Cryptographic Attack Possibilities over RSA Algorithm through Classical and Quantum Computation

Kapil Kumar Soni
Computer Science and Engineering
MANIT, Bhopal, INDIA
prof.kapilsoni@gmail.com

Akhtar Rasool
Computer Science and Engineering
MANIT, Bhopal, INDIA
akki262@yahoo.co.in

Abstract — **Cryptographic attack possibilities have several parameters and one of the possibilities is to attack over the cryptographic algorithm. Large integer factorization is still a challenging problem since the emergence of mathematics and computer science. Benchmark cryptographic protocol, the RSA Algorithm requires factorization of large integers. Classical computation does not have any polynomial time algorithm that can factor any arbitrary large integer. The remarkable but not efficient, classical algorithms for integer factorization are Trial Division, General Number Field Sieve and Quadratic Sieve. The influence of Shor's algorithm assures to get the efficient solution of such factorization problem in polynomial time and challenges the security parameters of the existing cryptosystem, but algorithm implementation limits to be executed on a quantum computer. The article illustrates the algorithms along with flowcharts and implements, Trial Division, Quadratic Sieve Algorithm and Shor's Algorithm for factoring integers and lastly concludes with the observed facts and analyzed results.**

Index Terms — **Quantum Computing, Cl-Bit & Qu-Bit, Factorization, Trial Division, Quadratic Sieve, Shor's Algorithm.**

I. Introduction

Quantum computing explores theoretical understanding over mathematical properties of quantum computation system such as superposition, entanglement for performing the simultaneous operations on data. In quantum computation the data is inherently available for parallel computation purpose; due to the superposition observed by the quantum bit states i.e. 0 & 1 can be present at same time. As far as the current technology is concerned, it provides the efficient computations by providing speed-ups over classical solutions, as could get in Shor's integer factorization [1],[2].

The RSA algorithm is used for cipher design and its protocol works on the principle of integer factorization that breaks out the composite number into a product of prime numbers. No cryptanalytic algorithm has implemented so for that can factor attacked integers in polynomial time i.e. O(n) time. Due to the complexity of such problem, the cipher becomes unbreakable and hence its cryptanalysis is one of the hard problem exist in computer science [3],[5],[6]. The known plaintext attack on a cryptosystem by brute force can be taken into consideration that the received plaintext / cipher text pair of given cryptosystem and target is to find secret key. The cryptosystem may be either symmetric cipher or public key cipher; the strength of such cryptosystem depends on the difficulty of breaking key [17]. A brute force attack tries to break the cryptosystem by searching the key by querying the encryption function successively with all possible keys until the cipher text does not get find. If the key space is "n" bits large then the total search space becomes "2^n" possibilities, so the classical computational brute force technique thus takes on average $2^n/2$ steps to find the accurate combination of key. Mathematically, a classical computers need at least ($\log_2 2^n = \log_2 10^k \approx k = n/\log_2 10$) processing steps [16],[17].

II. Quantum Methodology

A. Fundamentals of Quantum Computing

The basic idea behind Quantum Computation is superposition which allows the Quantum Bit to remain in state "0" and "1", both at same time. So, "n" Quantum Bits can support parallel computation of "2^n" different states possibilities. In addition to this, one might observe that the quantum computer can perform exponential number of operations in single step execution by following intrinsic parallelism, but the computational results for correct solutions are to be compromised with certain probability [1]. The available parallel computations can be ended through the measurement process that allows the system to sustain in deterministic state [2].

In quantum computing, a Qu-Bit (Quantum Bit) is a basic piece of quantum information, a Qu-Bit in quantum computing is similar to a bit in classical computing. The two states in which a Qu-Bit may be measured are known as basis states. A Qu-Bit is a unit vector over the complex vector space - C^2, and its superposition computational basis state is "$|\Psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$", with the sum of squares of both probability amplitudes (complex coefficients) equated to **1**, that is to say $|\alpha_0|^2 + |\alpha_1|^2 = 1$ [1],[4]. At times the Qu-Bit can be simultaneously present in both states, known as superposition. Multiple Qu-Bits remain simultaneously in superposition forms an entanglement and hence the computations that can a quantum computer perform becomes exponential. The depiction of single Qu-Bit can be visualized by considering the block sphere model where Qu-Bits are representing as a vector pointing from the centre of a sphere to its surface at unit radius. Entanglement ensures that the group of quantum particles present in quantum system cannot be identified independently of each other [1],[2].

