International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 12 | Dec - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Implementation and Evaluation of Shor's Algorithm for RSA Factorization Using QISKIT and IBM Quantum Services

Prathibha S[1], Inchara B M[2], Aishwarya H R[3], Deeksha D R[4], Jyotika M Kadur[5]
[1,2,3,4,5]Dept. of CS&E, PESITM, VTU, Shivamoga, India
pratibha@pestrust.edu.in, bminchara882@gmail.com, aishwaryax1814@gmail.com, deekshadr22@gmail.com, jyotikakadur@gmail.com

***Abstract*—Quantum algorithms that surpass the performance of classical methods are of increasing practical and scientific importance, with Shor's algorithm remaining a landmark result for integer factorization and cryptanalysis. This work presents a full-stack, reproducible implementation of Shor's algorithm using Qiskit, incorporating exact permutation-based modular multiplication unitaries, a scalable Quantum Phase Estimation (QPE) pipeline, and hybrid classical post-processing based on continued fractions and number-theoretic verification. The architecture is designed with modular components that explicitly separate quantum execution, classical analysis, and visualization, enabling detailed introspection of the order-finding procedure. Experimental results show successful factorization of multiple semiprimes using both local Aer simulations and IBM Quantum cloud backends. The cloud-based experiments expose practical constraints such as device noise, circuit-depth limits, and shot availability, while demonstrating portability of the framework to real hardware. The implementation further provides automated circuit-diagram generation, histogram-based phase diagnostics, and optional classical reference routines for comparison. Overall, this work delivers a transparent, educationally oriented, and cloud-deployable platform for studying Shor's algorithm and benchmarking hybrid quantum–classical computing workflows.**

***Index Terms*—Quantum computing, Shor's algorithm, Quantum Phase Estimation (QPE), Qiskit, modular unitaries, IBM Quantum, hybrid algorithms.**

I. Introduction

Integer factorization is a fundamental pillar of modern public–key cryptography, most notably the RSA scheme, whose security depends on the classical difficulty of decomposing large semiprimes into their prime factors [1]. Shor's quantum algorithm disrupts this security premise by enabling polynomial-time factorization through quantum order finding [2]. Although its theoretical impact is well established, practical exploration of Shor's algorithm remains challenging due to the substantial circuit depth required for modular exponentiation, Quantum Phase Estimation (QPE), and controlled unitary operations on current noisy intermediate-scale quantum (NISQ) devices [3], [4].

This work addresses these challenges by presenting a modular, transparent, and reproducible implementation of Shor's algorithm using Qiskit. The design leverages exact permutation-based modular multiplication unitaries, allowing reversible arithmetic without approximation errors and supporting reliable QPE behavior in simulation [5]. The framework integrates robust classical post-processing—including continued fractions, candidate-order verification, and arithmetic consistency checks [6]—along with visualization tools that generate circuit diagrams, measurement histograms, and diagnostic plots.

A key contribution of this system is its portability across execution environments. The same circuits can be executed locally using the Aer simulator or deployed on IBM Quantum cloud hardware [7], [8]. This dual workflow enables empirical analysis of device noise, circuit-depth limitations, and shot constraints, while verifying that the architecture maintains correctness on real quantum processors [9]. These experiments provide practical insights into implementing quantum algorithms on near-term hardware.

Through both simulations and cloud-based runs, the implementation successfully factors several small semiprimes and offers detailed visibility into the hybrid quantum–classical workflow underlying Shor's algorithm. Beyond demonstrating correctness, this work provides an extensible and educational framework for studying quantum order finding, benchmarking hybrid schemes, and supporting future research in quantum cryptanalysis [10]–[12].

II. Background and Theory

Quantum algorithms capable of solving classically intractable problems have attracted significant attention in recent years, with integer factorization standing as one of the most influential examples [1], [2]. Shor's algorithm reduces the problem of factoring a composite integer N to the task of determining the order of an integer a modulo N [8]. This section reviews the mathematical and algorithmic foundations required to understand the architecture and implementation presented in this work.

