Impact of Quantum Computing on RSA Cryptographic Security

Popuri Sushanth, Tejaswini Kancharla, and B. Soujanya*
Department of Computer Science and Engineering, GSCSE, GITAM (Deemed to be University), Visakhapatnam, India, Email Id- sbuddhar@gitam.edu

Abstract: The security of the RSA cryptosystem relies on the assumed classical intractability of large integer factorization. However, Shor's quantum algorithm demonstrates that factoring can be performed in polynomial time on a sufficiently large, fault-tolerant quantum computer, theoretically undermining RSA's foundational security assumption. The present paper provides a comparative analysis of RSA in relation to quantum computing through a combination of classical benchmarking, asymptotic quantum complexity modelling, and simulation-based analysis of noise in simplified Shor-like quantum computing. A 2048-bit RSA implementation was evaluated to measure key generation and encryption performance under classical computation. These results were contrasted with normalized growth estimates for classical factoring via the General Number Field Sieve (GNFS) and quantum factoring via Shor's algorithm. The analysis demonstrates asymptotic divergence between sub-exponential classical factoring and polynomial time in quantum scaling. Another noise-based simulation experiment aims to assess how circuit depths and gate errors impact the validity of Shor-like circuits. The results reveal the significant hardware limitation that prevents the exploitation of quantum technology. The findings clarify the distinction between theoretical vulnerability and present-day feasibility, contributing to informed long-term post-quantum cryptographic migration strategies.

Keywords:

INTRODUCTION

Quantum computing is a computational paradigm based on the formalism of quantum mechanics, in which information is represented and manipulated using quantum states in a complex Hilbert space. Unlike classical bits, which take definite values in the set {0,1}, quantum bits (qubits) are described as linear combinations of computational basis states, allowing quantum algorithms to exploit interference and entanglement during computation. These features enable non-classical algorithmic behaviours that, for specific problem classes, lead to provable reductions in computational complexity, such as polynomial-time quantum algorithms for problems that admit only sub-exponential-time classical solutions.

***Address correspondence to sbuddhar@gitam.edu** :*Assistant Professor, Department of Computer Science and Engineering, GSCE, GITAM (Deemed to be University), Visakhapatnam, India, Email Id- sbuddhar@gitam.edu

The cryptographic significance of quantum computing arises primarily from its impact on public-key cryptographic schemes whose security relies on the assumed hardness of number-theoretic problems such as integer factorization and the discrete logarithm problem. Shor's technique shows that these issues can be solved in polynomial time if there is indeed a big enough and fault-tolerant quantum computer. This means that the basic security assumptions behind commonly used systems like RSA and Diffie-Hellman are no longer true. Though such cryptographically relevant quantum computers do not exist yet, the theatrical susceptibility is widely recognised.The RSA public-key algorithm's basic security is based on the idea that the integer factorization is hard and difficult to interpret. The system uses two different huge prime integers, p and q, to make a composite modulus N= pq. The security of the private key is just a gamble that a foe won't be able to figure out the original primes from N. Classical computer architectures can only handle sub-exponential time complexities for factoring, which makes 2048-bit keys effectively unbreakable today. However, Shor's method for scalable quantum processors might change this.

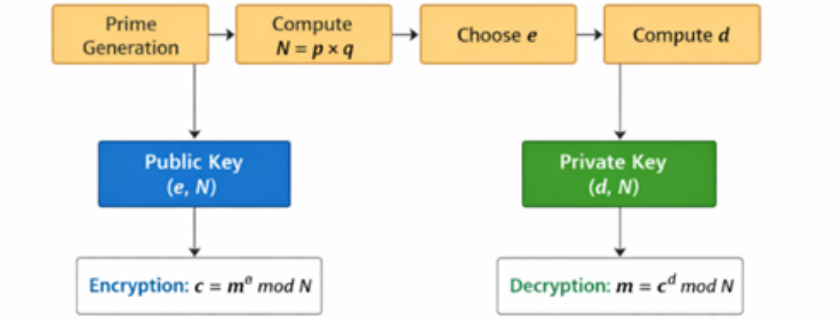

Figure.1. Workflow of the RSA cryptosystem illustrating key generation (prime selection, modulus computation, and exponent selection) followed by encryption using the public key and decryption using the private key