Extraction of super-positioned quantum information is known to be measurement i.e. to get the stored information from Qu-Bits instantaneously. Measurement process is applied probabilistically, but outcome will be deterministic, but it is non-reversible due to possibly sudden change in Qu-Bit state. Qu-Bit residing with basis state i.e. $|\Psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, while measuring information obtains **'0'** with probability $|\alpha_0|^2$ (the state of Qu-Bit become $|0\rangle$) and '**1**' with probability $|\alpha_1|^2$ (the state of Qu-Bit become $|1\rangle$). Quantum system also not allows copying quantum information as can be done in classical system just due to highly randomized behaviour of quantum information state [1],[2],[4].

B. Classical Computing & Quantum Computing

The comparison between quantum and classical computing model based on theoretical facts between Cl-Bit and Qu-Bit are being discussed in the Table I, The distinguishing parameters show processing capabilities of classical & quantum computations [1],[4].

Table I. Distinguishes Classical and Quantum Computing[1],[4]

CLASSICAL COMPUTING	QUANTUM COMPUTING
Deterministic Model of Computation with access of information on the state of Cl-Bit	Probabilistic Model of Computation with Unitary Transformations on Qu-Bit State
Uses Classical Bit for Information Storage	Uses Quantum Bit for Information Storage
Single Classical Register at a time contains one Classical Bit	Single Quantum Register at a time can have many bits of information
Classical Bits Can Be Copied and Allows Disturbing	Quantum Bits Can Not Be Copied Without Disturbing
Predicted and Finite Classical Outcomes for Classical Bit	Unpredicted and Infinite Superposition Outcome for Qu-Bit
Mutual Exclusive Behaviour of Classical Bits	Entangled Behaviour of Qu-Bits
Perhaps Sequential Processing	Highly Exponentially Parallel
Classical Logic Gates, Not Reversible (Except NOT Gate)	Quantum Logic Gates with Reversibility
Measurement Not Required, Due to Two Possibilities	Does Quantum Measurement Due to Superposition

III. CLASSICAL & QUANTUM FACTORIZATION METHODS

A. Classical Method: Trail Division Algorithm

Trial division is a kind of brute force method that divides the given number "N" to be factored by each number up-to one less than that. The method tries to find the factors on the basis of divisibility condition and starts dividing from smaller divisor and so on. If the number "N" is not divisible by any smaller number than the algorithm avoids checking the other large numbers present in a multiple of that number [3],[10]. The algorithm also reduces the computational effort by predicting a prime numbers as the possible factors of "N". Trial division assures to generate the factor as it validates all possibilities of finding factors, and on the basis of factor the algorithm decides the factor to be prime or composite [10],[11].

B. Classical Method: Quadratic Sieve Algorithm

The quadratic sieve uses classical computation and an extension of general number field sieve factorization method because the algorithm allows finding the factors that may be present in the prime powers with respect to N. The algorithm becomes fastest that general number sieve in order to factor a number up-to 1024 bits long [3]. The main idea behind the quadratic sieve is that if we have "a" and "b" such that $a^2 = b^2 \pmod N$ then we have $a^2 - b^2 = n \pmod N$ equivalent as $(a - b)(a + b) = N \pmod N$. Therefore we can take either "a – b" or "a + b" and test if it contains a non-trivial factor of N using the Euclidean algorithm [3],[10],[11].

C. Quantum Method: Shor's Algorithm

Integer factorization is not efficiently solvable on classical computers, though polynomial time factorization algorithm was proposed by Peter Shor that can do the prime factorization. Algorithm finds the order "r" of an element "a (mod N)" on the basis of randomization and reduces the overall computational efforts to be in polynomial time [3],[5]. The algorithm uses Euclidean method to compute the order "r" of number "a" by selecting a (mod n) at random, and then evaluates $\mathrm{GCD}(a^{r/2} - 1, n)$ in polynomial time, where $(a^{r/2} - 1)(a^{r/2} + 1) = a^r - 1 \equiv 0 \pmod N$, then algorithm finds two factors of "N" as $\mathrm{GCD}(a^{r/2} - 1, n)$ & $\mathrm{GCD}(a^{r/2} + 1, n)$ [5],[6],[7].