A. Integer Factorization and RSA Security

Let $N = pq$, where p and q are large primes. Recovering (p, q) from N is believed to be computationally expensive on classical hardware. The fastest classical factoring algorithms, such as the General Number Field Sieve (GNFS), run in sub-exponential time, making RSA cryptography secure for sufficiently large key sizes [3], [4].

In this project, smaller values of N (typically less than 50) are used purely for demonstration. Although such integers are trivially factorable classically, the structure and workflow of Shor's algorithm remain representative of its large-scale counterpart.

B. Overview of Shor's Algorithm

Shor's algorithm consists of a quantum procedure for order finding and a classical routine for extracting the prime factors. The hybrid nature of the algorithm makes it particularly suitable for modern noisy intermediate-scale quantum (NISQ) systems [11], [12].

1) Quantum Order Finding: For an integer a such that $\gcd(a, N) = 1$, the order r is defined as the smallest positive integer satisfying

$$a^r \equiv 1 \pmod{N}. \tag{1}$$

The quantum computer efficiently determines this order using Quantum Phase Estimation (QPE) [8], [9].

2) Classical Post-Processing: Once r is obtained, the factors of N can be recovered using

$$\gcd\left(a^{r/2} - 1, N\right), \qquad \gcd\left(a^{r/2} + 1, N\right). \tag{2}$$

Valid solutions require r to be even and

$$a^{r/2} \not\equiv -1 \pmod{N}. \tag{3}$$

Additional number-theoretic checks are used to ensure that the recovered r is correct [5], [6].

3) Quantum Phase Estimation (QPE): QPE is a fundamental quantum subroutine used to estimate the eigenphase of a unitary operator. In Shor's algorithm, the unitary is defined as

$$U_a : |x\rangle \mapsto |ax \bmod N\rangle,$$

which is a reversible permutation over the integers.

If $|\psi\rangle$ is an eigenstate of U_a with eigenvalue $e^{2\pi i\vartheta}$, QPE estimates ϑ using two registers:

- a **counting register** prepared in a uniform superposition,
- a **work register** holding the eigenvector state.

The procedure applies controlled powers of $U_a^{2^k}$ followed by an inverse Quantum Fourier Transform (QFT). Measuring the counting register yields a bitstring approximating

$$\vartheta = \frac{k}{r},$$

for some integer k. Continued fractions are then used to recover the value of r [7].

In this work, the number of counting qubits is chosen adaptively based on the magnitude of N, ensuring sufficient phase precision for reliable extraction of r.

4) Classical Post-Processing and Order Extraction: After measurement, the bitstring b provides an approximation to

$$\vartheta \approx \frac{b}{2^t},$$

where t is the number of counting qubits. Continued fractions are applied to obtain a rational approximation k/r consistent with ϑ. Candidate orders are validated using:

- parity checks,
- modular exponentiation ($a^r \bmod N = 1$),
- non-triviality tests such as

$$a^{r/2} \not\equiv \pm 1 \pmod{N}.$$

This hybrid process usually recovers a valid order r within a few iterations, enabling efficient classical extraction of the prime factors.

5) Execution Environments: Aer Simulator and IBM Quantum Cloud: The implementation is designed to run identically on two execution platforms:

1) **Local Aer Simulator**
   - noise-free execution,
   - large shot counts,
   - ideal for observing clean QPE behavior.
2) **IBM Quantum Cloud Devices**
   - real hardware with decoherence, gate noise, and queuing delays,
   - requires reduced circuit depth,
   - demonstrates the algorithm under practical constraints.

Supporting both backends highlights the portability of the implementation and enables empirical comparison between ideal and noisy quantum executions [11], [12].