This change in computing capability creates a serious weakness called the "Store Now, Decrypt Later" (SNDL) technique. In this paradigm, unethical people may intercept and store encrypted messages today with the goal of cracking them when quantum gear is ready in the coming future. Sectors including healthcare, international banking, and national security that rely on data privacy for decades are particularly vulnerable to this threat.

These risks can be mitigated, with the advent of quantum cryptography. Rather than complex mathematics, the focus is on the laws of physics. One mechanism that makes advantage of the physical properties of quantum states to allow users to exchange secrets is Quantum Key Distribution (QKD). Constantly breaking the system is the goal of every attempt at eavesdropping. Instead of entirely replacing classical techniques, current developments suggest a shift toward hybrid systems, which combine regular symmetric encryption with quantum-secured key exchange. Ambient noise and decoherence pose significant obstacles to practical implementation, contrary to the assumptions made in theoretical models, which often include flawless quantum processes. Making a quantum computer that can withstand errors from a mathematical theory is no simple task. In order to determine RSA's modern-day security, two factors must be considered: the algorithmic efficacy of Shor's technique and the limitations of the technology required to execute it on a big scale.

RELATED WORKS

Thanks to the NIST standardization work that kicked off in 2016, the shift to post-Quantum Cryptography (PQC) has gone from being a theoretical aim to a clear institutional objective. The existential threat to number-theoretic encryption was exposed decades ago by Shor [1]. To move coordinated research into the implementation phase, the NIST framework has provided the necessary benchmarks. Since traditional computers and quantum computers decompose numbers in very different ways, the disparity poses the greatest danger to the RSA protocol. It is customary to safeguard RSA using the General Field Sieve (GNFS). No exponential complexity is too great for it to handle. The asymptotic difficulty is defined as:

$$\exp\left((\log N)^{1/3}(\log\log N)^{2/3}\right). \qquad (1)$$

In contrast, this problem can be solved in polynomial time using Shor's method. This ability to identify orders contradicts the mathematical concepts that have protected RSA for decades, according to recent rigorous investigations [2] by Datta. More and more academic research in recent years has concentrated on cryptography's foundational structures. Researchers Hiroka and Morimae [5] examined quantum primitives such one-way puzzles and pseudorandom generators

using meta-complexity frameworks. The findings suggest that in the future, cryptographic security may move away from specific number-theoretic problems that may have revealed weaknesses and toward broader average-case hardness assumptions that provide better defense against quantum attackers. In terms of operational priorities, the transition period remains paramount. Along with developing hybrid models that combine classical asymmetric primitives with symmetric quantum-resistant layers, Bhanu Prakash et al. [6] look at the prospect of raising the size of RSA keys as a temporary protection. While RSA is very efficient for today's technology, the "Store Now, Decrypt" tests highlight a major conflict: Migratory preparations must be made immediately in light of the "Later" threat. Despite the abundance of cryptanalytics literature, few studies have examined the relationship between theoretical asymptotic complexity and the performance of RSA in the presence of hardware noise and decoherence. Instead of seeing quantum vulnerability as a variable affected by the physical limitations of modern quantum computers, the majority of current research treats it as a constant in theory.

Three crucial conclusions arise from the current research status:

1. RSA remains secure under classical computational models but is theoretically vulnerable to scalable quantum computation [1] [2]
2. Practical implementation of Shor’s algorithm at cryptographic scale requires substantial quantum resources beyond current hardware capabilities [3].
3. Post-quantum cryptographic research is advancing toward standardized alternatives while exploring deeper complexity-theoretic foundations [4] [5].

