TECHNICAL SEMINAR | SRINIVAS RAO TAMMIREDDY | 217Y1A05C0 | DR. S. PRATAP SINGH

Advantages of Quantum Computing in Factor Attacks on RSA Encryption.

Abstract

The RSA cryptosystem is a widely used public-key encryption method that relies on the difficulty of factoring large integers for security. Traditional computers struggle with this task, making RSA a robust encryption standard. However, the emergence of quantum computers poses a significant threat, as they can efficiently factor large numbers using Shor’s algorithm. This presentation explores how quantum computing compromises RSA encryption and examines its implications for the future of cryptography.

Aim

- To explain how quantum computers break RSA encryption.
- To highlight the advantages of quantum computing over classical computing.
- To discuss the future of cryptography in a post-quantum world.

Contents

RSA INTRODUCTION

Introduction to RSA

Brief description of RSA Encryption and Decryption.

RSA ALGORITHM

What is RSA?

- RSA (Rivest-Shamir-Adleman) is a public-key cryptosystem used for secure communication.
- It is based on the mathematical difficulty of factoring large prime numbers.

RSA ALGORITHM

Key Components

- **Public Key `(e, N)`:** Used for encryption.
- **Private Key `(d, N)`:** Used for decryption.
- **Modular Arithmetic:** Security depends on the infeasibility of computing the modular inverse of large numbers.

