

A quantum algorithm for computing multiplicative order of integers modulo a prime

Maheswara Rao Valluri

School of Mathematical and Computing Sciences

Fiji National University

P. O. Box 7222

Derrick Campus

Suva

Fiji

Abstract

One of the challenging problems in number theory is to compute efficiently multiplicative order of integers modulo a prime. This problem was listed under open problems in the number theoretic complexity II proposed by Leonard M. Adleman and Kevin S. McCurley in 1994. This paper offers a polynomial-time quantum algorithm for computing the order $k = \min \{x \mid x \in \mathbb{N}, a^x \equiv 1 \pmod{p}\}$, if $p \in \text{Primes}$ and $\gcd(a, p) = 1$, where $a, p \in \mathbb{N}$, and it requires $\mathcal{O}((\log_2 p)^{2+O(1)})$ quantum operations.

Subject Classification: 2020 - 11F06, 68Q12, 68Q30.

Keywords: Group order, Order finding, Qubits, Quantum computation.

1. Introduction

One of the most interesting and challenging problems in number theory is to compute multiplicative order of integers modulo a prime. In 1994 [1], Leonard M. Adleman and Kevin S. McCurley proposed the following problem as one of the open problems in number theory: "Let p be prime and let $a \in \mathbb{Z}_p^*$ be an integer. Then, compute the order $k = \min \{x \mid x \in \mathbb{N}, a^x \equiv 1 \pmod{p}\}$, if $p \in \text{Primes}$ and $\gcd(a, p) = 1$, where $a, p \in \mathbb{N}$ is hard". Analogy to this problem is "Compute the order $k = \min \{x \mid x \in \mathbb{N}, a^x \equiv 1 \pmod{n}\}$, with $\gcd(a, n) = 1$, where $a \in \mathbb{N}$ and $n \in \text{Composites}$ is hard" too. These are considered NP-problems. If one can efficiently compute multiplicative order of an integer modulo a composite, we can efficiently

* E-mail: maheswara.valluri@fnu.ac.fj

factorize an RSA composite and then break the RSA cryptosystem [30]. This idea works as follows: Let n be an RSA composite. If order of an integer $a \in \mathbb{Z}_n$ is an even, say k , then $a^k \equiv 1(\text{mod } n)$. Since $(a^{\frac{k}{2}} + 1)(a^{\frac{k}{2}} - 1) \equiv 1(\text{mod } n)$, $\gcd(a^{\frac{k}{2}} - 1, n)$ gives a non-trivial factor of n ; and one can then easily break the RSA cryptosystem using a non-trivial factor of n . This idea was proposed by Peter Shor in 1994 [3] offering a polynomial-time quantum algorithm for the integer factorization problem and the discrete logarithmic problem using the order finding approach. The aim of this paper is to offer a polynomial time quantum algorithm for the unsolved open problem presented in [C20 of 1].

Quantum computing is one of the modern scientific fields that can be used to investigate computational power and some properties of quantum mechanical principles. One of the significant properties of quantum computing is that its computational power is exponentially faster than a conventional computer. So, one can use a quantum computer to solve the computational hard problems such as the integer factorization problem and the discrete logarithmic problem using quantum algorithms, that a conventional computer can not solve. In the early 1980's, quantum computer was suggested by Yuri Manin [9], Richard Feynman [10,11], and Paul Benioff [12]. Later, David Deutsch defined the universal quantum Turing machine [13] in 1985. The first quantum algorithm was developed by Deutsch and Jozsa [14] and then by Simon [15]. Furthermore, the development of quantum complexity was done by Bernstein and Vazirani [16]. Peter Shor discovered an efficient quantum algorithm for factorizing a large integer and solving the discrete logarithmic problem using the quantum Fourier transformation for Simon's algorithm [3]. Furthermore, there were many algorithms [17-25] proposed based on Shor's algorithm. In this paper, we offer a polynomial time quantum algorithm for computing multiplicative order of an integer modulo a prime by revisiting Shor's algorithm.

This paper is organized as follows: In Section 2, we recall basic principles. In Section 3, we present quantum computation and its Fourier transformation. In Section 4, we present our algorithm for computing multiplicative order of an integer modulo a prime and its computational complexity. Finally, we discuss conclusion remarks in Section 5.

2. Basic Principles

In this Section, we recall some background related to number theoretic principles and theorems from [2,4]. We begin this section by recalling definition of group order.