However, a gap remains in experimentally grounded comparative analyses that bridge classical RSA execution characteristics, formal quantum complexity derivations, and realistic hardware constraints within a unified framework. This study improves that interaction by explaining the discrepancy between asymptotic vulnerability and current practicality by empirically evaluating RSA computational performance with precisely defined quantum complexity scaling.

PROPSOED WORK

3.1 Threat of Quantum Computing to RSA

Peter Shor suggested a quantum technique in 1994 that could factor big numbers in polynomial time, which went against the idea that RSA was hard. For an RSA modulus N=pq, Shor's algorithm reduces integer factorization to the problem of order finding, which can be solved efficiently using the Quantum Fourier Transform (QFT).

Classical complexity: $\text{Factoring} \in \exp\left((\log N)^{1/3} (\log\log N)^{2/3}\right)$ (2)

Quantum complexity: $T_{\text{Shor}} = O\left((\log N)^3\right)$ (3)

This represents a super-polynomial improvement relative to the best-known classical sub-exponential factoring algorithms.

Resource analyses indicate that factoring a 2048-bit RSA modulus using Shor's algorithm would require large-scale fault-tolerant quantum computation. In particular, surface-code–based estimates suggest that thousands of logical qubits and millions of physical qubits would be required, with runtimes on the order of hours to days under optimistic architectural assumptions [7]. By contrast, classical integer factorization using the General Number Field Sieve exhibits sub-exponential growth in logN, leading to astronomically large computational resource requirements at 2048-bit scale. Precise estimates of runtime are dependent on assumptions about hardware and options for parallelization, but the amount of traditional resources needed quickly becomes unmanageable. Although such large-scale quantum hardware does not currently exist, continued advances in qubit scalability and quantum error correction make this a credible long-term cryptographic threat [7].

3.2 Why Shor's Algorithm Runs in $O((\log N)^3)$

Shor's quantum factoring algorithm provides a polynomial-time method for integer factorization by reducing the problem to quantum period finding (order finding) using the Quantum Fourier Transform [1] [2] This approach was originally introduced by Shor and later analysed in detail in subsequent quantum computing literature. The order-finding subroutine forms the computational core of the algorithm and enables efficient factorization on a quantum computer.

Let N be the integer to be factored and let n = log N denote the input size in bits. The best-known classical factoring algorithm (General Number Field Sieve) runs in sub-exponential time:

$\exp\left(O(\, n^{1/3} (\log n)^{2/3}\,)\right)$ (4)

making classical integer factorization super-polynomial in n [2]. This computational gap between classical sub-exponential algorithms and polynomial-time quantum algorithms forms the basis of the long-term quantum threat to RSA

3.2.1 Quantum Order-Finding Procedure
Assume:

- N is neither even nor a prime power
  $a < N$ is chosen randomly such that $\gcd(a, N) = 1$

Factoring reduces to finding the order r such that: $a^r \equiv 1 \pmod{N}$ [1]

Step 1: Domain Size
We evaluate the function $f(k) = a^k \bmod N$ (5)
in superposition over: $0 \leq k < 2^m$
To guarantee successful period extraction, choose m such that: $N^2 \leq 2^m < 2N^2$
Taking logarithms: $m = O(\log N) = O(n)$
Thus, the input register requires $m = O(n)$ qubits.

Step 2: Modular Exponentiation Cost
To compute: $a^k \bmod N$
we use repeated squaring.

Number of Multiplications
Exponentiation by repeated squaring requires: $O(\log k)$
Since $k < 2^m$, $\log k = O(m) = O(n)$
So, we require: $O(n)$ modular multiplications.

Cost of Each Multiplication
Using standard long multiplication: Integer multiplication cost = $O(n^2)$
Thus, one modular multiplication costs: $O(n^2)$
Total Cost of Modular Exponentiation
$O(n) \times O(n^2) = O(n^3)$ (6)
Hence, computing $f(k)$ costs: $O(n^3)$
Step 3: Preparing Uniform Superposition
To create: (7)

$$\frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m - 1} |k\rangle$$

we apply m Hadamard gates: $m = O(n)$
This contributes: $O(n)$
which is negligible compared to $O(n^3)$.