Shor's Algorithm Description, Analysis and Flowchart:-

The Algorithm Steps: [5],[6],[8],[9].

1. **Selects the random integer "a" such that 1 < a < N, where "N" is the number to be factored………**(Random number generation requires O(1) time)**.**
2. **Computes "x" as x = GCD(a, N), generally it does using Euclidean algorithm………**(GCD computing requires O(log N) time)**.**
3. **Verifies if (x > 1 or x ≠ 1), then "a" becomes non-trivial factor of N and algorithm ends**

by returning pair (x, N/x)………(Verification requires O(1) time)**.**

4. **If verification fails, then algorithm uses period finding subroutine to find "r" i.e. period of function "f" in order of "a" with respect to "N" and function is f(x) = a^x (mod N)………………..**(Does modular exponentiation and requires O(N^2 log N log(log N)) ≈ O(n^3), where n = No. of Qu-bits in N)**.**

5. **If "r" satisfies "odd" or "$a^{(r/2)}$ ≡ - 1 (mod(N))", so go back to Step 1, otherwise find non-trivial factors of "N" as GCD ($a^{(r/2)}$ +1, N) & GCD($a^{(r/2)}$ -1, N)…………**(Overall required time is O(n^3) or O(log N)^3)**.**

Shor's Algorithm Circuit Description with Example:-

The circuit description can be referred in Fig. 1 and the corresponding Algorithm's Flowchart is referred by Fig. 2, both pretends the idea of Shor's algorithm which is based upon the classical finding of order "r", but by achieving exponential speedup. In the consideration of example, if suppose the number "N = 15" to be factored, then initially a random number "a" is chosen from the set = {1,…,N}= {1,…,15} say "a = 7", so the classical algorithm finds all successive powers of "a" by function " f(x) = a^x mod N", and there must exist any "x" for which "a^x mod N will definitely repeat the outcome after successive powers" then the value of "x" is selected as the order of the function "f(x)". As in taken example {7^1 mod 15 = 7, …. , 7^5 mod 15 = 7, …. , 7^4 mod 15 = 1, …. , …. , 7^8 mod 15 = 1} where outcome set also belong to {1, …, N}. Now in our case, repeated outcome is "1" that is being obtained after every 4th successive power of 7. So, in order to find the factor of given number "N =15", the value of order "r" is chosen as "4", and hence the algorithm computes the non trivial factors of "N" as {GCD($7^{4/2}$ - 1, N) = 3, GCD($7^{4/2}$ + 1, N) = 5} and "3.5 = 15" the algorithm verifies the output.

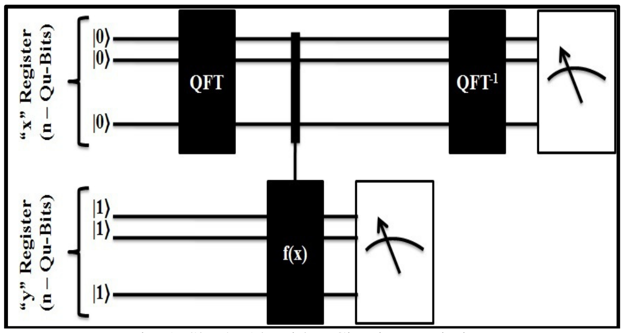

Fig. 1: Shor's Algorithm Circuit Description

The Shor' algorithm does the same as we discussed in the example, by taking clever combination of Hadamard gate and classical reversible gate operations. Shor's algorithm computes the value of "f(x)" as superposition of outcomes for "f(x)" entangled with as superposition of all possible inputs "x", simple saying that the algorithm computes all possibilities simultaneously. The circuit implementation of such algorithm is likely to be tedious to understand but superficially we can simulate the process of computation.