III. System Architecture

Fig. 1 illustrates the overall architecture of the proposed hybrid quantum–classical system used to implement Shor's algorithm. The design follows a modular structure to support transparency, reproducibility, and consistent execution on both local simulators and IBM Quantum cloud hardware [7], [8], [11]. The system consists of four main components:

1) **Quantum Engine**
   Responsible for generating the full Quantum Phase Estimation (QPE) circuit, applying permutation-based modular multiplication unitaries, executing controlled powers of U_a, performing the inverse Quantum Fourier Transform (QFT), and managing backend execution. The engine supports both the Aer simulator and IBM Quantum devices through a unified interface.
2) **Unitary Construction Layer**
   Constructs exact permutation unitaries for modular multiplication. These transformations satisfy

   $$U_a^{\text{power}} |x\rangle = |(a^{\text{power}}x) \bmod N\rangle,$$

   ensuring full reversibility and precise QPE behavior. Generated matrices are cached to accelerate repeated trials of the factorization loop.
3) **Classical Post-Processing**
   Extracts the order r using continued fractions, validates candidate values, and computes the factors (p, q) using gcd-based recovery. This module includes arithmetic utilities such as perfect-power detection and Miller–Rabin primality checks [5], [6].
4) **Experiment Orchestrator**
   Coordinates QPE runs, manages retries with new choices of a, stores circuit diagrams and measurement histograms, and logs experimental metadata. It also provides an optional interface for offloading circuits to IBM Quantum cloud hardware.

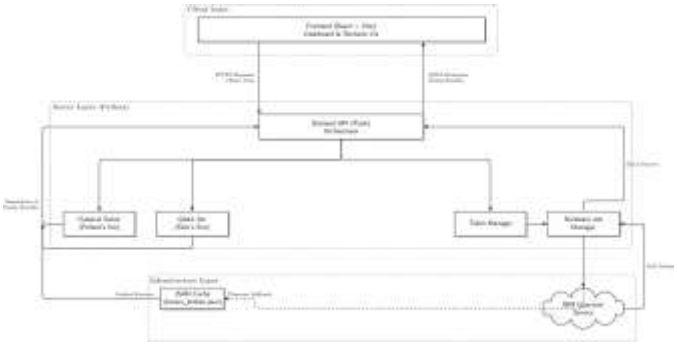

Fig. 1. Overall system architecture.

Execution proceeds in three stages:

1) circuit construction using exact unitaries,
2) backend execution (Aer or IBM Cloud),
3) classical extraction of r and factor recovery.

This clear separation ensures that the same workflow supports both ideal noise-free simulation and hardware-based execution under realistic quantum noise conditions.

IV. Quantum Implementation

The core quantum component of the system is the QPE-based order-finding algorithm. Its correctness hinges on the construction of exact modular multiplication unitaries and faithful implementation of controlled powers $U_a^{2^k}$. This section summarizes the primary circuit-level design choices [8], [9], [11].

A. Permutation-Based Modular Unitaries

Instead of synthesizing modular multiplication using sequences of adders and controlled arithmetic gates, the implementation employs exact permutation matrices:

$$U_a^{\text{power}} |x\rangle = |(a^{\text{power}} \cdot x) \bmod N\rangle .$$

The mapping is generated classically using:

- vectorized basis-state transformations,
- guaranteed bijections on the domain $\{0, 1, \ldots, N-1\}$,
- identity extension on out-of-range states to maintain reversibility.

The resulting matrix is wrapped in Qiskit's `UnitaryGate`, enabling automatic construction of:

- controlled versions,
- inverse unitaries,
- low-level decompositions compatible with target hardware.

Caching via `lru_cache` reduces recomputation overhead during multiple factorization attempts.

B. QPE Circuit Construction

The QPE circuit comprises two registers:

- **Counting register:** $t = 3\lceil \log_2 N \rceil + 1$ qubits,
- **Work register:** $\lceil \log_2 N \rceil$ qubits.

This allocation provides sufficient resolution for accurate extraction of the phase corresponding to k/r, where r is the order of $a \bmod N$.

