תמונה שמכילה גופן, גרפיקה, טקסט, לוגו

התיאור נוצר באופן אוטומטי

Software Engineering Department  
ORT Braude College

Capstone Project Phase A – 61998

**Quantum Secure Socket Layer (QSSL)**

**24-2-R-17**



**Submitters:**

**Raz Tibi – Raz.Tibi@e.braude.ac.il  
Matan Czuckerman - Matan.Czuckermann@e.braude.ac.il**

**Supervisor:**

**Ronen Zilber**

Table Of Content

[1 Abstract 4](#_Toc189501694)

[2 Introduction 4](#_Toc189501695)

[3 Literature review 6](#_Toc189501696)

[3.1 Useful terms 6](#_Toc189501697)

[3.1.1 Learning With Errors 6](#_Toc189501698)

[3.1.2 Lattice 6](#_Toc189501699)

[3.1.3 Security terms 7](#_Toc189501700)

[3.1.4 Digital signature terms 7](#_Toc189501701)

[3.2 CRYSTALS project 7](#_Toc189501702)

[3.2.1 Module Lattices 8](#_Toc189501703)

[3.2.2 CRYSTAL – Kyber 8](#_Toc189501704)

[3.2.3 CRYSTAL – Dilithium 10](#_Toc189501705)

[3.3 Falcon 10](#_Toc189501706)

[3.3.1 Algorithm Highlights 11](#_Toc189501707)

[3.3.2 NTRU 11](#_Toc189501708)

[3.4 SPHINCS + 11](#_Toc189501709)

[4 Background 12](#_Toc189501710)

[4.1 Introduction to Quantum Computers 12](#_Toc189501711)

[4.1.1 Qubits (Quantum Bits) 12](#_Toc189501712)

[4.1.2 Superposition 13](#_Toc189501713)

[4.1.3 Entanglement 13](#_Toc189501714)

[4.1.4 Key Aspects of Quantum Interference 14](#_Toc189501715)

[4.1.5 Quantum Gates and Circuits 14](#_Toc189501716)

[4.1.6 Quantum Error Correction 15](#_Toc189501717)

[4.1.7 Quantum Supremacy 15](#_Toc189501718)

[4.1.8 Challenges in Quantum Computing 15](#_Toc189501719)

[4.2 Cryptography 16](#_Toc189501720)

[4.2.1 Asymmetric Cryptography 17](#_Toc189501721)

[4.2.2 RSA 17](#_Toc189501722)

[4.2.3 Elliptic Curve Cryptography 18](#_Toc189501723)

[4.2.4 Diffie Helman 19](#_Toc189501724)

[4.2.5 SSL 20](#_Toc189501725)

[4.2.6 TLS 20](#_Toc189501726)

[4.2.7 Post Quantum Cryptography 23](#_Toc189501727)

[5 Shor’s Algorithm 24](#_Toc189501728)

[5.1 Feasibility and Impact 25](#_Toc189501729)

[5.2 The Procedure 25](#_Toc189501730)

[5.2.1 The Goal 25](#_Toc189501731)

[5.2.2 Preliminary Steps 25](#_Toc189501732)

[5.2.3 The Quantum Algorithm 25](#_Toc189501733)

[5.3 Quantum Advantage 26](#_Toc189501734)

[6 Related Work 28](#_Toc189501735)

[6.1 Open Quantum Safe Project 28](#_Toc189501736)

[6.2 Liboqs 28](#_Toc189501737)

[6.3 Quantum safe TLS 28](#_Toc189501738)

[7 Expected Achievements 29](#_Toc189501739)

[8 QSSL 29](#_Toc189501740)

[8.1 Product Diagram and GUI 30](#_Toc189501741)

[8.1.1 Overview 30](#_Toc189501742)

[8.1.2 QSSL Protocol 34](#_Toc189501743)

[9 References 36](#_Toc189501744)

# Abstract

The imminent rise of quantum computing presents a critical threat to current cryptographic systems, particularly those underpinning Secure Socket Layer (SSL), Transport Layer Security (TLS), and other protocols relying on RSA-based encryption and Diffie-Hellman key exchange. This article addresses the severe vulnerability of these widely deployed security protocols to quantum attacks. Specifically, Shor's quantum algorithm designed for integer factorization and discrete logarithm problems, poses a significant risk by rendering RSA and Diffie-Hellman protocols insecure. As quantum computers advance, the integrity of global digital communications is at risk, demanding urgent action to develop quantum-resistant encryption methods.

We introduce Quantum (Safe) SSL (QSSL), a novel approach to secure communication protocols designed to withstand attacks from both classical and quantum computers. QSSL incorporates post-quantum cryptographic algorithms that are currently believed to be resistant to quantum computational methods, including Shor's algorithm. This research outlines the theoretical framework of QSSL, discusses its implementation challenges, and evaluates its potential to safeguard digital communications in the post-quantum era.

By proactively addressing the quantum threat to SSL, TLS, RSA-based protocols, and Diffie-Hellman key exchanges, QSSL aims to ensure the continued security and privacy of internet communications, financial transactions, and sensitive data exchanges in a world where quantum computers become a reality. This work contributes to the ongoing efforts in cryptography and cybersecurity to prepare for the quantum computing revolution and its implications for global information security.

# Introduction

In an era where digital communication underpins nearly every aspect of modern society, the security of our information systems stands as a critical priority. For decades, we have relied on cryptographic algorithms that have effectively safeguarded our digital interactions. These pillars of cryptography - RSA (Rivest-Shamir-Adleman), Diffie-Hellman (DH) key exchange, and Elliptic Curve Cryptography (ECC) - have formed the foundation of secure communications, e-commerce, and data protection.

However, we stand at the threshold of a technological revolution that threatens to undermine these cornerstones of digital security. The advent of quantum computing, while promising unprecedented computational power for solving complex problems, also poses an existential threat to our current cryptographic paradigms. This book delves into the heart of this impending crisis and charts a course toward a secure, quantum-resistant future.

The vulnerability of our current cryptographic systems lies in their mathematical foundations. RSA's security is based on the difficulty of factoring large numbers, while DH and ECC rely on the discrete logarithm problem. These problems, while computationally infeasible for classical computers, fall prey to quantum algorithms, most notably Shor's algorithm. Developed by Peter Shor in 1994, this quantum algorithm can efficiently solve both the factoring and discrete logarithm problems, effectively breaking RSA, DH, and ECC in polynomial time on a sufficiently powerful quantum computer.