Overall Time Complexity $T(N) = O(n^3)$

Since $n = \log N$: $T(N) = O((\log N)^3)$ (8)

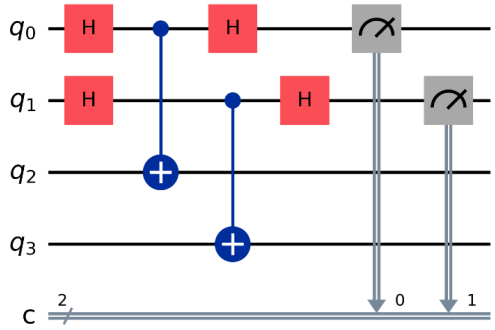

Figure.2. Small-scale illustrative quantum circuit generated using the Qiskit framework

The circuit prepares a superposition using Hadamard gates, applies controlled operations representing modular exponentiation, and measures the control register. The following section presents empirical RSA performance measurements and asymptotic comparisons between classical and quantum factoring.

3.3 Noise Sensitivity Analysis of Shor-like Quantum Circuits

Investigating the effects of background noise and circuit depth on the precision of quantum processes is necessary for moving forward from theory to practice. Shor's method is theoretically designed for polynomial-time speed, but its usefulness in the real world is limited by the physical limitations of modern hardware, namely cumulative gate faults and decoherence. We used the Cirq simulation framework to build a circuit that mimics the order- finding backbone of Shor's approach in order to look at these dynamics. The first step in this architecture is to prepare counting qubits in a uniform superposition using Hadamard gates. Then, controlled operations that are meant to mimic the behaviour of modular exponentiation are carried out. The procedure ends with the use of an inverse Quantum Fourier Transform (IQFT) on the counting register, which makes it possible to find the period before the last measurement is made.

Our exponential design tested two different levels of algorithmic complexity to provide us a baseline for comparison. The initial setup used a short circuit depth that could handle small number factorization, like N=15. The second design used regulated repetition to make the circuit substantially deeper. The opportunity to simulate the additional resources required to process larger RSA moduli presented itself today. To connect gap between perfect theory and real-world hardware, we

included a depolarizing noise model to the simulation and tested a range of gate error probability from p=0 to p=0.1. We were able to get an empirical success probability by running each configuration more than 1,000 times and keeping track of the measurement peaks at the expected success values, which were 2, 4, and 6. This information is very important since it shows the point at which hardware noise starts to outweigh the quantum approach's theoretical benefits.

RESULT AND DISCUSSION

4.1 Classical RSA Performance (2048-bit)

We used the Miller- Rabin test to construct a 2048-bit RSA key pair using the Extended Euclidean Algorithm to get the modular inverse. The value of the public exponent was set to e=65537.

Table 1. Empirical RSA Execution Performance

Metric	Value
Modulus Size	2048 bits
Public Exponent (e)	65537
Key Generation Time	9.9834 s
Encryption Time	0.000228 s
Decryption Verification	Successful
Total Execution Time	10.181 s

4.2 Interpretation

The data demonstrates that the most expensive part of RSA is creating the keys. The main reason for this is because you have to run probabilistic primality testing over and over again to locate enormous prime factors. The encryption process, on the other hand, just takes a few modular multiplications and employs the common public exponent e=65537, therefore it occurs quite rapidly. The fact the test data could be decrypted, shows that both the implementation and the integrity of the key pairs that were made are valid. Ultimately, our results indicate that RSA remains very effective and efficient on contemporary classical systems at a 2048-bit security level.