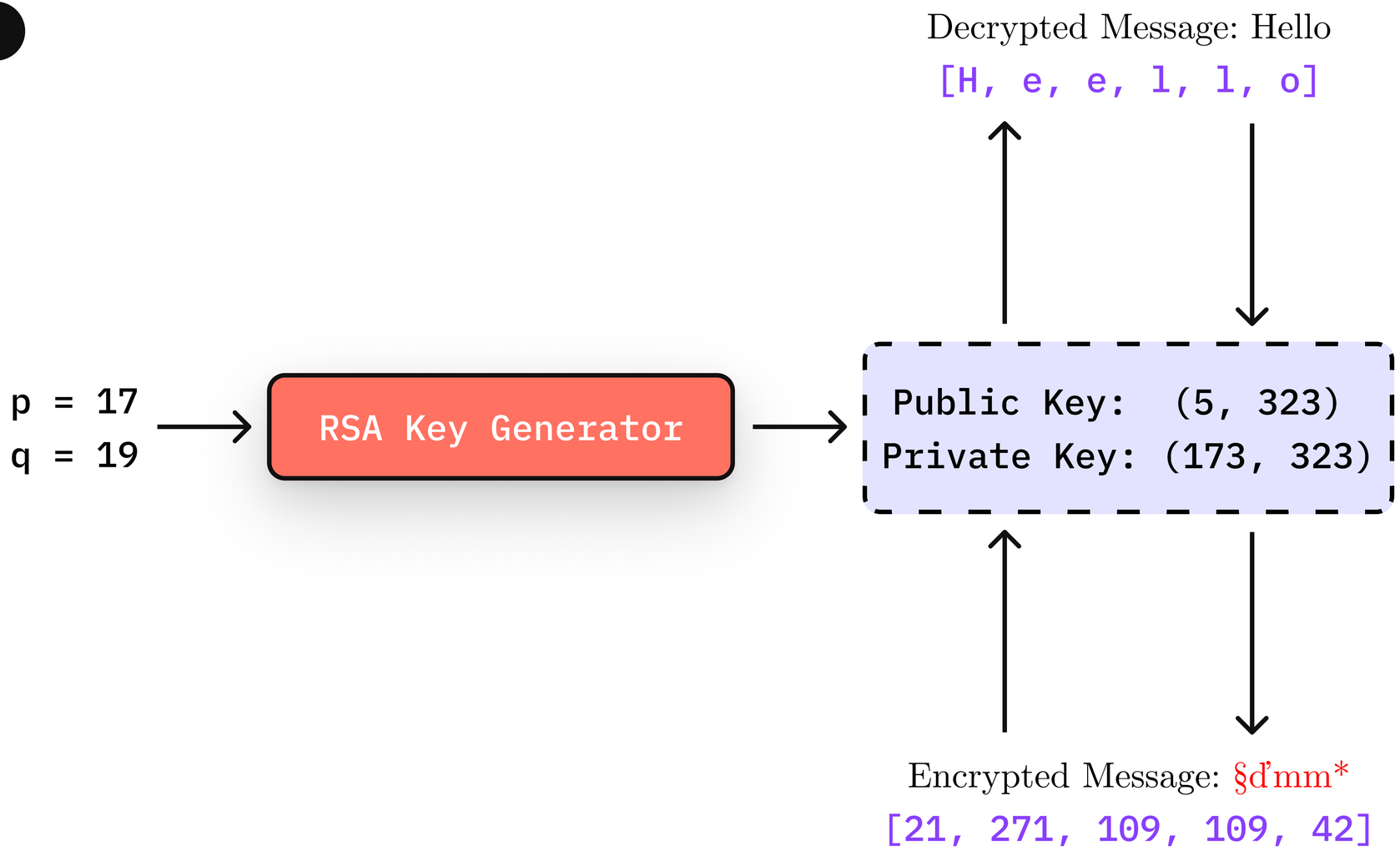
RSA ALGORITHM

RSA Algorithm

Algorithm 1 RSA Encryption Algorithm

Key Generation:
Choose two large prime numbers p and q.
Compute $N = p \times q$.
Compute Euler's totient function $\phi(N) = (p-1) \times (q-1)$.
Choose an integer e such that $1 < e < \phi(N)$ and $gcd(e, \phi(N)) = 1$.
Compute the private key d as the modular inverse of e modulo $\phi(N)$, i.e., $d \equiv e^{-1} \mod \phi(N)$.
Public Key: (e, N); Private Key: (d, N).
Encryption:
Convert plaintext message M to an integer m such that $0 < m < N$.
Compute ciphertext C using $C \equiv m^e \mod N$.
Decryption:
Recover plaintext m using $m \equiv C^d \mod N$.
Convert m back to the original message M.

RSA ALGORITHM



A simple Demonstration of RSA

RSA ALGORITHM

RSA Implementation

```python
# Step 1: Key Generation
import random
from sympy import gcd, isprime, mod_inverse

def generate_rsa_keypair(p, q):
    # Check if both numbers are prime
    if not (isprime(p) and isprime(q)):
        raise ValueError('Both numbers must be prime.')
    elif p == q:
        raise ValueError('p and q cannot be the same')

    # Compute n (modulus) and Euler's Totient (phi_n)
    n = p * q
    phi_n = (p - 1) * (q - 1)

    # Choose e such that 1 < e < phi_n and gcd(e, phi_n) = 1
    e = 5  # Example choice, typically chosen as a small prime number

    g = gcd(e, phi_n)
    while g != 1:
        e = random.randrange(2, phi_n - 1)
        g = gcd(e, phi_n)

    # Compute d such that (d * e) % phi_n = 1
    d = mod_inverse(e, phi_n)

    # Return public key (e, n) and private key (d, n)
    return ((e, n), (d, n))
```

RSA ALGORITHM

RSA Implementation

```
# Example usage
p, q = 17, 19 # Generally Large primes

# Generate public key (e, n) and private key (d, n)
public_key, private_key = generate_rsa_keypair(p, q)

# Step 2: Encryption
def encrypt(message, public_key):
    e, n = public_key
    # Encrypt the message using the public key
    return pow(message, e, n)

# Step 3: Decryption
def decrypt(ciphertext, private_key):
    d, n = private_key
    # Decrypt the ciphertext using the private key
    return pow(ciphertext, d, n)

# Example usage
message = 12  # Example message
ciphertext = encrypt(message, public_key)
decrypted_message = decrypt(ciphertext, private_key)
```

FACTORING ATTACK

The Factoring Attack

Attempts to Break RSA Security

FACTORING ATTACK

Involves factoring the modulus n to obtain the prime factors p and q, allowing computation of the private key d.