The implications of this vulnerability are profound. Our entire digital infrastructure - from secure web browsing and email to financial transactions and government communications - could be compromised once large-scale quantum computers become a reality. While the timeline for the development of such computers remains uncertain, the potential consequences are too severe to ignore. We must act now to develop and implement quantum-resistant cryptographic solutions.

This book presents a comprehensive exploration of the transition to quantum-safe cryptography. We will examine in detail the workings of Shor's algorithm, understanding precisely how it threatens our current systems. We will then turn our attention to the promising field of post-quantum cryptography, with a particular focus on lattice-based algorithms. These mathematical structures offer hope for creating cryptographic systems that resist both classical and quantum attacks.

Central to our discussion will be the practical implementation of quantum-safe cryptography. We will explore the efforts of the OpenSSL project, a widely-used open-source cryptographic library, in integrating post-quantum algorithms. Additionally, we will delve into the work of the Open Quantum Safe (OQS) project, which aims to develop and prototype quantum-resistant cryptographic algorithms.

The culmination of this book will be the presentation of a new protocol: Quantum-(Safe) Secure Sockets Layer (QSSL). This protocol builds upon the current Transport Layer Security (TLS) standard, integrating quantum-resistant algorithms to create a communication protocol designed to withstand the cryptanalytic capabilities of quantum computers. QSSL represents not just a theoretical construct, but a practical step toward securing our digital future.

As we embark on this exploration of quantum-safe cryptography, we must recognize that we are at a critical juncture in the history of information security. The decisions and developments made in the coming years will shape the landscape of digital security for decades to come. We hope that this book will serve not only as a guide to understanding the challenges we face, but also as a roadmap for navigating the transition to a quantum-safe world.

The journey ahead is complex and challenging, but it is one we must undertake. The security of our digital future depends on our ability to adapt and innovate in the face of quantum computing's disruptive potential. Let us begin this crucial exploration into the next frontier of cryptography - a world where our most sensitive information remains secure, even in the face of unprecedented computational power.

# Literature review

**NIST** – National Institute of Standards and Technology has initiated a process to solicit, evaluate, and standardize one or more quantum-resistant public-key cryptographic algorithms. Currently, public-key cryptographic algorithms are vulnerable to attacks from large-scale quantum computers.

It is intended that the new public-key cryptography standards will specify one or more additional unclassified, publicly disclosed digital signature, public-key encryption, and key-establishment algorithms that are available worldwide, and are capable of protecting sensitive government information well into the foreseeable future, including after the advent of quantum computers.

NIST announced its selection of 4 algorithms for standardization: the key encapsulation mechanism CRYSTALS-Kyber, and three signature schemes CRYSTALS-Dilithium, Falcon, and SPHINCS+.

## Useful terms

### Learning With Errors

learning with errors (**LWE**) is a mathematical problem that is widely used to create secure encryption algorithms. It is based on the idea of representing secret information as a set of equations with errors. In other words, LWE is a way to hide the value of a secret by introducing noise to it. In more technical terms, it refers to the computational problem of inferring a linear -ary function over a finite ring from given samples some of which may be erroneous. The LWE problem is conjectured to be hard to solve, and thus to be useful in cryptography.

### Lattice

In geometry and group theory, a lattice in the real coordinate space

is an infinite set of points in this space with the properties that coordinate-wise addition or subtraction of two points in the lattice produces another lattice point, that the lattice points are all separated by some minimum distance, and that every point in the space is within some maximum distance of a lattice point.  
Lattice-based cryptography is the generic term for constructions of cryptographic primitives that involve lattices, either in the construction itself or in the security proof. Lattice-based constructions support important standards of post-quantum cryptography. Unlike more widely used and known public-key schemes such as the RSA, Diffie-Hellman or elliptic-curve cryptosystems — which could, theoretically, be defeated using Shor's algorithm on a quantum computer — some lattice-based constructions appear to be resistant to attack by both classical and quantum computers. Furthermore, many lattice-based constructions are considered to be secure under the assumption that certain well-studied computational lattice problems cannot be solved efficiently.

### Security terms

Security in terms of indistinguishability has many definitions, depending on assumptions made about the capabilities of the attacker. It is normally presented as a game, where the cryptosystem is considered secure if no adversary can win the game with significantly greater probability than an adversary who must guess randomly.   
The most common definitions used in cryptography are:

* **IND-CPA -** indistinguishability under chosen plaintext attack
* **IND-CCA1 -** indistinguishability under (non-adaptive) chosen ciphertext attack**.**
* **IND-CCA2 -** indistinguishability under adaptive chosen ciphertext attack**.**

Security under either of the latter definitions implies security under the previous ones: a scheme which is **IND-CCA1** secure is also **IND-CPA** secure, and a scheme which is **IND-CCA2** secure is both **IND-CCA1** and **IND-CPA** secure. Thus, **IND-CCA2** is the strongest of the three definitions of security.

### Digital signature terms

There are two common formal definitions for the security of a digital signature scheme. Each of these definitions is presented as a “game”, or an experiment that is run between an attacker and some honest challenger.

* **EUF-CMA** - Existential Unforgeability under Chosen Message Attack
* **SUF-CMA -** Strong Existential Unforgeability under Chosen Message Attack

The main difference here is that this stronger definition ensures that the attacker cannot “maul” the signature. For example, a scheme where the attacker can re-randomize a valid signature so that it’s still valid, but looks different than the original value, would not satisfy **SUF-CMA**.

This might seem like a silly requirement, but it matters in some protocols. For example, this sort of malleability is possible in the ECDSA signature scheme, and has led to many problems in **Bitcoin**.

## CRYSTALS project

The "Cryptographic Suite for Algebraic Lattices" (**CRYSTALS**) encompasses two cryptographic primitives: **Kyber**, an **IND-CCA2**-secure key-encapsulation mechanism (KEM); and **Dilithium**, a strongly **EUF-CMA**-secure digital signature algorithm. Both algorithms are based on hard problems over module lattices, are designed to withstand attacks by large quantum computers, and have been submitted to the NIST post-quantum cryptography project.