Definition 1: If G is a multiplicative group, then its cardinality is the order of the group, and it is denoted by $O(G)$.

For example, the cardinality of $G = \mathbb{Z}_n^* = \{1, 2, \dots, n-1\}$, where $n \in \mathbb{N}$, is $O(G) = n-1$.

Definition 2: If an element $a \in G$, then the cyclic subgroup generated by a is the subgroup $\langle a \rangle = \{a^k : k \in \mathbb{Z}\}$ of the group G . The order of an element $a \in G$ is the order of cyclic subgroup $\langle a \rangle$.

We now define the multiplicative order of an element a in the cyclic group $\mathbb{Z}_n^*$.

Definition 3: If $a \in \mathbb{Z}_n^*$, where $n \in \mathbb{N}$, the multiplicative order of $a \pmod{n}$ to be $\text{ord}_n(a) = \min\{k \in \mathbb{N} : a^k \equiv 1 \pmod{n}\}$.

For example, let $\mathbb{Z}_{15} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$. Then, the order of $7 \pmod{15}$ is 4, since $7^4 \equiv 1 \pmod{15}$.

Theorem 4: (Fermat's Little Theorem) If p is prime and $\gcd(a, p) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$.

This is a special case of the following Euler's theorem.

Theorem 5: (Euler's Theorem) Let n be a positive integer. Then $a^{\varphi(n)} \equiv 1 \pmod{n}$ for any positive integer a such that $\gcd(a, n) = 1$.

If n is a prime, then $\varphi(n) = n-1$. It is clear that the Fermat's Little Theorem tells us that $\text{ord}_p(a)$ is at most $p-1$.

Remark 6: If $\text{ord}_p(a) \mid p-1$, then $a^{p-1} \equiv 1 \pmod{p}$ as well.

Theorem 7: (Lagrange Theorem) Let G be a finite group and let H be a subgroup of G . Then $O(H)$ divides $O(G)$.

There is a direct application of the Lagrange theorem to the Fermat's Little theorem.

Theorem 8: For p is prime, the order of an element $a \in \mathbb{Z}_p^*$ divides $\varphi(p)$.

Proof: By Euler's theorem, we know that $a^{\varphi(p)} \equiv 1 \pmod{p}$ for prime p . We assume that $a^k \equiv 1 \pmod{p}$, where $k > 0$. By Euclidean algorithm, there exists unique $q, r \in \mathbb{N}$ such that $k = q\varphi(p) + r$, where $0 \leq r < \varphi(p)$.

It follows that $1 \equiv a^k \equiv a^{q\varphi(p)+r} \equiv (a^{\varphi(p)})^q a^r \equiv a^r \pmod{p}$. This is contradiction unless $r = 0$. So, it follows that $\varphi(p) | k$. This concludes the theorem.

Theorem 9: (Fundamental Theorem of Orders) Let $\mathbb{Z}_p^*$ be a multiplicative group and let $a \in \mathbb{Z}_p^*$ has order k . Then $a^k \equiv 1 \pmod{p}$ if and only if $\text{ord}_p(a)$ divides k .

For example, let $5 \in \mathbb{Z}_{19}^*$. Since $5^9 \equiv 1 \pmod{19}$, we have $9 | 18$.

Theorem 10: Let p be prime and let $a \in \mathbb{Z}_p^*$. If $a^m \equiv 1 \pmod{p}$ and $a^n \equiv 1 \pmod{p}$, then $a^{\text{gcd}(m,n)} \equiv 1 \pmod{p}$.

Remark 11: Note that $\text{gcd}(a^m - 1, a^n - 1) = a^{\text{gcd}(m,n)} - 1$.

Lemma 12: Let p be an odd prime. Then, the probability that $\text{ord}_p(a) = k$ is odd for a uniformly chosen $a \in \mathbb{Z}_p^*$ is at most $\frac{1}{2}$.

Proof: Let $\mathbb{Z}_p^* = \{1, 2, \dots, p-1\}$ be a multiplicative group modulo prime p . Note that $\mathbb{Z}_p^*$ is a cyclic group. By Fermat's little theorem, we know that $a^{p-1} \equiv 1 \pmod{p}$ for every integer $a \in \mathbb{Z}_p^*$. Since $\mathbb{Z}_p^*$ is a cyclic group, there is an element a which generates all elements of $\mathbb{Z}_p^*$. Let $k = \text{ord}_p(a)$. Then, $a^k \equiv 1 \pmod{p}$, which means that p divides $a^k - 1$. Since a is chosen uniformly at random, the probability of the order k is to be odd at most $\frac{1}{2}$. This concludes the lemma.