4.3 Asymptotic Complexity Comparison

To assess long-term security resilience, we performed a comparison study of the growth rates of conventional and quantum factorization. The General Number sieve

(GNFS) is the most efficient known classical technique for factoring big numbers. It is the classical benchmark. Its sub-exponential complexity is modelled as:

$$L_N\left[\frac{1}{3},\left(\frac{64}{9}\right)^{\frac{1}{3}}\right] = \exp\left((\log N)^{1/3}(\log\log N)^{2/3}\right) \quad (9)$$

In this comparison, Shor's quantum algorithm represents the adversarial threat, operating with a polynomial-time complexity of: $O((\log N)^3)$

Even if constant components were left out to make the comparison fair, the difference is clear: when N (the modulus) goes up, the cost of computing for classical systems goes up for faster than the cost of computing for quantum systems, which grows at a cubic rate. This demonstrates that significant improvements in algorithm performance are to blame, rather than the slowness of traditional technology, for RSA's vulnerability.

Table.2 Asymptotic Growth Comparison

Key Size	Classical (GNFS Approx.)	Shor (Polynomial Approx.)
64-bit	5.48×10^3	8.73×10^4
128-bit	1.85×10^5	6.98×10^5
256-bit	2.02×10^7	5.59×10^6
512-bit	1.02×10^{10}	4.47×10^7
1024-bit	3.82×10^{13}	3.58×10^8
2048-bit	1.98×10^{18}	2.86×10^9

Note: Values represent normalized asymptotic growth estimates rather than measured runtimes.

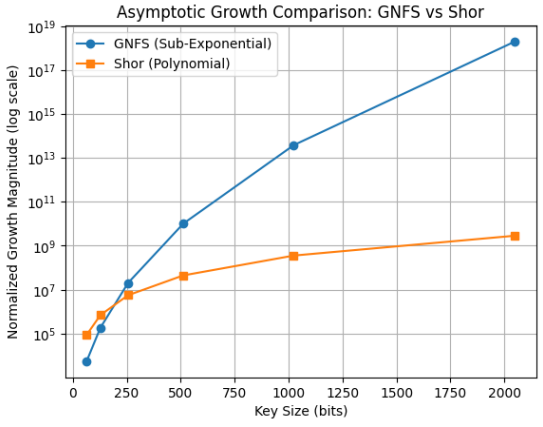

Figure.3. Log-scale comparison of normalized asymptotic growth for classical factoring (GNFS) and quantum factoring (Shor) as RSA key size increases, showing sub-exponential versus polynomial scaling.

4.4 Shor's Interpretation

Due to the addition of polynomial overhead and the elimination of constant factors, Shor's equation may provide a more precise result than the GNFS estimate. However, the disparity widens as the key size increases.

At 2048 bits:

- Around ten times the original size, the GNFS normalized growth magnitude
- The equation of the Shor polynomial is still about 10

This illustrates the fundamental asymptotic distinction:

- GNFS grows sub-exponentially in logN
- Shor grows polynomially in logN0

The growing gap shows how weak things may be when quantum computers can scale up.

4.5 Impact of Noise and Circuit Depth on Shor-like Circuits

The aforementioned theoretical speedups are crucially validated by our simulation findings. The success rate study shoes that there is a definite link between deeper circuits and more vulnerable to quantum noise. Shallow circuits maintained their high fidelity, while deeper circuits that were needed for cryptographically important factorizations quickly lost performance because of gate faults and decoherence.

These results show that Shor's technique is better theoretically, but it can't be used in real life until fault-tolerant technology is further advanced. To factor a 2048-bit modulus, you need a circuit depth and error suppression that current NISQ-era (Noisy Intermediate-Scale Quantum) devices can't provide.

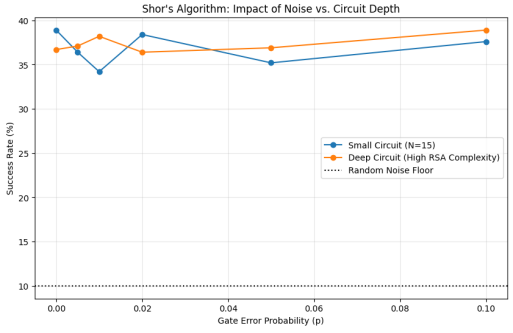

Figure.4. Impact of depolarizing gate noise on the success rate of simulated Shor-like circuits. Two configurations are compared: a shallow circuit representing small-scale factorization and a deeper circuit approximating the increased circuit depth required for factoring larger RSA moduli. The dotted line indicates the random measurement baseline.