### Module Lattices

Module lattices can be thought of as lattices that lie between the ones used in the definitions of the LWE problem, and those used for the Ring-LWE problem. If the ring underlying the module has a sufficiently high degree (like 256), then these lattices inherit all the efficiency of the ones used in the Ring-LWE problem, and additionally have the following advantages, when used in our cryptographic algorithms:

* The only operations required for **Kyber** and **Dilithium** for all security levels are variants of Keccak, additions/multiplications in for a fixed q, and the NTT (number theoretic transform) for the ring .
* This means that increasing/decreasing the security level involves virtually no re-implementation of the schemes in software or hardware. Changing a few parameters is all that one needs to convert an optimized implementation for one security level into an optimized implementation for a different one.
* The lattices used in **Kyber** and **Dilithium** have less algebraic structure than those used for Ring-LWE and are closer to the unstructured lattices used in LWE. It is therefore conceivable that if algebraic attacks against Ring-LWE appear (there are none that we are aware of at this point), then they may be less effective against schemes like Kyber and Dilithium.

### CRYSTAL – Kyber

The provided table – Figure 1 presents the values of n, k, and q as per the Kyber specification. The private key of **Kyber** uses **k** number of polynomials which have a degree of **n** (called s). This is generated using random small coefficients.

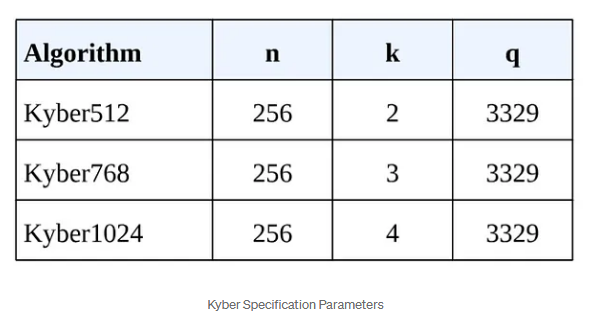
 A **Kyber** public key consists of two elements. A matrix of random polynomials **A** (k × k) and a vector of polynomials **t**. Matrix A is generated using coefficients (< **q**). To calculate vector **t**, an additional error vector **e** is required. This is also generated using random small coefficients. Then we can calculate **t=A×s+e.**   
Note that all operations are under the polynomial ring   
we can keep the secret **s** safe and broadcasts his public key **(A, t)** to everyone.

Figure 1

#### Encryption

To encrypt the message **m**, we need to convert it into a binary polynomial. Then we need to multiply it by **⌊q/2​⌉** (integer closest to q/2). For example we take m to be 11 (binary 1101)

We need 3 random small polynomials **r**,**e₁**,**e₂**.

Then we encrypt the value **m** using the public key (**A**,**t**). The encryption procedure calculates two values **u**and**v.**

#### Decryption

We can use the secret polynomial **s** retrieve the secret **m**from ciphertext. Note that m is still noisy.

Figure 2

The noise can be removed by comparing the received value to the closest valid message. In this case, by checking if closer to ⌊*q*/*2*​⌉= 0 or ⌊*q*/*2*​⌉. Then we get the rounded polynomial and then diving by ⌊*q*/*2*​⌉ will give **m**.

Now both parities has shared secret **m**which they can use for asymmetric encryption.

### CRYSTAL – Dilithium

The Generation algorithm in Dilithium creates a matrix **A.** Each entry of this matrix is a polynomial in the defined ring. The generation algorithm also creates random private vectors **s1** and **s2**, whose components are elements of **R** the polynomial ring. The public key is the matrix A and . It is infeasible for a quantum computer to know the secret values given just **t** and **A**. This problem is called Module-Learning With Errors (MLWE) problem, and it is a variant of LWE.

#### Identification scheme

1. The prover wants to prove they know the private key. They generate a random secret nonce **y** whose coefficient is less than a security parameter. They then compute **Ay** and set a commitment **w1** to be the “high-order” bits of the coefficients in this vector.
2. The verifier accepts the commitment and creates a challenge **c**.
3. The prover creates the potential signature (notice the usage of the random secret nonce and of the private key) and performs checks on the sizes of several parameters which makes the signature secure. This is the answer to the challenge.
4. The verifier receives the signature and computes **w1** to be the “high-order” bits of (notice the usage of the public key). They accept this answer if all the coefficients of z are less than the security parameter, and if **w1** is equal to **w0**.

The identity scheme is an interactive protocol that requires participation from both parties. How do we turn this into a non-interactive signature scheme where one party issues signatures and other parties can verify them?

Using the **Fiat–Shamir transformation**: instead of the verifier accepting the commitment and sending a challenge **c**, the prover computes the challenge as a hash **H(M || w1)** of the message **M** and of the value **w1**. This is an approach in which the signer has created an instance of a lattice problem, which only the signer knows the solution to. This in turn means that if a message was signed with a key, it could have only been signed by the person with access to the private key, and it can be verified by anyone with access to the public key.

## Falcon

**Falcon** is a cryptographic signature algorithm submitted to NIST Post-Quantum Cryptography Project. **Falcon** is based on the theoretical framework of Gentry, Peikert and Vaikuntanathan for lattice-based signature schemes. It instantiate that framework over NTRU lattices, with a trapdoor sampler called "fast Fourier sampling". The underlying hard problem is the short integer solution problem (SIS) over NTRU lattices, for which no efficient solving algorithm is currently known in the general case, even with the help of quantum computers.

### Algorithm Highlights

* **Security**: a true Gaussian sampler is used internally, which guarantees negligible leakage of information on the secret key up to a practically infinite number of signatures (more than 264).
* **Compactness**: thanks to the use of **NTRU** lattices, signatures are substantially shorter than in any lattice-based signature scheme with the same security guarantees, while the public keys are around the same size.
* **Speed**: use of fast Fourier sampling allows for very fast implementations, in the thousands of signatures per second on a common computer; verification is five to ten times faster.
* **Scalability**: operations have cost for degree , allowing the use of very long-term security parameters at moderate cost.
* **RAM Economy**: the enhanced key generation algorithm of Falcon uses less than 30 kilobytes of RAM, a hundredfold improvement over previous designs such as **NTRUSign**. Falcon is compatible with small, memory-constrained embedded devices.

### NTRU

**NTRU** is an open-source public-key cryptosystem that uses lattice-based cryptography to encrypt and decrypt data. It consists of two algorithms: **NTRUEncrypt**, which is used for encryption, and **NTRUSign**, which is used for digital signatures. Unlike other popular public-key cryptosystems, it is resistant to attacks using Shor's algorithm.