Algorithm 1: Conventional algorithm for computing order of an integer modulo a prime

Input: Prime $p \in \mathbb{N}$ and an integer $1 < a < p$

Runtime: $\mathcal{O}\left(\left(\log_2\left(\frac{p-1}{2}\right)\right)(\log_2 p)^2\right)$ arithmetic operations

Procedure:

1. Compute $a^{\frac{p-1}{2}}$:
2. if $a^{\frac{p-1}{2}} \equiv -1 \pmod{p}$, then
3. Declare $\text{ord}_p(a)$ is $p-1$; **return** a as a witness.
4. else

5. for i from 1 to $p - 1$
6. Compute $b \leftarrow a^i \pmod{p}$
7. while $b = 1$, declare $\text{ord}_p(a)$ is k
8. end if

Output : $\text{ord}_p(a)$.

The conventional algorithm requires $\mathcal{O}\left((\log_2\left(\frac{p-1}{2}\right)(\log_2 p)^2\right)$ arithmetic operations to find multiplicative order of an integer modulo a prime in a worst-case scenario.

Quantum computation

In this Section, we recall some of the basic principles and properties of quantum computation from [5]. A quantum analogue of a conventional computer operations on qubits involves a quantum state. A quantum state is described as a basis vector in a Hilbert space. A state of a qubit in the Hilbert space is described as follows:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle, \quad (1)$$

where $\alpha, \beta \in \mathbb{C}$ are amplitudes such that $|\alpha|^2 + |\beta|^2 = 1$, and $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ are basis vectors of the Hilbert space.

A state of n qubits system is described in 2^n dimensional vector space, represented as follows:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |x_i\rangle, \quad (2)$$

where the amplitudes are complex numbers such that $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$ and each $|x_i\rangle$ is a basis vector of the Hilbert space. In quantum computing, qubits in a register may be observed by measuring the quantum system. If the measurements are made by the machine with respect to the basis, the probability of seeing the basis state $|x_i\rangle$ is $|\alpha_i|^2$. The result of such a measurement is to collapse the quantum system, and then the system function to a distinct state.

In quantum computation, quantum gates on a qubit are represented by unitary matrices. One of the most important quantum gates, the Hadamard gate H is represented by

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}. \quad (3)$$

Other such quantum gates are the Pauli matrices,

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad (4)$$

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad (5)$$

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad (6)$$

and the controlled phase gate

$$R_n = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^n}} \end{pmatrix}. \quad (7)$$

Quantum Fourier Transformation

Quantum Fourier Transformation (QFT) was invented by Don Coppersmith [7] and is a linear transformation on qubits. This is the quantum analogue of the Discrete Fourier Transformation (DFT). We note that the QFT is a generalization of the Hadamard transformation in higher dimensions. The implementation of the QFT requires only $O(n \log n)$ operations on n - qubits [8]. The QFT was used in many applications such as Shor's quantum algorithm [3], quantum hidden shift problem algorithm [26] and quantum phase estimation algorithm [27-29]. We now briefly discuss how it works. Consider an n - dimensional system with $|x\rangle$ from the computational basis $\{|0\rangle, |1\rangle, |2\rangle, \dots, |2^n - 1\rangle\}$. On this system, we define the QFT as follows:

$$QFT |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi i x k}{2^n}} |k\rangle. \quad (8)$$

The QFT allows us to transfer an integer $|x\rangle$ in the relative phase of uniform superposition consisting of all the states $|k\rangle$ in the computational basis, each with the same amplitude $\frac{1}{\sqrt{2^n}}$.

The Inverse Quantum Fourier Transformation (IQFT) is defined as follows for the above $QFT |x\rangle$:

$$IQFT |k\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} e^{-\frac{2\pi i x k}{2^n}} |x\rangle. \quad (9)$$