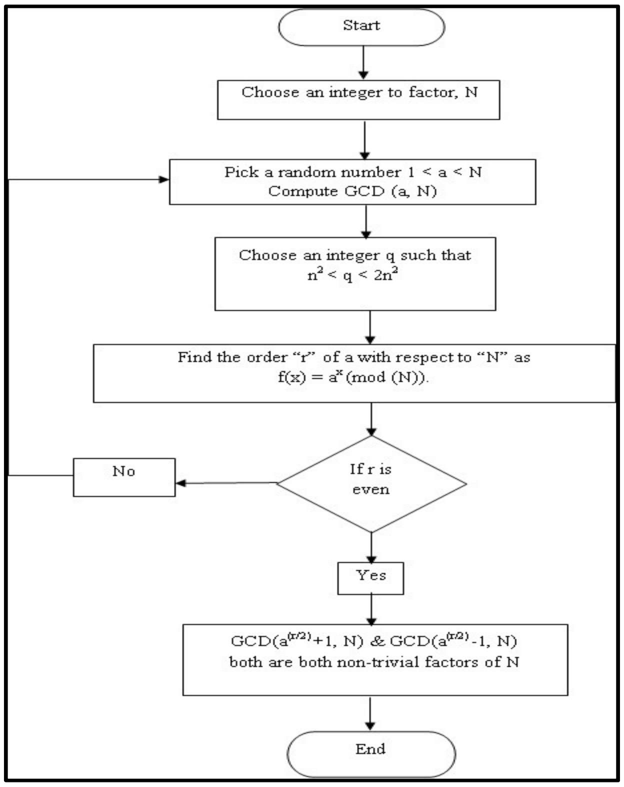
Start
Choose an integer to factor, N
Pick a random number 1 < a < N
Compute GCD (a, N)
Choose an integer q such that
n² < q < 2n²
Find the order "r" of a with respect to "N" as
f(x) = aˣ (mod (N)).
If r is even
No
Yes
GCD(a^(r/2)+1, N) & GCD(a^(r/2)-1, N)
both are both non-trivial factors of N
End

Fig. 2: Flowchar of Shor's Factorization Algorithm [5][6]

As an input, two registers "x" and "y" are taken of size $\log_2 N$ = n, where "N" is the number to factor and "n" becomes the Qu-Bits required to obtain such factor. Both the registers are put into superposition where register "x" contains the superposition of input "x" and register "y" creates the all possibilities of superposition outcome of "f(x)". As in the Fig. 1, "x" is passed to QFT (Quantum Fourier Transform) that is responsible of performing Hadamard gate operation over the Qu-Bits and allows the superimposed possibilities of inputs. For the value of randomly selected "a" the circuit prepares superposition of register "x", and then "f(x)" i.e. modular exponentiation sub-circuit generates all possible outcomes to find the value of "r", and at this point the register "y" will have the following state as $\sum_1^N |x\rangle|f(x)\rangle$. Such superposition state will always ignore the non zero possibility of outcome and selects the sub space where amplitudes are set to one.

As the circuit finds the value order "r", it needs to be verified again just due to the probabilistic model of computation i.e. quantum computation. So, the last step of circuit which is equivalent to quantum algorithm performs this verification phase by applying Inverse Quantum Fourier Transform (QFT^{-1}) that takes the measured order "r" from register "x" with high probability and tries to find the greatest common divisor values for both {GCD($7^{r/2}$ - 1, N) =

f_1, GCD($7^{r/2}$ + 1, N) = f_2} where $f_1.f_2$ = N. If both the simultaneous computed values are not measured as correct one, then the algorithms needs the repetition or the iteration of whole algorithm for the new value of "a". This is just happening due to the probabilistic nature of machine that can be considered as one of the drawback to handle with the quantum computations. But advantageously one can expect the all exponentiation based computational possibilities in single step of execution hence can be taken into the account of achieving the speedup in the comparison to classical computation.

IV. QUANTUM ALGORITHMIC SIMULATION - LIBQUANTUM

Quantum computer simulators are developing to analyse the feasibilities of quantum computations. And also to facilitate the execution of quantum program over the available classical computers, supported at last by the capability of output measurement in execution environment [12],[14]. LibQuantum needs "c – language" compiler with complex number support. The simulator works on Linux platform. And it has the following features as [13],[15]:

- Quantum algorithm simulation becomes possible.
- Performance is high and memory consumption is too low.
- Supports the realization of Decoherence
- LibQuantum is freely available over GNU General Public License (GPL) as version 3.
- LibQuantum provides C library for the simulation of quantum computations.
- It is available with updated library and supports the general quantum simulation.