Step 1:

Set a Random number g

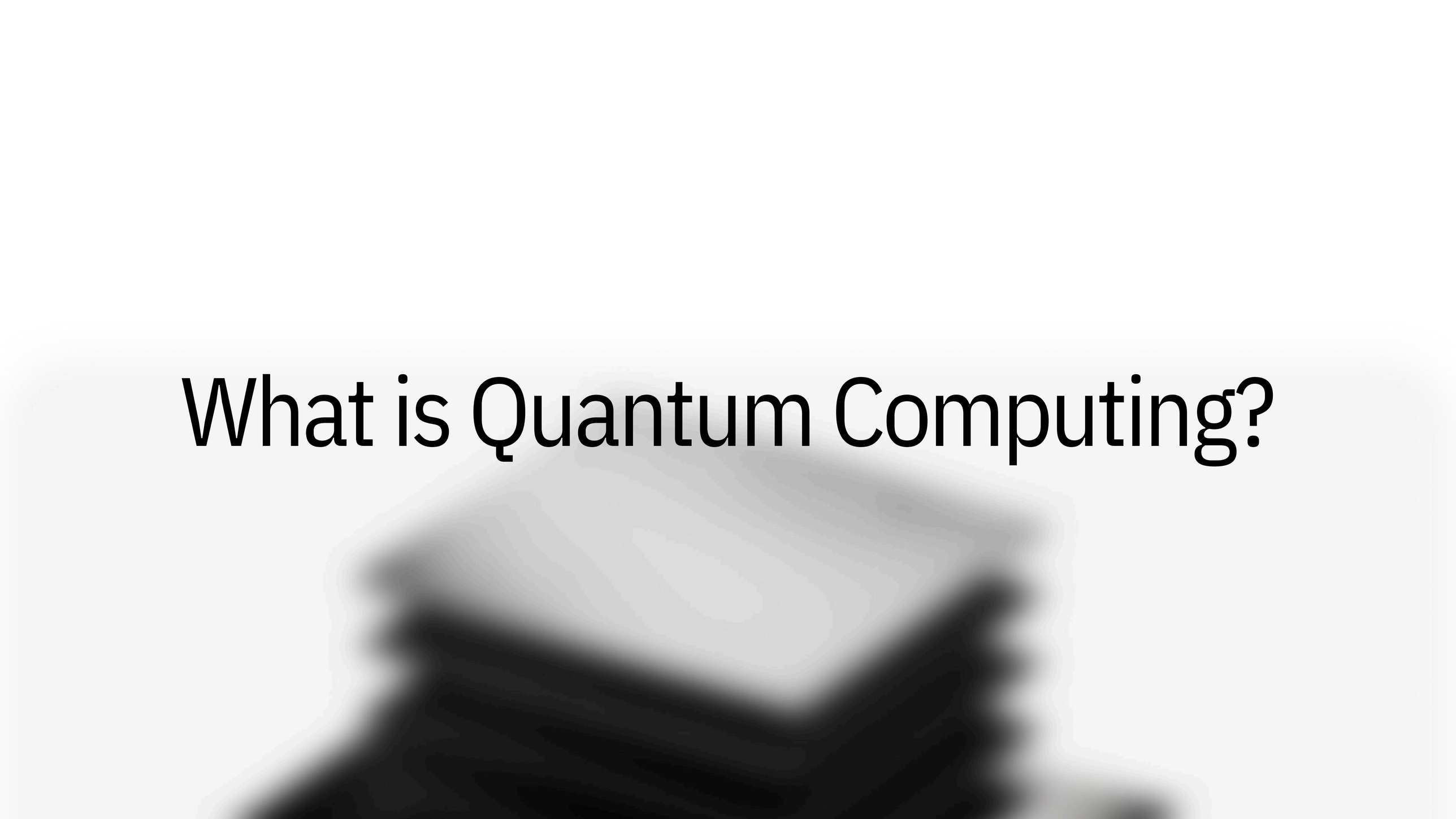
Step 2:

Compute **`r :`** g^r = mN + 1

Step 3:

Now (g^(r/2) +/-1) have the factors

What is Quantum Computing?



QUANTUM COMPUTING

Key Quantum Concepts

- Qubits: Fundamental unit of quantum computing
- Superposition: A qubit can be both 0 and 1 simultaneously
- Entanglement: Two qubits can be linked regardless of distance
- Quantum Gates: Operate on qubits like classical logic gates

QUANTUM COMPUTING

How Quantum Computing Works?

- Qubits represent multiple states at once (Superposition)
- Entangled qubits process information faster
- Probability-based computing instead of fixed logic

QUANTUM COMPUTING

How Quantum Computing Works?