We note that

$$IQFT |k\rangle = QFT^{-1}(QFT |x\rangle) = |x\rangle. \quad (10)$$

Quantum algorithm

In this Section, we present a quantum algorithm that can compute multiplicative order of an integer a in $\mathbb{Z}_p^*$. In general, we need to choose an integer $a \in \mathbb{Z}_p^*$ such that $\gcd(a, p) = 1$ and then compute $a^{(p-1)/2^i} \equiv r \pmod{p}$ for $i = 1, 2, \dots, 2^{i-1}$. If $r = -1$, then the order of a modulo p is $\frac{(p-1)}{2^{i-1}}$. However, it is hard to compute order of a in $\mathbb{Z}_p^*$ for large p in this approach as one needs to factorize $(p-1)$. For instance, for prime $p = 40973$ and $a = 7$, we have $O_p(a) = 10243$, and $(p-1) = 2^2 \times 10243$. This shows that $O_p(a)$ is one of the factors of $(p-1)$. We note that $O_p(a) \mid (p-1)$ by the Lagrange's theorem. In this scenario, we can use Shor's quantum algorithm for computing order of an integer a modulo p efficiently. We first compute $a^{\frac{p-1}{2}}$. If $a^{\frac{p-1}{2}} \equiv -1 \pmod{p}$, then we declare that the order of $a \pmod{p}$ is $(p-1)$, otherwise we use Shor's algorithm to find the order k of an integer a in $\mathbb{Z}_p^*$ such that $a^k \equiv 1 \pmod{p}$. Note that we do not require to compute $\gcd(a, p)$, since p is prime.

Quantum algorithm for computing order of an integer modulo a prime

Input: Prime $p \in \mathbb{N}$, and an integer $1 < a < p$

Runtime: $\mathcal{O}((\log_2 p)^{2+O(1)})$ operations

Procedure: $a^{\frac{p-1}{2}}$:

1. Compute $a^{\frac{p-1}{2}}$:
2. **if** $a^{\frac{p-1}{2}} \equiv -1 \pmod{p}$, **then**
3. Declare $\text{ord}_p(a)$ is $p-1$; **return** a as a witness.
4. **else if** $a^{\frac{p-1}{2}} \equiv -1 \pmod{p}$, **then**
5. Quantum order finding. Compute $\text{ord}_p(a)$:
6. **if** $a^k \equiv 1 \pmod{p}$ using quantum order finding algorithm, **then**
7. Declare $\text{ord}_p(a)$ is k ; **return** a as a quantum certificate of $\text{ord}_p(a)$.
8. **end if**
9. **end if**

Output: $\text{ord}_p(a)$.

Here we present quantum part of the algorithm by revisiting Shor's quantum algorithm [3]. The following steps are used to compute order of an integer a in $\mathbb{Z}_p^*$, that is k such that $a^k \equiv 1 \pmod{p}$ on a quantum

computer. For the algorithm, the machine has two quantum registers: source (input) register and target (output) register, both hold integers in the form of binary representation. We now describe how the algorithm works:

Step 1: Given a prime number $p \in \mathbb{N}$ such that $p \leq q \leq 2p$, where $q = 2^n$.

Step 2: In this step both the source and target registers are initialized in the state $|0\rangle|0\rangle$.

Step 3: We put the source register in the uniform superposition of states representing integer $x(\bmod p)$, load the target register with all zeros. This leaves the machine in the following state:

$$|\Psi_1\rangle = \frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |x\rangle |0\rangle. \quad (11)$$

Note that this keeps each qubit of source register into the superposition $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$

Step 4: Choose a random $a \in (1, p)$ such that $\gcd(a, p) = 1$. Since p is prime, it always holds $\gcd(a, p) = 1$. So, we do not require to compute $\gcd(a, p)$.

Step 5: Next, we compute $a^x(\bmod p)$ in the target register by applying a gate U_a . This leaves the machine in the following state:

$$|\Psi_2\rangle = \frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |x\rangle |a^x(\bmod p)\rangle. \quad (12)$$

Step 6: Apply the quantum Fourier transformation on the source register that maps $|x\rangle$ to the following state:

$$\frac{1}{\sqrt{q}} \sum_{z=0}^{q-1} \exp\left(\frac{2\pi i x z}{q}\right) |z\rangle. \quad (13)$$

This leaves the machine of (12) in the following state:

$$|\Psi_3\rangle = \frac{1}{q} \sum_{x=0}^{q-1} \sum_{z=0}^{q-1} \exp\left(\frac{2\pi i x z}{q}\right) |z\rangle |a^x(\bmod p)\rangle. \quad (14)$$

Step 7: To detect periodicity in a^x , we observe both $|z\rangle$ in the source register and $|a^x(\bmod p)\rangle$ in the target register of the machine and measure both the arguments of this superposition for obtaining the values of $|z\rangle$ in the first argument and some $|a^r(\bmod p)\rangle$ as the response for the target register ($0 < r < k$).