4.6 Discussion

4.6.1 Classical RSA Feasibility

Benchmarks show that 2048-bit RSA is still very light for encryption (0.00028 seconds) and easy to use for key creation (0.98 seconds). These numbers show once again that RSA's long life is not limited by how fast it works, but by now possible quantum-based attacks will be in the future.

4.6.2 Asymptotic Divergence Between Classical and Quantum Attacks

At longer bit lengths, the difference between sub-exponential and polynomial scaling becomes the most important aspect. The relative difficulty for a classical system with 2048-bit integers is astronomical compared to the theoretical quantum estimate. This proves that the danger to RSA is only based on algorithms.

The General Number Field Sieve (GNFS), the quickest classical factoring method known, has sub-exponential complexity:

$$\exp\left((\log N)^{1/3}(\log\log N)^{2/3}\right) \quad (10)$$

Shor's algorithm, on the other hand, factors integers in polynomial time: $O((\log N)^3)$

Theoretical Vulnerability vs. Practical Constraints

What technology is capable of doing is light years away from Shor's idea. The following would be required to crack a 2048-bit modulus:

- Thousands of stable logical qubits.
- Millions of physical qubits to support fault-tolerant error correction.
- Deep, high-fidelity circuits that can survive environmental noise.

CONCLUSION & FUTURE WORK

Incorporating both theoretical quantum complexity and actual classical benchmarking, this study offered a detailed assessment of RSA security. Results demonstrate that RSA maintains its computational efficiency and is almost hard to crack for classical systems with a 2048-bit security level. However, the

cryptographic landscape is forever altered by the theoretical pre-eminence of Shor's method. To attack RSA's underlying hardness assumption, it offers a polynomial-time method. Since current quantum technology lacks the necessary qubit count and fault-tolerant error correction to execute Shor's algorithm on a cryptographic scale, this vulnerability is still theoretical at this time. However, the analysis reveals that the remaining lifespan of RSA is now dependent on the rate of technological improvement rather than the strength of the mathematics. Our goal is to refine the ratio of physical-to-logical qubits needed for reliable order-finding operations by expanding our noisy sensitivity investigations. This continuing study will help us understand when exactly postquantum migration may happen by pinpointing the exact hardware requirements for an assault against RSA-2048.

REFERENCES

[1]. Shor, P. W. (1994). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS), 124–134.

[2]. Datta, N. (2020). Shor’s quantum factoring algorithm. DAMTP Quantum Information Lecture Notes, University of Cambridge.

[3]. Brandão, P. R., & Silva, C. (2025). Quantum computing and cryptography. Preprints.

[4]. Subramani, S., et al. (2025). Review of security methods based on classical cryptography and quantum cryptography. Cybernetics and Systems.

[5]. Hiroka, T., & Morimae, T. (2024). Quantum cryptography and meta-complexity. arXiv preprint arXiv:2410.01369.

[6]. Prakash, B., et al. (2025). A numerical and security analysis of RSA: From classical encryption to post-quantum strategies.

[7]. Gidney, C. (2025). How to factor 2048-bit RSA integers with less than a million noisy qubits.

[8]. Bhatia, V., & Ramkumar, K. R. (2020). An Efficient Quantum Computing technique for cracking RSA using Shor’s Algorithm

[9]. Moolchand Sharma , Vikas Choudhary , R. S. Bhatia , Sahil Malik , Anshuman Raina & Harshit Khandelwal (2020): Leveraging the power of quantum computing for breaking RSA encryption, Cyber-Physical Systems