## SPHINCS +

SPHINCS+ is a stateless hash-based signature scheme, which was submitted to the NIST post-quantum crypto project. It incorporates multiple improvements, specifically aimed at reducing signature size.

These signature schemes are obtained by instantiating the SPHINCS+ construction with SHAKE256, SHA-256, and Haraka, respectively.

# Background

## Introduction to Quantum Computers

Quantum computers operate on the principles of quantum mechanics, which allows them to perform complex calculations much faster than classical computers. Classic physics cannot explain the operation of these quantum devices, and a scalable quantum computer could perform some calculations exponentially faster than any modern computer.  
In particular, a large-scale quantum computer could break widely used encryption schemes.

### Qubits (Quantum Bits)

Qubits are the fundamental unit of information in quantum computing, analogous to classical bits. [Unlike classical bits that are either **0** or **1**, qubits](https://www.technologyreview.com/2019/01/29/66141/what-is-quantum-computing/) can represent numerous possible combinations of **1** and **0** at the same time. This ability to simultaneously be in multiple states is called **superposition**. To put qubits into superposition, researchers manipulate them using precision lasers or microwave beams.

#### Physical representation

Qubits can be implemented using various quantum systems, such as:

* + - * Superconducting circuits (which isolate an electrical current by eliminating electrical resistance)
      * Trapped ions (which confine a single atomic particle using electromagnetic fields)
      * Photons
      * Quantum dots
      * Nitrogen-vacancy centers in diamond.

#### State representation

The state of a qubit is typically represented using Dirac notation, e.g., |0⟩, |1⟩, or a superposition α|0⟩ + β|1⟩, where α and β are complex numbers satisfying |α|^2 + |β|^2 = 1.

#### Measurement

When measured, a qubit collapses to either |0⟩ or |1⟩ with probabilities determined by α and β.

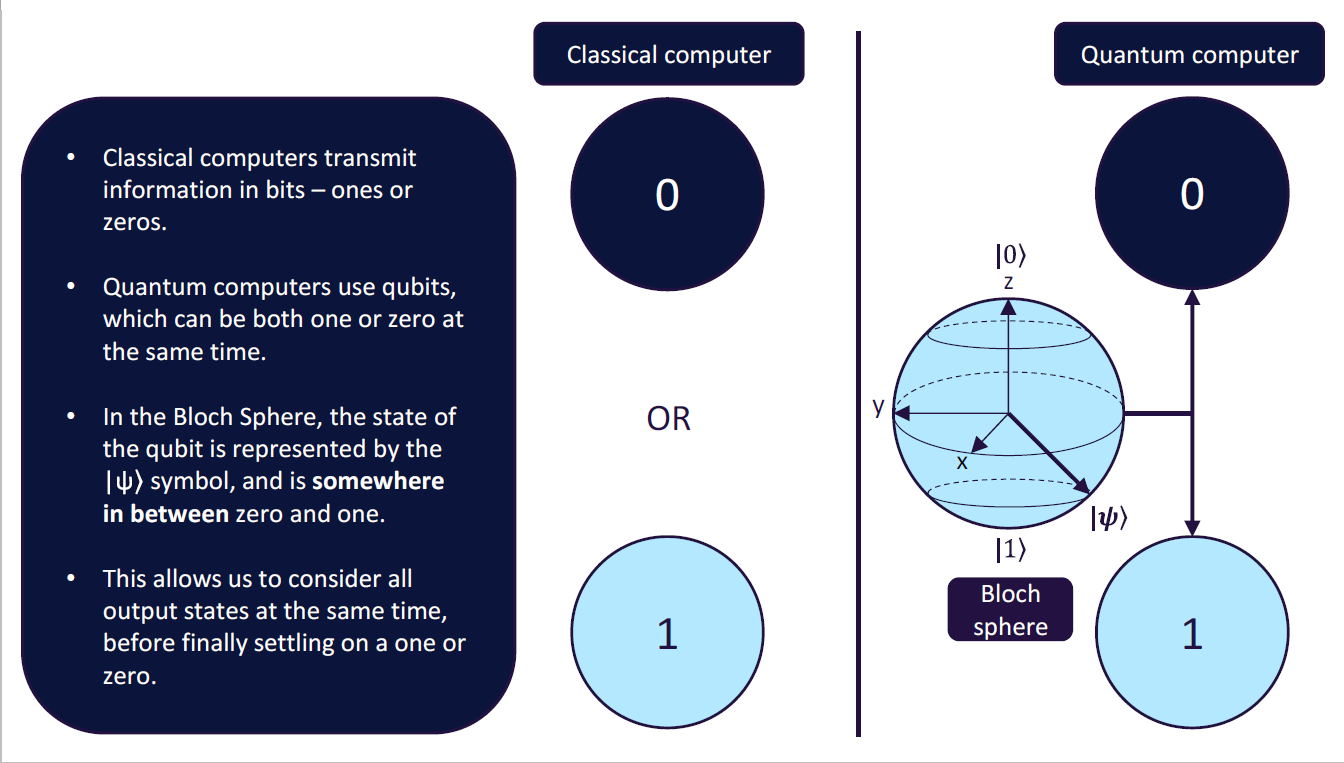


Figure 3

### Superposition

Superposition is a fundamental principle of quantum mechanics that allows qubits to exist in multiple states simultaneously.   
Mathematical representation: A qubit in superposition is represented as α|0⟩ + β|1⟩. Time frame where qubit remains stable and uncorrupted by external noise is called **coherence.**  Qubits are represented by bits in a superposition state vector – described using complex numbers. When measured or based on certain interactions with the environment, wave function collapses and qubit is transformed into a classical bit based on probabilities, Affect of collapse is called **decoherence**.

**Implications**

This property allows quantum computers to perform parallel computations on exponentially many states simultaneously.

**Interference**

Superposition states can interfere with each other, leading to constructive or destructive interference in quantum algorithms.

### Entanglement

Entanglement is a quantum phenomenon where two or more qubits become correlated in such a way that the quantum state of each qubit cannot be described independently, no matter the distance between them.This correlation can be used while particle is in quantum superposition state – **coherence**.

#### Applications

Entanglement is a crucial resource for quantum teleportation, superdense coding, and quantum cryptography.