V. IMPLEMENTATION, VALIDATION & RESULTS DISCUSSION

A. Implementation and Validation Methodology

The article outcome is the research implementation of Trail Division, Quadratic Sieve and Quantum Shor's Factorizing Algorithms and the results are compared theoretically and practically by using the kind of results which are tested practically, with the increase in "N" (number to factor), that require gradual increase of Cl-Bits and Qu-Bits, and analyzed through the obtained results over classical and quantum computers.

B. Empirical and Practical Evaluated Results

Empirically the algorithms are found to be suitable for factoring the given number within the stipulated time. As per the theoretical complexity is concerned, Shor's proved to be best, but only on quantum computers, rather quadratic sieve proves the challenge over classical cryptosystem. The below mentioned Table II and Table III shows the empirical complexities of classical and quantum factoring algorithms and shows the practical evaluated results of the implemented version of classical algorithms and the simulated version of Shor's using the suggested simulator environment. All results are tested over classical computer, so Fig. 3, shows the actual, comparative growth and running time of classical and quantum factoring algorithms [5],[6],[7],[10],[11],[18].

Table II. Empirical Complexities of Classical & Quantum Algorithms

Worst Case	Shor's Algorithm	Quadratic Sieve Method	Brute Force Method
	$O(\log N)^3$	$\left(e^{\sqrt{\ln(n)\ln(\ln(n))}}\right)$	$O(2^n)$

Table III. Evaluated Classically Implemented & Quantum Simulated Algorithms Result (in Sec.)

"N" No. to Factorize	Trial Division Algorithm	Quadratic Sieve Algorithm	Shor's Quantum Algorithm
15	0.000083	0.000006	0.002662
21	0.000090	0.000006	0.006453
91	0.000119	0.000006	0.652305
161	0.000124	0.000006	1.228225
221	0.000137	0.000008	2.49871
323	0.000144	0.000008	7.297567
377	0.000146	0.000008	16.189502
667	0.000150	0.000009	36.45686

C. Results Analysis & Observed Facts

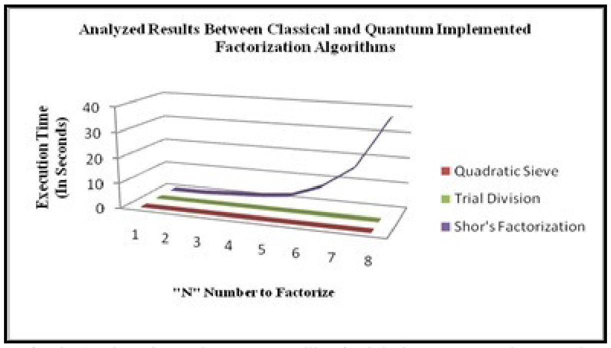

Fig. 3: Analyzed Results Between Classical & Quantum Implemented Factorization Algorithms

As per the theoretical complexities of factoring algorithms are concerned, the growth of classical algorithms are exponential but Quadratic Sieve's growth is less than Trial Division, hence found more efficient and fast classical algorithm. Shor's Algorithm is the fastest algorithm between all due to polynomial complexity, but has the limitation to be implemented over only the quantum computers. Do remember that the quantum computer simulator works upon the classical machine, hence we have to compromise for gaining actual speedup and parallelism (i.e. overall performance). As per the evaluated results the running time of Trial Division varies with the gradual increase in "N" i.e. the number to be factored, but with same instances the running growth of time of Quadratic Sieve becomes almost constant, and proves to be best algorithm which is highly efficient for factoring large numbers.

One can clearly bserve from Fig. 3, though theoretically Shor's algorithm is fastest, but the observed findings give us a different view. Shor's Algorithm run on classical computer simulator cannot give us the desired results. As quantum computer is not available yet, the exact results cannot be obtained. Shor's algorithm comprises to be a challenge over cryptography system as it can factor large integer in polynomial time instead of classical exponential time algorithms. But due to the probabilistic algorithm, we cannot always guarantee about desired output.