- Qubits represent multiple states at once (Superposition)
- Entangled qubits process information faster
- Probability-based computing instead of fixed logic

QUANTUM COMPUTING

Some Quantum Algorithms

- Shor’s Algorithm – Breaks RSA encryption by fast factorization
- Grover’s Algorithm – Speeds up unstructured searches
- Quantum Simulation – Models complex molecules for drug discovery

FACTORING ATTACK

Feature	Classical Computer	Quantum Computer
Data Unit	Bits (0 or 1)	Qubits (0 & 1 at the same time)
Processing	Sequential	Parallel (Superposition)
Communication	Independent Bits	Entangled Qubits
Speed	Slower for complex problems	Exponential Speedup

SHOR’S ALGORITHM

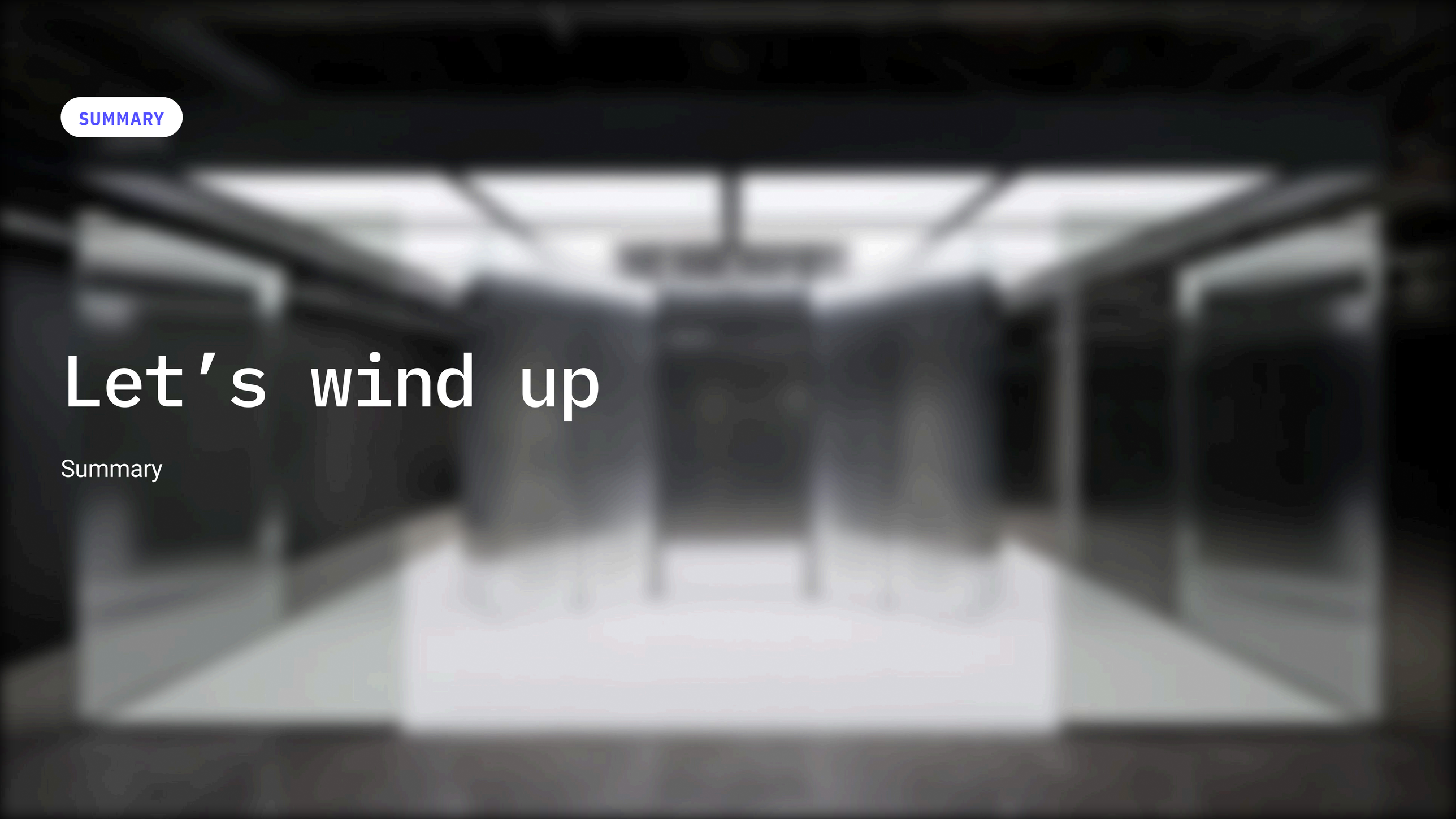
The Shor’s Algorithm

The fast Factoring method

SHOR'S ALGORITHM

Will See in part -II

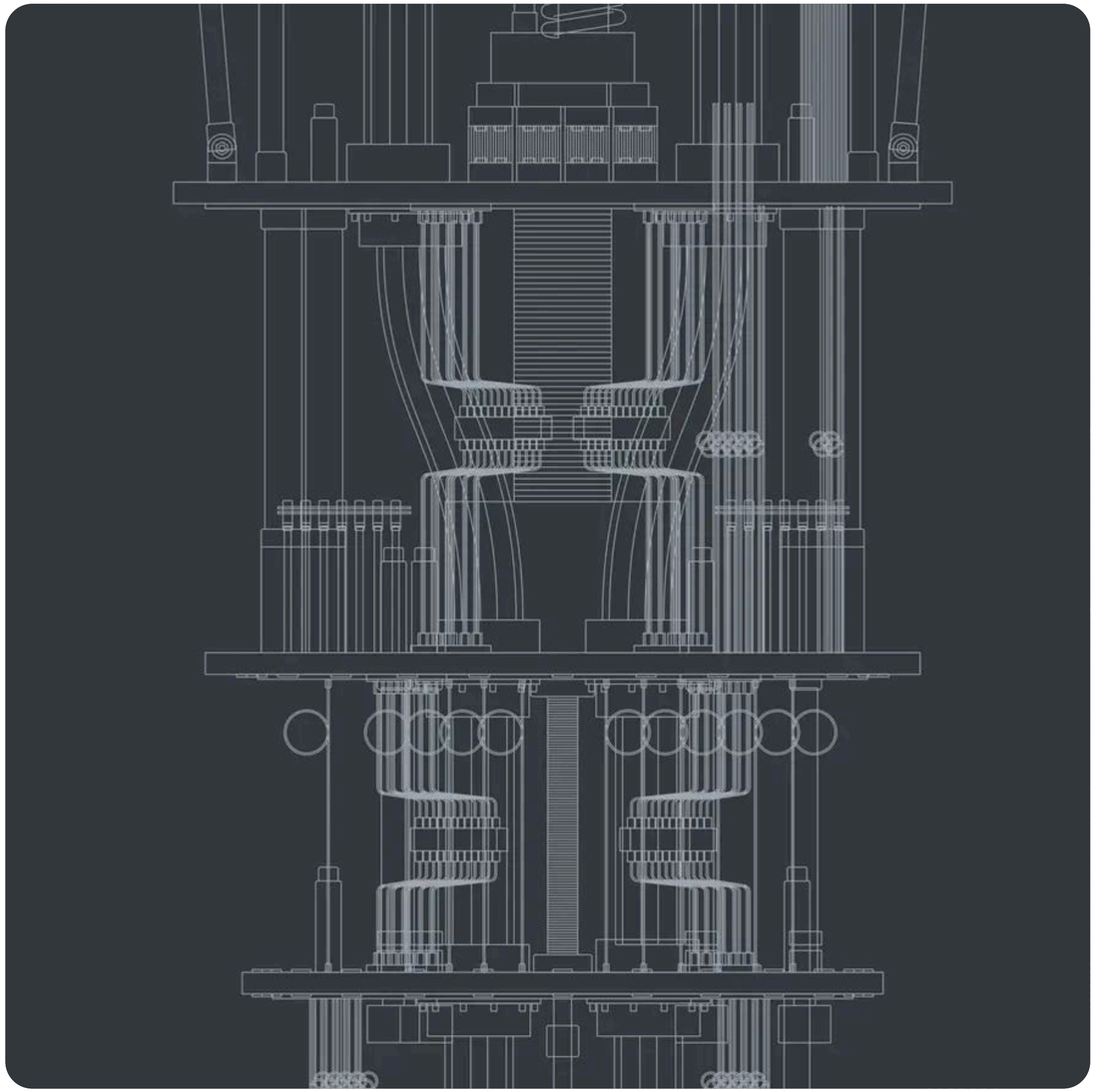
We shall see bye



SUMMARY

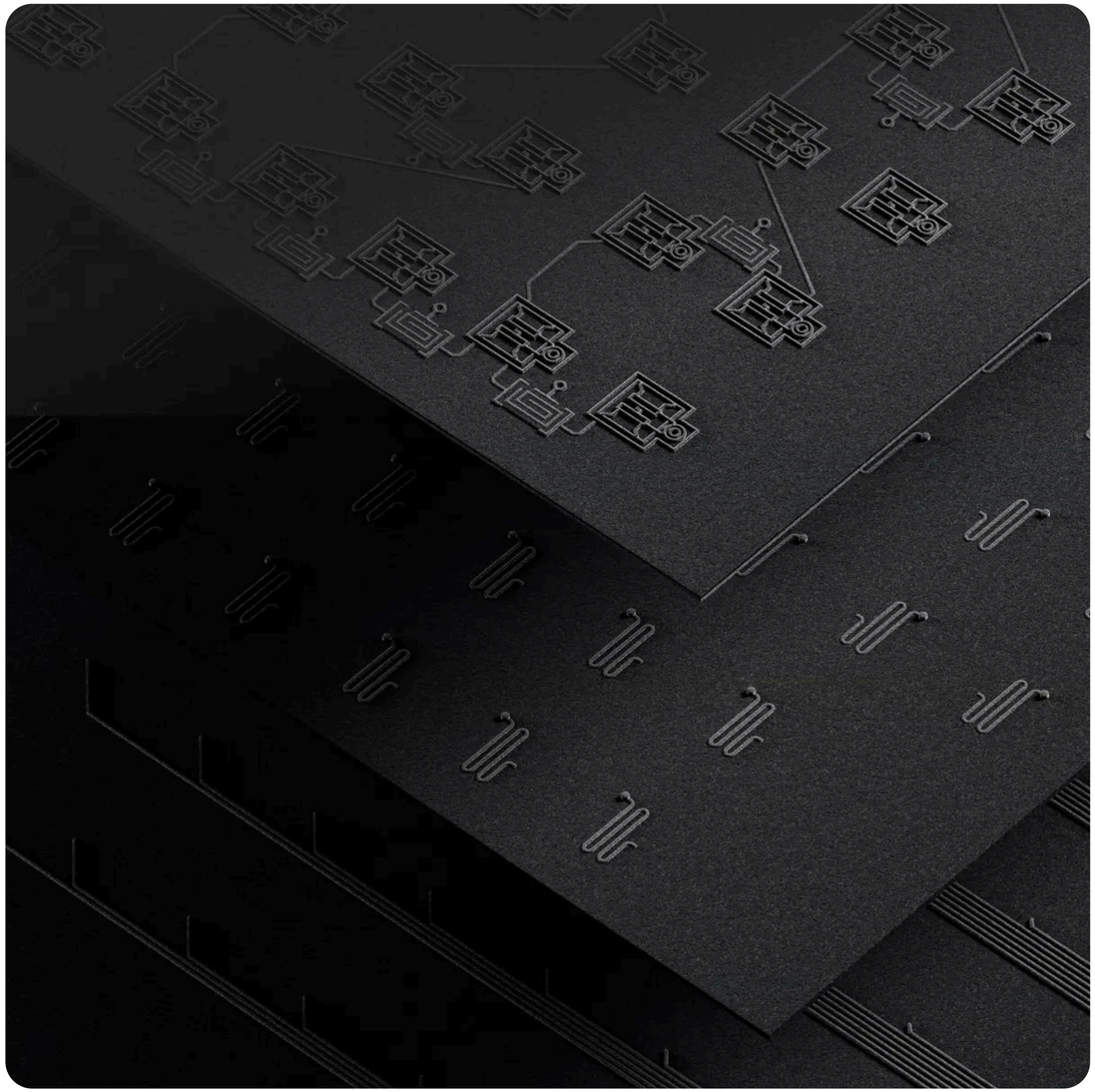
Let’s wind up

Summary



Challenges in Quantum Computing

- Decoherence: Qubits lose information quickly
- Error Correction: Requires redundancy & high precision
- Scalability: Building large-scale quantum computers
- Cost & Hardware Limitations: Expensive & delicate technology



Conclusion

- Quantum Computing is revolutionary but still in development
- Holds immense potential in multiple industries
- Challenges remain, but rapid progress is being made

QUNATUM COMPUTING

Questions?

Open floor for questions and discussion