* **Quantum teleportation** is a technique for transferring quantum information from one location to another without physically transmitting the quantum state itself.
* **Superdense coding** is a quantum communication protocol that allows the transmission of two classical bits of information by sending only one qubit.

#### Measurement

Measuring one entangled qubit instantly affects the state of its entangled partner, regardless of distance.

### Key Aspects of Quantum Interference

Interference in quantum mechanics is analogous to wave interference in classical physics, but it occurs with probability amplitudes rather than physical waves. It's a direct consequence of the superposition principle and is key to many quantum algorithms.

#### Superposition and Amplitude

Quantum states are described by complex-valued amplitudes.  
The probability of measuring a particular outcome is the square of the magnitude of its amplitude.  
In superposition, multiple basis states coexist with different amplitudes.

#### Constructive and Destructive Interference

Constructive interference occurs when amplitudes add up, increasing the probability of a particular outcome.  
Destructive interference occurs when amplitudes cancel out, decreasing the probability of an outcome.

#### Phase

The phase of a quantum state is crucial for interference.  
States with the same magnitude but different phases can interfere differently.

### Quantum Gates and Circuits

In quantum computing, a **quantum logic gate** (or simply **quantum gate**) is a basic quantum circuit operating on a small number of qubits. Quantum logic gates are the building blocks of quantum circuits, like classical logic gates are for conventional digital circuits. Unlike many classical logic gates, quantum logic gates are reversible.

#### Single-qubit gates

* Hadamard gate (H) - Creates superposition
* Pauli gates (X, Y, Z) - Rotations around different axes
* Phase gates (S, T) - Introduce phase shifts

#### Multi-qubit gates

* Controlled-NOT (CNOT) - Two-qubit gate, fundamental for entanglement
* Toffoli gate - Three-qubit gate, universal for reversible classical computation

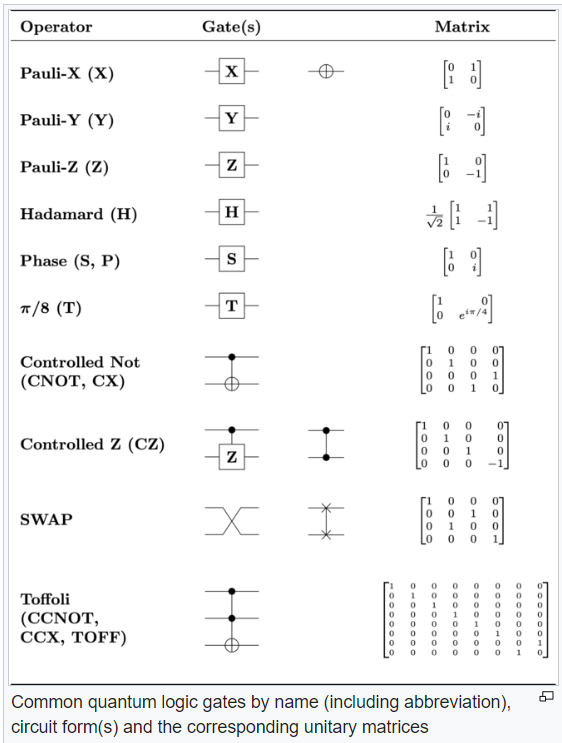


Figure 4

### Quantum Error Correction

Due to the fragile nature of quantum states, error correction is crucial for reliable quantum computation. Errors comes from Decoherence, gate imperfections and measurement errors. Types of errors are: Bit flips, phase flips, or combinations.

**Correction Techniques**:

* Quantum error correcting codes (e.g., Shor code, Steane code)
* Surface codes
* Topological quantum computing

### Quantum Supremacy

It’s the point at which a quantum computer can solve certain computation or optimization problems that is demonstrably beyond the reach of even the most powerful supercomputer. companies are already starting to experiment with quantum computers made by companies like IBM, Rigetti, and D-Wave, a Canadian firm.

### Challenges in Quantum Computing

Major hurdles in developing practical quantum computers include:

* + - **Scalability**: Building large-scale systems with many qubits
    - Coherence times: Maintaining quantum states long enough for computation
    - **Error rates**: Reducing errors in quantum gates and measurements
    - **Quantum-classical interface**: Efficiently transferring information between systems
    - **Algorithm development:** Creating new algorithms that exploit quantum properties

## Cryptography

Cryptography is the science and art of securing information by transforming it into a format that is unreadable to unauthorized individuals. It plays a critical role in ensuring the confidentiality, integrity, and authenticity of data, whether it's stored, transmitted, or processed.  
At its core, cryptography involves techniques like encryption and decryption. **Encryption** converts plain, readable data (plaintext) into a scrambled, unreadable format (ciphertext) using a specific algorithm and a key. Only those with the correct key can reverse the process through **decryption** to recover the original data.  
Cryptography has evolved over centuries, from simple ciphers like the Caesar cipher used by Julius Caesar, to complex mathematical algorithms employed in modern computing. Today, cryptography is foundational in securing communication systems, protecting sensitive information, and ensuring secure transactions over the internet.   
Key areas of cryptography include:

* **Symmetric Cryptography**: Both the sender and the receiver use the same key for encryption and decryption. Examples include the Advanced Encryption Standard (AES) and the Data Encryption Standard (DES).
* **Asymmetric Cryptography**: Uses a pair of keys—a public key for encryption and a private key for decryption. This is used in protocols like RSA and ECC (Elliptic Curve Cryptography).
* **Hash Functions**: Cryptographic hash functions generate a fixed-size output (hash) from input data, which is typically used for integrity checks. Common examples include SHA-256 and MD5.
* **Digital Signatures**: Provide a way to verify the authenticity of a message or document, ensuring it was sent by a specific sender and has not been altered.

Cryptography is essential in securing everything from personal communications and financial transactions to national security systems. As technology advances, cryptography continues to adapt, developing new methods to combat emerging threats.

### Asymmetric Cryptography

Public-key cryptography allows communicating parties to establish a shared secret key over a public channel. The concept of public-key cryptography also enables creating digital signatures for authentication of the source and validation of the integrity of data. Asymmetric cryptographical algorithms in use today are: RSA, Elliptic Curve, Diffie Helman, El Gamal All based on some combination of prime factorization or discrete logarithms.