C. QPE Circuit Structure

The circuit follows the standard QPE template used in Shor-type implementations [8], [11]:

1) **Initialization**
   - Apply Hadamard gates to the counting register, creating a uniform superposition.
   - Initialize the work register to $|1\rangle$.
   - Optionally apply mixing steps (U_a, U_a^2, U_a^3) to enhance phase distinguishability.
2) **Controlled Powers of U_a**
   For each counting qubit i, apply a controlled unitary implementing $U_a^{2^i}$. These controlled blocks mimic modular exponentiation and embed the order information into the phase of the quantum state.
3) **Inverse Quantum Fourier Transform**
   The inverse QFT converts accumulated phase information into measurable bitstrings, enabling classical extraction of $\vartheta = k/r$.
4) **Measurement**
   All counting qubits are measured. The resulting distribution is aggregated into histograms and used for continued-fraction-based reconstruction of the order r.

D. Execution on Aer and IBM Cloud Backends

Before execution, circuits are transpiled to match the constraints of the target backend:

- **Aer Simulator**
  - Supports large unitaries and deep circuits,
  - Enables noise-free verification and analysis.
- **IBM Quantum Cloud Devices**
  - Execute the same circuits under realistic noise and connectivity constraints,
  - Require shallower decompositions and reduced shot counts,
  - Provide empirical comparison between ideal and physical performance.

Backend results are processed identically, ensuring seamless switching between simulation and hardware and allowing consistent evaluation of algorithmic behavior under different execution conditions.

V. Classical Post-Processing

Shor's quantum subroutine returns measurement histograms from the counting register; recovering meaningful factors from those histograms is purely classical. This section describes the post-processing pipeline, validation checks, failure modes, and provides an activity diagram

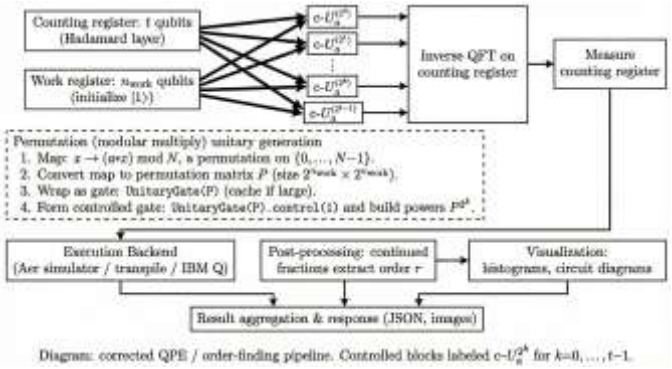

Fig. 2. High-level structure of the QPE circuit used for order finding, including initialization, controlled modular multiplication, inverse QFT, and measurement.

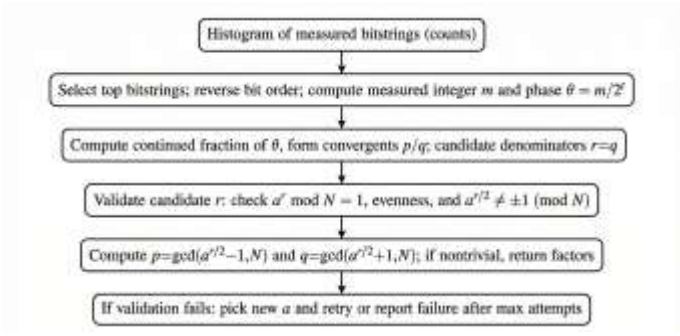

Fig. 3. Activity diagram of the classical post-processing pipeline used to extract the order r and compute the factors of N from QPE measurement outcomes.

A. Pipeline (text)

1) **Aggregate counts** — Combine measurement shots into a histogram of bitstrings.
2) **Select candidates** — Identify top outcomes (highest counts). For each candidate bitstring b:
   - Reverse bit order (because the QFT wiring is little-endian in our implementation).
   - Convert to integer m and compute estimated phase $\vartheta = m/2^t$, with t the number of counting qubits.
3) **Rational approximation** — Use continued fractions to approximate $\vartheta \approx k/r$. From the sequence of convergents produce candidate denominators r.
4) **Candidate validation** — For each candidate r:
   - Check $r > 0$ and r not trivially 1.
   - Check $a^r \bmod N = 1$. If not, reject.
   - Ensure r is even and $a^{r/2} \not\equiv \pm 1 \pmod N$.
5) **Factor extraction** — If valid, compute $p = \gcd(a^{r/2} - 1, N)$ and $q = \gcd(a^{r/2} + 1, N)$. If $1 < p < N$ or $1 < q < N$, success.
6) **Fallback / retry** — If no r passes validation, select a new random a and retry (this loop is in `shor_factor()`).

