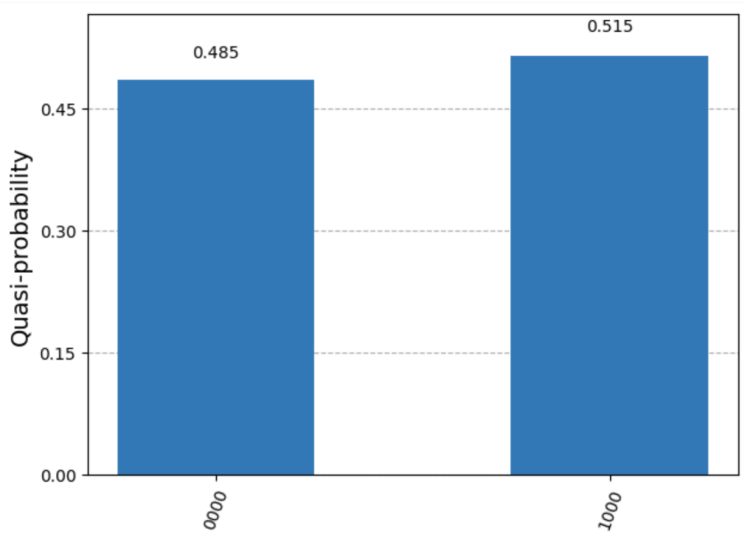
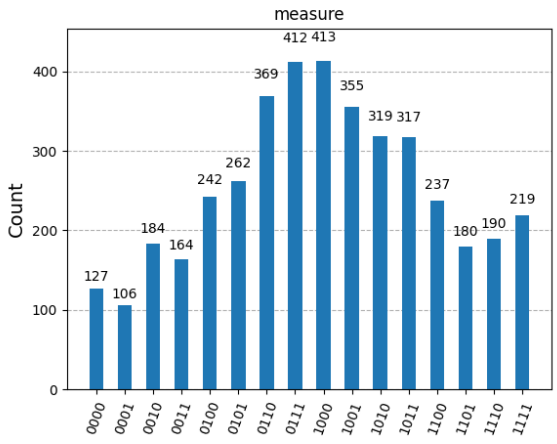


Figure 7: Ideal Simulator Results

Executing on Real Quantum Computer

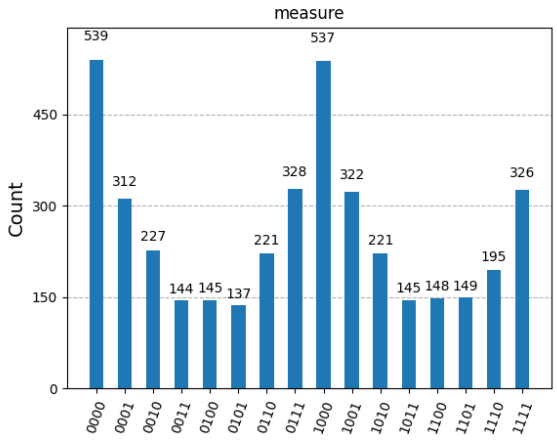
The code was executed on ibm_brisbane due to immediate availability, but the results were wrong Fig. 8 . FakeSherbrooke was more reliable during experiments Fig. 9, and I executed my circuit on a ibm_sherbrooke backend, but with noticeable noise. During all Assignments, I've noticed that Brisbane is not responding reliably in more complex circuits, while Sherbrooke provides less noisy results. Exporting Operations metrics, I can observe that Brisbane has fewer operations after transpilation, which may explain the more noisy results.

Figure 8: Results/Counts



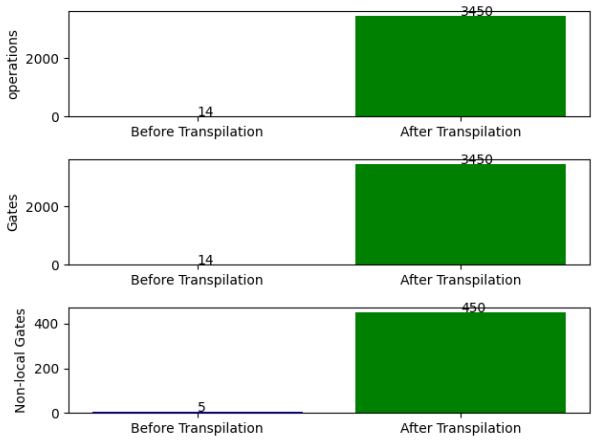
ibm_brisbane

Figure 9: Results/Counts

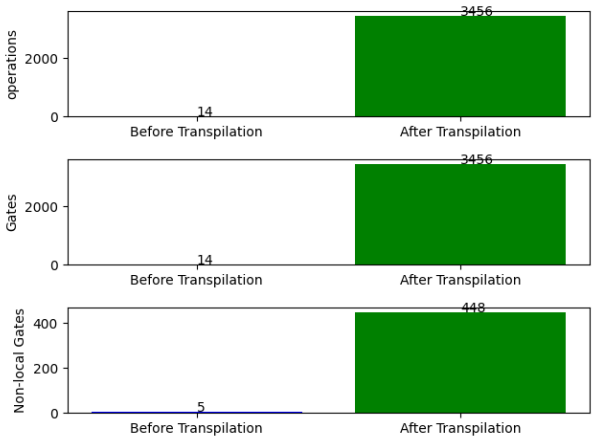


FakeSherbrooke

Figure 10: Operations



ibm_brisbane



ibm_sherbrooke

Post-processing

Post-processing of the Shor's Circuit results Fig. 7, for a=4 and N=15, is the final step of Shor's Algorithm. From the ideal simulator, the measurement results were the states $|0000\rangle$ and $|1000\rangle$, with almost equal probability.
From equation (2), we know that the final state of the circuit will be:

$$|2^m \phi\rangle$$

Since we have 2 peaks, we can denote the final measurement as l, with:

$$l_0 = [0000]_2 = [0]_{10}$$

$$l_1 = [1000]_2 = [1 * 2^4 + 0 * 2^3 + 0 * 2^1 + 0 * 2^0] = [8]_{10}$$

and:

$$l_i = 2^m \phi_i, i \in \{0, 1\}$$

Therefore:

$$\phi_0 = 0$$

$$\phi_1 = 8/2^4 = 1/2$$

The period is encoded in the phase ϕ, as:

$$\phi = s/r$$

Using continuous fractions to detect r

$$\phi_0 = \frac{s_0}{r_0} \rightarrow \frac{0}{0} = \frac{s_0}{r_0}$$

$$\phi_1 = \frac{s_1}{r_1} \rightarrow \frac{1}{2} = \frac{s_1}{r_1}$$

Therefore, we have two results:

$$r_0 = 0, r_1 = 2$$

Checking equation (1), to find the period:
The first solution of r_0 is not acceptable; thus we'll evaluate only r_1

$$f(r_1) = a^{r_1} mod N = 4^2 mod 15 = 1$$

Since f(2) = 1, we know that $r = 2$.
Condition checking:

$$4^{2/2} mod 15 = 4 \neq \pm 1$$

Factoring N, into two primes:

$$p = gcd(r - 1, N) = 3$$

$$q = gcd(r + 1, N) = 5$$