Step 8: To extract the required value of k , we apply the concept of probability distribution as follows:

Given the pure state $|\Psi_3\rangle$, the probability of different results for this measurement are given by the probability distribution.

$$\begin{aligned} \text{Prob}(z, a^r \pmod{p}) &= \left| \frac{1}{q} \sum_{x=0; a^x \equiv a^r \pmod{p}}^{q-1} \exp\left(\frac{2\pi i x z}{q}\right) \right|^2 \\ &= \left| \frac{1}{q} \sum_{B=0}^{\lfloor (q-r-1)/k \rfloor} \exp(2\pi i (yk+r)z/q) \right|^2 \\ &= \left| \frac{1}{q} \sum_{B=0}^{\lfloor (q-r-1)/k \rfloor} \exp\left(\frac{2\pi i y \{kz\}}{q}\right) \right|^2, \end{aligned} \quad (15)$$

where $\{kz\}$ means $kz \pmod{p}$. This can be expressed as $-\frac{k}{2} \leq \{kz\} \leq \frac{k}{2}$. This implies that $-\frac{k}{2} \leq kz - tq \leq \frac{k}{2}$ for some t . So, $\text{Prob}(z, a^r \pmod{p}) > \frac{1}{3k^2}$. We then have $|\frac{z}{q} - \frac{t}{k}| \leq \frac{1}{2q}$. Since $\frac{z}{q}$ is known, k can be obtained by the continuous function expansion of $\frac{z}{q}$.

Step 9: The algorithm outputs the required value of k and stops running it.

Continued Fractions

In step 8 of the algorithm 2, we use continuous fractions to approximate a real number using finite number of integers. The following definition and example can help us to understand better the concept of continued fractions.

Definition: A real number α can be approximated by set of positive integers $c_0, c_1, \dots, c_n$ as $CF_n = c_0 + \frac{1}{c_1 + \frac{1}{c_2 + \frac{1}{\dots + \frac{1}{c_n}}}} = \frac{\beta_n}{\gamma_n}$, where β_n and γ_n are integers.

$$\text{For example, } \pi = \frac{22}{7} = 3.14, \quad 3.14 = 3 + \frac{14}{100} = 3 + \frac{1}{\frac{100}{14}} = 3 + \frac{1}{7 + \frac{2}{14}} \approx \frac{22}{7}.$$

Computational complexity

The proposed algorithm has two parts of computation: (a) conventional computation and (b) quantum computation. One can efficiently compute

$a^{\frac{p-1}{2}}$ by standard square and multiplication approach which take $\mathcal{O}\left(\left(\log_2\left(\frac{p-1}{2}\right)\right)(\log_2 p)^2\right)$ bit operations on a conventional computer, and quantum part of the algorithm for computing $a^k \equiv 1(\text{mod } p)$ takes $\mathcal{O}((\log_2 p)^{2+O(1)})$ qubit operations alike [6]. Therefore, the computational complexity for computing multiplicative order of an integer modulo a prime requires $\mathcal{O}((\log_2 p)^{2+O(1)})$ qubit operations in a worst-case scenario.

Conclusion

In this paper, we have presented a quantum algorithm for computing multiplicative order of an integer a modulo a prime p . This is a polynomial-time quantum algorithm and it requires $\mathcal{O}((\log_2 p)^{2+O(1)})$ qubit operations on a quantum computer. It is noteworthy that the proposed quantum algorithm is much more efficient than the conventional algorithm for computing multiplicative order of an integer modulo a prime.

References

- [1] Adleman, L. M. and McCurley, K. S., "Open problems in number theoretic complexity II", *Lect. Notes Comput. Sci.* 877, 291-322 (1994).
- [2] Crandall, R. and Pomerance, C., "Prime numbers: A computational perspective", Springer-Verlag, New York (2001).
- [3] Shor, P. W., "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", *SIAM J. Comput.* 26, 1484-1509 (1997).
- [4] Burton, D. M., "Elementary Number Theory", McGraw-Hill Higher Education, 6th Edition, (2005).
- [5] Nielsen, M. and Chuang, I., "Quantum Computation and Quantum Information", Cambridge University Press, New York, (2000).
- [6] Vandersypen, L.M.K., Steffen, M. and Breyta, G et al., "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance", *Nature* 414(6866), 883-887 (2001).
- [7] Coppersmith, D., "An approximate Fourier transform useful in quantum factoring". Technical Report RC19642, IBM, (1994).
- [8] Hales, L. and Hallgren, S., "An improved quantum Fourier transform algorithm and applications", *Proceedings 41st Annual Symposium*

on Foundations of Computer Science: 515-525 (November 12-14, 2000).