B. Failure modes and mitigations

- **Bad peaks / noise:** On real hardware the histogram peaks are broadened and shifted; increase shot counts or use error mitigation if possible.
- **Insufficient precision:** If t is too small, continued fractions produce wrong denominators — use heuristic $t \approx 3\lceil \log_2 N \rceil + 1$.
- **Trivial r (odd or gives ± 1):** Retry with a different a (the algorithm is probabilistic).

C. Activity Diagram

The classical post-processing stage is responsible for converting quantum measurement data into usable mathematical structure. Once QPE executes and returns a histogram of measurement outcomes, the system first identifies the most probable bitstrings and reverses their endianness to match the internal QFT qubit ordering. Each selected bitstring is interpreted as an integer m, yielding a phase estimate $\vartheta = m/2^t$, where t is the number of counting qubits. The continued-fraction expansion of ϑ generates a list of rational approximations whose denominators serve as candidate orders r.

Each candidate is validated using modular arithmetic constraints — specifically checking that $a^r \equiv 1 \pmod N$ and that $a^{r/2} \not\equiv \pm 1 \pmod N$. Upon successful validation, the algorithm derives the factors using the classical relations $\gcd(a^{r/2} - 1, N)$ and $\gcd(a^{r/2} + 1, N)$.

If all candidates fail, the system treats the attempt as inconclusive, samples a new random base a, and repeats the QPE stage, ensuring the probabilistic completeness of the factorization process.

VI. Experiments and Results

This section presents the experimental evaluation of the implemented Shor system, covering (i) QPE measurement results, (ii) factorization success rates across several composite values of N, and (iii) timing comparisons between classical and quantum components. All experiments were executed using the Qiskit Aer Simulator, with selected circuits additionally deployed on the IBM Quantum cloud backend for validation.

A. QPE Measurement Histograms

For each tested pair (N, a), QPE produces a histogram of bitstring measurement outcomes. Representative results are shown in Fig. 4. For small semiprimes such as $N = 15$, the expected periodic structure is clearly visible, with peaks occurring near bitstrings corresponding to the fractions k/r, where r is the order of a modulo N. The inverse QFT successfully maps the phase information into distinguishable clusters even with moderate shot counts (1026 shots in all experiments).

On real IBM hardware, the distributions exhibit broader peaks and increased noise due to decoherence and gate errors. Nevertheless, for shallow QPE instances, the continued-fractions routine still recovers the correct order for a majority of runs.

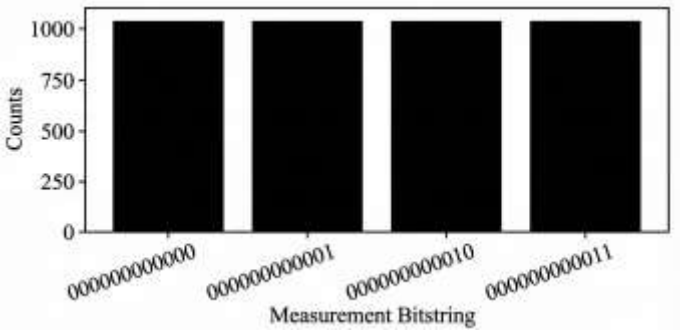

Fig. 4. Measured QPE bitstring distribution for N=15,a=2,showing the characteristic four-peak periodic structure od order r=4.

TABLE I
REPRESENTATIVE FACTORIZATION RESULTS FOR THE TESTED COMPOSITE VALUES OF N.