VI. CONCLUSION

Integer factorization is a hard problem and no classical algorithm is still available that can factor integers in polynomial time, reason being the well-known cryptographic systems are still secured. We considered the RSA cryptosystem based on factorization problem, with the attack possibilities using classical and quantum computer system. In this paper we have compared three methods of factorizing an integer namely Quadratic Sieve, Trial Division and Shor's Algorithm. Classical Quadratic Sieve is found efficient to factor large integer than Trial Division, and theoretically Shor's Algorithm finds superior among all, as the algorithm takes polynomial time to factor the number in less than its classical counterparts and challenges the security parameters of the existing cryptosystem, but algorithm implementation limits to be executed on a quantum computer. In concluding facts, the realistic quantum computer existence will break the security parameter of RSA algorithm and hence challenges the overall cryptosystem.

REFERENCES

[1] Michael A. Nielsen and Isaac L. Chuang, "Quantum Computation and Quantum Information", 10th Anniversary Edition, 2010, UK: Cambridge University Press.

[2] N. David Mermin, "Quantum Computer Science – An Introduction", I Edition, August 2007, Cambridge University Press.

[3] Shah Muhammad Hamdi, Syed Tauhid Zuhori, Firoz Mahmud, Biprodip Pal, "A Compare between Shor's Quantum Factoring Algorithm and General Number Field Sieve", In: IEEE International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT), PP No. 1-6, October 2014.

[4] Adina Barila, "From classical computing to quantum computing", 12th International Conference On Development And Application Systems, Suceava, Romania, IEEE Conference, PP No. 198 – 203, May 2014.

[5] P. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring", In: IEEE 35th Annual Symposium on Foundations of Computer Science, PP. No. 124–134, Nov 1994.

[6] A. Ekert and R. Jozsa, "Quantum computation and Shor's factoring algorithm" Reviews of Modern Physics, Vol. 68, PP. No. 733-753, 1996.

[7] Enrique Martin-Lopez, A. Laing, T. Lawson, Xiao-Qi Zhou, and J. O'Brien, "Experimental realization of Shor's quantum factoring algorithm using qubit recycling", Nature Photonics, Vol. 6,PP. No. 773–776, arXiv:1111.4147, 2012.

[8] Zhengjun Cao, Lihua Liu, "On the Complexity of Shor's Algorithm for Factorization", In: IEEE Second International Symposium on Information Science and Engineering, PP.No. 164-168, April 2009.

[9] Shweta Nagaich, Y.C.Goswami, "Shor's Algorithm For Quantum Numbers Using MATLAB Simulator", In: Fifth International Conference on Advanced Computing & Communication Technologies, PP. No. 165-168, April 2015.

[10] Jelena M. Smiljanić; Predrag N. Ivanis, "Attacks on the RSA cryptosystem using integer factorization", In: 19th Telecommunications Forum (TELFOR) Proceedings of Papers, PP. No. 550-553, November 2011.

[11] M. Cosnard; J. Philippe, "The Quadratic Sieve Factoring Algorithm on Distributed Memory Multiprocessors", In: IEEE Proceedings of the Fifth Distributed Memory Computing Conference, PP. No. 254-262, April 1990.

[12] IBM Quantum Experience – IBM Research: http://research.ibm.com/ibm-q, http://research.ibm.com/ibm-q/quantum-primer, http://research.ibm.com/ibm-q/research.

[13] LibQuantum: The C Library for Quantum Computing and Quantum Simulation, http://www.libquantum.de/.

[14] Tomi H Johnson and et al, "What is Quantum Simulator", EPJ Quantum Technology through Springer, PP. No. 1–10, Jul 2014.

[15] Johan Brandhorst – Satzkorn, "A Review of Freely Available Quantum Computer Simulation Software", http://liu.diva.portal.org/smash/get/diva2:544257/FULLTEXT01.pdf, June 2012.

[16] Andreas de Vries, "Quantum Computation: An Introduction for Engineers and Computer Scientists", I Edition, 2012, Books on Demand Publication - Germany.

[17] George F. Viamontes, Igor L. Markov, John P. Hayes, "Quantum Circuit Simulation", I Edition, 2009, Springer Publication – US.

[18] Neal Koblitz, "A Course in Number Theory and Cryptography", II Edition, 2008, Springer India Private Limited.