- [9] Manin, Yu.I., "Vychislimoe i nevychislimoe (computable and noncomputable)", Soviet Radio, 13-15 (1980).
- [10] Feynman, R., "Simulating physics with computers", *International Journal of Theoretical Physics*, 21(6/7), 467-488 (1982).
- [11] Feynman, R., "Quantum mechanical computers", *Optics News*, 11, 11-20 (1985).
- [12] Benioff, P.A., "Quantum mechanical Hamiltonian models of Turing machines", *Journal of Statistical Physics*, 29(3), 515-546 (1982).
- [13] Deutsch, D., "Quantum theory, the Church-Turing principle, and the universal quantum Turing machine", In *Proceedings of the Royal Society of London*, Vol. A400, 97-117 (1985).
- [14] Deutsch, D., and Jozsa, R., "Rapid solution of problems by quantum computation", In *Proceedings of the Royal Society of London*, Vol. A439, 553-558 (1992).
- [15] Simon, D., "On the power of quantum computation", *SIAM Journal on Computing*, 26(5), 1474-1483 (1997).
- [16] Bernstein, E. and Vazirani, U., "Quantum complexity theory", *SIAM Journal on Computing*, 26(5), 1411-1473 (1997).
- [17] Childs, A.M. and Ivanyos, G., "Quantum computation of discrete logarithms in semigroups", *Journal of Mathematical Cryptology*, 8(4), 405-416 (2014).
- [18] Chau, H.F. and Lo, H.K., "Primality test via quantum factorization", *International Journal of Modern Physics C* 08, 131-138 (1997).
- [19] Vela, A.D. and Escartin, J.C.G., "A quantum primality test with order finding", *Quantum Information and Computation*, 18(13/14), (2018).
- [20] Proos, J. and Zalka, C., "Shor's discrete logarithm quantum algorithm for elliptic curves", *Quantum Information and Computation*, 3(4), 317-344 (2003).
- [21] Maslov, D., Mathew, J., Cheung, D. and Pradhan, D.K., "An $O(m^2)$ -depth quantum algorithm for the elliptic curve discrete logarithm problem over $GF(2^m)$ ", *Quantum Information and Computation*, 9(7), 610-621 (2009).

- [22] Haner, T., Roetteler, M. and Svore, K.M., "Factoring using $2n + 2$ qubits with Toffoli based modular multiplication", *Quantum Information and Computation*, 18(7/8), 673-684 (2017).
- [23] Eker, M., and Hastad, J., "Quantum algorithms for computing short discrete logarithms and factoring RSA integers". In Lange and Takagi [28], 347-363 (2017).
- [24] Beauregard, S., "Circuit for Shor's algorithm using $2n + 3$ qubits", *Quantum Information & Computation*, 3(2), 175-185 (2003).
- [25] Roetteler, M., Naehrig, M., Svore, K.M. and Lauter, K., "Quantum resource estimates for computing elliptic curve discrete logarithms", available at arXiv:1706.06752, (2017).
- [26] Dam, W.V., Hallgren, S., and Ip, L., "Quantum Algorithms for Some Hidden Shift Problems", *SIAM J. Comput.*, 36(3), 763-778 (2006).
- [27] Griffiths, R.B., and Niu, C.S., "Semiclassical Fourier transform for quantum computation", *Phys. Rev. Lett.*, 76, 3228-3231 (1996).
- [28] Kitaev, A.Y., "Quantum measurements and the Abelian Stabilizer Problem", <https://arxiv.org/abs/quant-ph/9511026> (1995).
- [29] Abrams, D.S. and Lloyd, S., "Quantum algorithm providing exponential speed increase for finding eigenvalues and eigenvectors", *Phys. Rev. Lett.*, 83, 5162-5165 (1999).
- [30] Rivest, R., Shamir, A. and Adleman, L., "A method for obtaining digital signature and public-key cryptosystem", *Communication of the ACM*, 21(2), 120-126 (1978).
- [31] Musaeed Abouaroek & Khaleel Ahmad (2021) Performance analysis of NTRU algorithm with non-post-quantum algorithms, *Journal of Discrete Mathematical Sciences and Cryptography*, 24:5, 1349-1363, DOI: 10.1080/09720529.2021.1932926

Received July, 2021

Revised January, 2022.