### RSA

RSA (Rivest–Shamir–Adleman) is a public-key cryptosystem, one of the oldest widely used for secure data transmission. The initialism "RSA" comes from the surnames of Ron Rivest, Adi Shamir and Leonard Adleman, who publicly described the algorithm in 1977. In a **public**-key cryptosystem, the **encryption** key is public and distinct from the **decryption** key, which is kept secret (**private**). An RSA user creates and publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers are kept secret. Messages can be encrypted by anyone, via the public key, but can only be decrypted by someone who knows the private key.

A basic principle behind **RSA** is the observation that it is practical to find three very large positive integers such that for all integers both have the same reminder when divided by . .

However, when given only e and n, it is extremely difficult to find d.

The security of RSA relies on the practical difficulty of factoring the product of two large prime numbers, the "factoring problem". Breaking **RSA** encryption is known as the **RSA** problem. Whether it is as difficult as the factoring problem is an open question.

There are no published methods to defeat the system if a large enough key is used. **RSA** is a relatively slow algorithm. Because of this, it is not commonly used to directly encrypt user data. More often, **RSA** is used to transmit shared keys for symmetric-key cryptography, which are then used for bulk encryption–decryption.

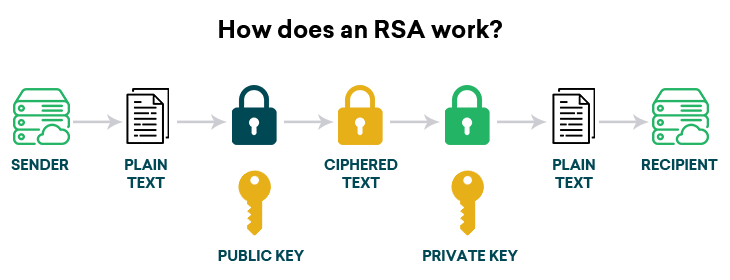


Figure 5

### Elliptic Curve Cryptography

Elliptic-curve cryptography (**ECC**) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC allows smaller keys to provide equivalent security, compared to cryptosystems based on modular exponentiation in Galois fields, such as the **RSA** cryptosystem and **ElGamal** cryptosystem.

An elliptic curve is a plane curve over a finite field (rather than the real numbers) which consists of the points satisfying the equation:

along with a distinguished point at infinity, denoted ∞. The coordinates here are to be chosen from a fixed finite field of characteristic not equal to 2 or 3, or the curve equation would be somewhat more complicated. This set of points, together with the group operation of elliptic curves, is an abelian group, with the point at infinity as an identity element. The security of elliptic curve cryptography depends on the ability to compute a point multiplication and the inability to compute the multiplicand given the original point and product point. The size of the elliptic curve, measured by the total number of discrete integer pairs satisfying the curve equation, determines the difficulty of the problem.

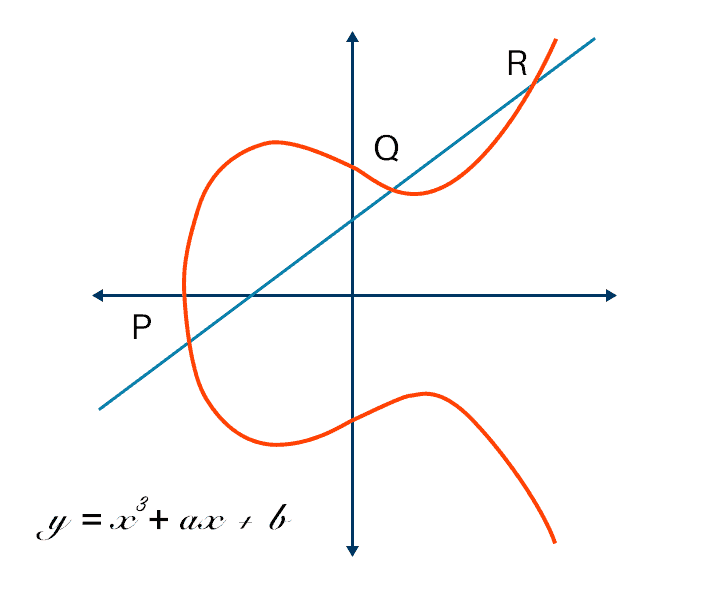
The primary benefit promised by **elliptic curve cryptography** over alternatives such as **RSA** is a smaller key size, reducing storage and transmission requirements. For example, a 256-bit elliptic curve public key should provide comparable security to a 3072-bit RSA public key. Elliptic curves are applicable for key agreement, digital signatures, pseudo-random generators and other tasks. Indirectly, they can be used for encryption by combining the key agreement with a symmetric encryption scheme. They are also used in several integer factorization algorithms that have applications in cryptography, such as Lenstra elliptic-curve factorization.