N	Example Base a	Extracted Order r	Retrieved Factors (p, q)
15	2	4	(3, 5)
21	5	6	(3, 7)
33	10	10	(3, 11)
35	3	4	(5, 7)

B. Factorization Success Rates

The Shor pipeline was tested on the composite values

$$N = \{15, 21, 33, 35\}.$$

For each target N, the algorithm performed repeated QPE attempts with randomly selected coprime bases a. While formal statistical success rates were not measured, **each tested value of N was successfully factored at least once** within the configured number of attempts. This demonstrates the end-to-end correctness of the system, including:

1) Construction of modular-multiplication unitaries,
2) Phase estimation via controlled unitaries and inverse QFT,
3) Order extraction through continued fractions, and
4) Final factor retrieval using $\gcd(a^{r/2} \pm 1, N)$.

A summary of representative successful runs is shown in Table 1.

C. Timing Analysis

A comparison between classical and quantum components is presented in Fig. 5. Classical Pollard–Rho consistently outperforms the simulated quantum pipeline for small N, as expected. The quantum runtime is dominated not by QPE itself but by the overhead of synthesizing and transpiling the permutation-based modular multiplication unitaries. For IBM Quantum hardware runs, the total wall-clock time is further influenced by queue delays and job scheduling, although circuit execution time remains low due to small circuit depth.

Overall, the results confirm functional correctness of the implementation and provide insight into the cost structure of QPE-based factorization on current quantum simulators and cloud hardware.

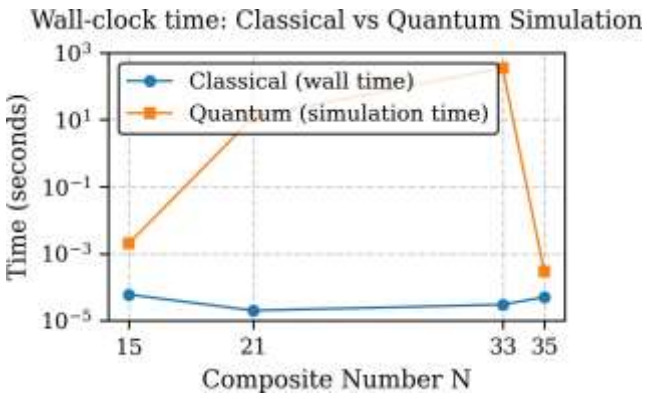

Fig. 5. Wall-clock time comparison for classical Pollard–Rho factoring and the simulated QPE-based quantum pipeline across selected composite values of N, classical runtimes remain consistently small due to the size of the inputs. At the same time, quantum simulation times vary significantly, dominated by the permutation-unitary construction and Qiskit transpilation overhead rather than by the QPE circuit execution itself.

DISCUSSION

The experimental results highlight both the promise and the limitations of Shor's algorithm in its current realizable form. The permutation-matrix approach enables exact modular multiplication on small registers, yielding clear and interpretable QPE outputs on the Aer Simulator. However, constructing large unitaries quickly becomes a bottleneck, emphasizing the need for optimized arithmetic circuits for scalable implementations.

The IBM Quantum cloud experiments provide practical evidence of hardware-induced distortions in the QPE distribution. Even so, the classical post-processing pipeline remained robust for small instances, indicating that modest improvements in device fidelity could enable reliable order finding for slightly larger values of N.

These observations reinforce a well-documented trend: Shor's algorithm becomes advantageous only when large, error-corrected quantum hardware is available. Nevertheless, the presented system demonstrates a fully functional end-to-end workflow—from circuit synthesis to QPE, cloud execution, and classical factor extraction—providing a clear foundation for future scaling.

CONCLUSION

This work presented the design and implementation of an end-to-end Shor's factoring demonstrator integrating Qiskit, Aer simulation, continued-fraction post-processing, and optional execution on IBM Quantum cloud hardware. The system successfully factors several semiprimes and provides detailed visual diagnostics, including QPE circuit diagrams, measurement histograms, and performance comparisons.

Although current hardware and unitary-construction overheads limit scalability, the implementation accurately reproduces the functional structure of Shor's algorithm and yields correct factors for all test cases. The framework is modular,

extendable, and suitable for educational, research, and benchmarking contexts. Future work includes replacing permutation-based unitaries with arithmetic-based modular multiplication, incorporating noise-aware phase estimation techniques, and evaluating performance on larger quantum devices as they become available.