Figure 6

### Diffie Helman

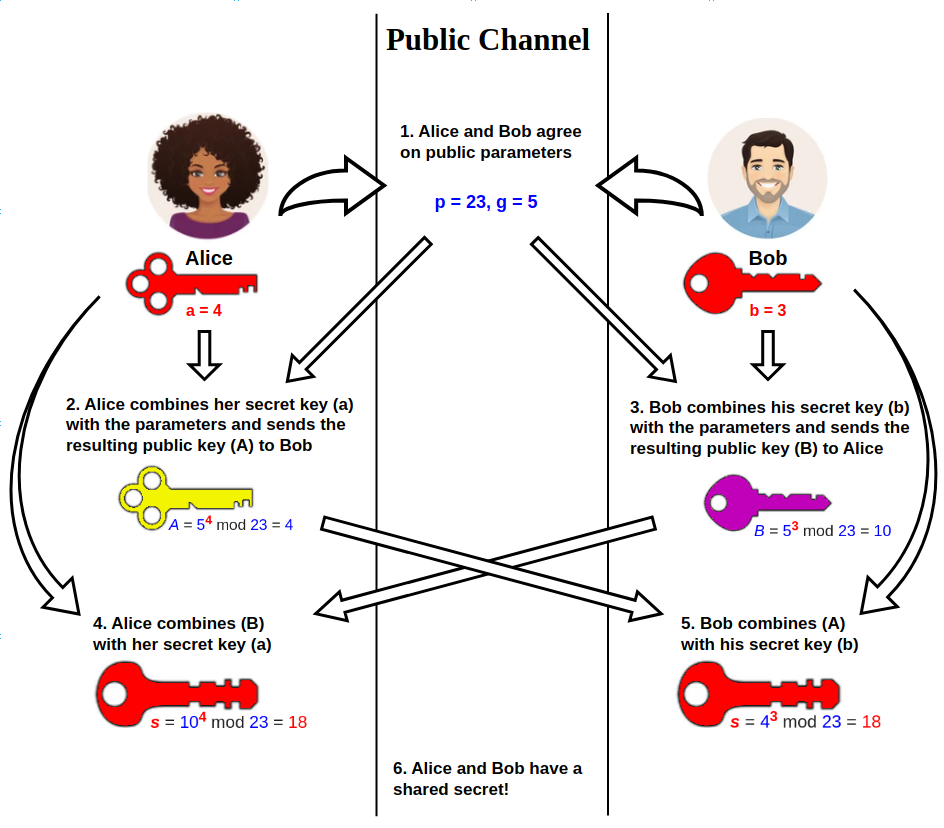
Diffie–Hellman (**DH**) key exchange is a mathematical method of securely exchanging cryptographic keys over a public channel and was one of the first public-key protocols as conceived by Ralph Merkle and named after Whitfield Diffie and Martin Hellman.  
Traditionally, secure encrypted communication between two parties required that they first exchange keys by some secure physical means, such as paper key lists transported by a trusted courier. The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel. This key can then be used to encrypt subsequent communications using a symmetric-key cipher.  
Although Diffie–Hellman key exchange itself is a non-authenticated key-agreement protocol, it provides the basis for a variety of authenticated protocols, and is used to provide forward secrecy in **Transport Layer Security's** ephemeral modes (referred to as EDH or DHE depending on the cipher suite).

Figure 7

### SSL

SSL (Secure Sockets Layer) and its successor, TLS (Transport Layer Security), are protocols for establishing authenticated and encrypted links between networked computers. Although the SSL protocol was deprecated with the release of TLS 1.0 in 1999, it is still common to refer to these related technologies as “SSL” or “SSL/TLS.” The most current version is TLS 1.3, defined in RFC 8446.

Figure 8

### TLS

Transport Layer Security (TLS) is a cryptographic protocol designed to provide communications security over a computer network. The protocol is widely used in applications such as email, instant messaging, and voice over IP, but its use in securing HTTPS remains the most publicly visible.

The TLS protocol aims primarily to provide security, including privacy (confidentiality), integrity, and authenticity through the use of cryptography, such as the use of certificates, between two or more communicating computer applications. It runs in the presentation layer and is itself composed of two layers: the TLS record and the TLS handshake protocols.

The closely related Datagram Transport Layer Security (DTLS) is a communications protocol that provides security to datagram-based applications. In technical writing, references to "(D)TLS" are often seen when it applies to both versions.

Client-server applications use the TLS protocol to communicate across a network in a way designed to prevent eavesdropping and tampering.

Once the client and server have agreed to use TLS, they negotiate a stateful connection by using a handshaking procedure. The protocols use a handshake with an asymmetric cipher to establish not only cipher settings but also a session-specific shared key with which further communication is encrypted using a symmetric cipher.   
During this handshake, the client and server agree on various parameters used to establish the connection's security.

#### TLS Handshake

When the connection starts, the record encapsulates a "control" protocol – the handshake messaging protocol. This protocol is used to exchange all the information required by both sides for the exchange of the actual application data by TLS. It defines the format of messages and the order of their exchange. These may vary according to the demands of the client and server – i.e., there are several possible procedures to set up the connection. This initial exchange results in a successful TLS connection (both parties ready to transfer application data with TLS) or an alert message.

1. **Negotiation phase:**

* A client sends a **ClientHello** message specifying the highest TLS protocol version it supports, a random number, a list of suggested cipher suites and suggested compression methods. If the client is attempting to perform a resumed handshake, it may send a session ID. If the client can use Application-Layer Protocol Negotiation, it may include a list of supported application protocols, such as HTTP/2.
* The server responds with a **ServerHello** message, containing the chosen protocol version, a random number, cipher suite and a compression method from the choices offered by the client.  
  To confirm or allow resumed handshakes the server may send a session ID. The chosen protocol version should be the highest that both the client and server support.
* The server sends its **Certificate** message (depending on the selected cipher suite, this may be omitted by the server).
* The server sends its **ServerKeyExchange** message (depending on the selected cipher suite, this may be omitted by the server). This message is sent for all DHE, ECDHE and DH\_anon cipher suites.
* The server sends a **ServerHelloDone** message, indicating it is done with handshake negotiation.
* The client responds with a **ClientKeyExchange** message, which may contain a **PreMasterSecret**, public key, or nothing. (Again, this depends on the selected cipher.) This **PreMasterSecret** is encrypted using the public key of the server certificate.
* The client and server then use the random numbers and **PreMasterSecret** to compute a common secret, called the "master secret". All other key data (session keys such as IV, symmetric encryption key, MAC key) for this connection is derived from this master secret (and the client- and server-generated random values), which is passed through a carefully designed pseudorandom function.

1. The client now sends a **ChangeCipherSpec** record, essentially telling the server, "Everything I tell you from now on will be authenticated (and encrypted if encryption parameters were present in the server certificate)." The **ChangeCipherSpec** is itself a record-level protocol with content type of 20.

* The client sends an authenticated and encrypted Finished message, containing a hash and MAC over the previous handshake messages.
* The server will attempt to decrypt the client's Finished message and verify the hash and MAC. If the decryption or verification fails, the handshake is considered to have failed and the connection should be terminated.

1. Finally, the server sends a **ChangeCipherSpec**, telling the client, "Everything I tell you from now on will be authenticated (and encrypted, if encryption was negotiated)."

* The server sends its authenticated and encrypted Finished message.
* The client performs the same decryption and verification procedure as the server did in the previous step.