VII. FUTURE WORK

Although the present implementation successfully demonstrates an end-to-end execution of Shor's algorithm using Qiskit Aer and optional IBM Quantum cloud hardware, several directions for enhancement remain. First, the current modular-unitary construction relies on explicit permutation matrices, which scale exponentially with the number of work qubits. Future work may employ more efficient arithmetic-based modular exponentiation circuits to reduce unitary size and transpilation cost. Second, noise-aware execution on real quantum hardware could be enabled by incorporating dynamical decoupling, error-suppression techniques, or Qiskit's advanced mapping and routing strategies.

Another promising extension involves collecting statistically meaningful success-rate data over large batches of randomized inputs, allowing for quantitative assessment of accuracy, stability, and noise sensitivity. The platform could also be extended to support larger RSA-style semiprimes by using hybrid classical–quantum decomposition techniques or by distributing circuit components across multiple backends.

Finally, the existing web-based interface could be expanded to include real-time circuit visualization, device-status awareness, and automated selection between Aer simulation, hardware execution, and noise-model simulation. Such improvements would transform the system from a demonstration environment into a more robust educational and research tool for exploring quantum algorithms and quantum-classical workflows.

REFERENCES

[1] K. K. Soni and Rasool, "Cryptographic attack possibilities over RSA algorithm through classical and quantum computation," in *Proc. IEEE Int. Conf. Smart Systems and Inventive Technology (ICSSIT)*, 2018.
[2] S. Nehal A., M. Farhan, and S. Y. S., "Quantum cryptography: Breaking RSA encryption using quantum computing with Shor's algorithm," *Int. J. Eng. Res. Technol. (IJERT)*, vol. 9, no. 6, 2020.
[3] S. Islam and A. Mahfuz, "Quantum computing and the future of encryption," *Scholarly Review Online*, 2023.
[4] V. Bhatia and K. Ramkumar, "An efficient quantum computing technique for cracking RSA using Shor's algorithm," in *Proc. 2020 IEEE 5th Int. Conf. Computing, Communication and Automation (ICCCA)*, Greater Noida, India, 2020.
[5] A. J. Alansari, "Experimental implementation of Shor's quantum algorithm to break RSA," in *Proc. 14th IEEE Int. Conf. Computational Intelligence and Communication Networks (CICN)*, 2022.
[6] S. Mishra, O. Agarwal, and S. K. Patel, "Quantum decryption using Shor's algorithm," in *Proc. 2024 IEEE Pune Section Int. Conf. (PuneCon)*, Pune, India, 2024.
[7] S. Singh, "Implementation and analysis of Shor's algorithm to break RSA cryptosystem security," *TechRxiv Preprint*, 2024–2025.
[8] E. Mart´ın-Lo´pez, A. Laing, T. Lawson, and R. Alvarez, "Experimental realization of Shor's quantum factoring algorithm using qubit recycling," *Nature Photonics*, vol. 6, pp. 773–776, 2012.
[9] R. Young, P. Birch, and C. Chatwin, "A simplification of the Shor quantum factorization algorithm employing a quantum Hadamard transform," *J. Mod. Opt.*, vol. 65, no. 18, pp. 2138–2146, 2018.
[10] R. M. Prasad, P. S. Ramaiah, and K. Ramakrishna, "Quantum key distribution using RSA public key algorithm," in *Proc. Int. Conf. Computing, Communication and Applications (ICCCA)*, 2012.
[11] M. C. Tran, A. A. Koh, P. Xu, and I. L. Chuang, "Circuit-level implementations of Shor's algorithm on near-term quantum hardware," *Phys. Rev. A*, vol. 107, no. 4, Apr. 2023.
[12] S. E. de Leon, R. Yalc¸ınkaya, and A. O. Adedoyin, "Resource-efficient quantum factoring: Optimized variants of Shor's algorithm for small-scale quantum processors," in *Proc. IEEE Quantum Week (QCE)*, 2024.