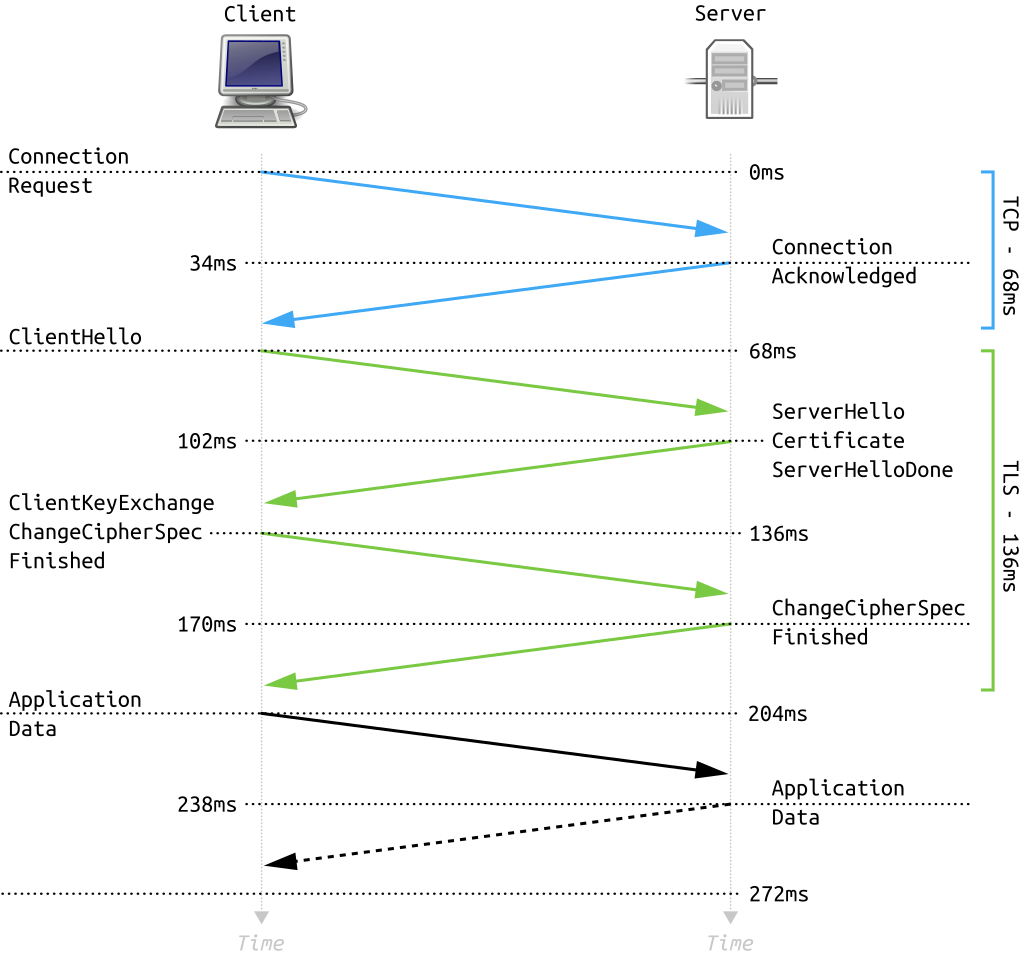
1. **Application phase**: at this point, the "handshake" is complete and the application protocol is enabled, with content type of 23. Application messages exchanged between client and server will also be authenticated and optionally encrypted exactly like in their Finished message. Otherwise, the content type will return 25 and the client will not authenticate

Figure 9

### Post Quantum Cryptography

Post-quantum cryptography (PQC) refers to cryptographic algorithms believed to be secure against attacks by both classical and quantum computers. While quantum computers excel at certain tasks like factoring large numbers (threatening RSA) and solving discrete logarithm problems (endangering elliptic curve cryptography), they are not superior for all computational tasks. Many complex problems remain challenging even for quantum computers, forming the basis for post-quantum cryptographic systems. Several promising approaches to PQC have emerged:

* **Lattice-based cryptography**: Based on the difficulty of solving certain problems in high-dimensional lattices, such as the shortest vector problem. These systems offer relatively small key sizes and efficient operations.
* **Hash-based cryptography**: Relies on the security of cryptographic hash functions, which are believed to be quantum-resistant. Particularly useful for digital signatures, though they can have large signature sizes.
* **Supersingular isogeny-based cryptography**: Utilizes the complexity of finding isogenies between supersingular elliptic curves. It offers the smallest key sizes among post-quantum systems but can be computationally intensive.
* **Code-based cryptography**: Based on the difficulty of decoding general linear codes. These systems have been studied for decades and offer fast encryption and decryption, but typically have large key sizes.
* **Multivariate cryptography**: Relies on the difficulty of solving systems of multivariate polynomial equations over finite fields. Generally fast for encryption and signatures but often has large key sizes.

Each of these approaches has its strengths and weaknesses in terms of key size, computational efficiency, and security assumptions, and research is ongoing to refine and validate these systems for widespread use in a post-quantum world.

# Shor’s Algorithm

Shor's algorithm is a quantum algorithm for finding the prime factors of an integer. It was developed in 1994 by the American mathematician Peter Shor. It is one of the few known quantum algorithms with compelling potential applications and strong evidence of super polynomial speedup compared to best known classical (non-quantum) algorithms. On the other hand, factoring numbers of practical significance requires far more qubits than available in the near future. Another concern is that noise in quantum circuits may undermine results, requiring additional qubits for quantum error correction.

Shor proposed multiple similar algorithms for solving the factoring problem, the discrete logarithm problem, and the period-finding problem. "Shor's algorithm" usually refers to the factoring algorithm but may refer to any of the three algorithms. The discrete logarithm algorithm and the factoring algorithm are instances of the period-finding algorithm, and all three are instances of the hidden subgroup problem.

On a quantum computer, to factor an integer N, Shor's algorithm runs in polynomial time, meaning the time taken is polynomial in log⁡N, where N  is the size of the integer given as input. Specifically, it takes quantum gates of order  O((log⁡N)2(log⁡log⁡N)(log⁡log⁡log⁡N))using fast multiplication, or even  O((log⁡N)2(log⁡log⁡N))utilizing the asymptotically fastest multiplication algorithm currently known due to Harvey and Van Der Hoven, thus demonstrating that the integer factorization problem can be efficiently solved on a quantum computer and is consequently in the complexity class bounded-error quantum polynomial time (BQP). This is significantly faster than the most efficient known classical factoring algorithm.

תמונה שמכילה טקסט, צילום מסך, מספר, גופן

התיאור נוצר באופן אוטומטיprogress in implementing shor’s algorithm for prime factorization

Figure 10

## Feasibility and Impact

If a quantum computer with a sufficient number of qubits could operate without succumbing to quantum noise and other quantum-decoherence phenomena, then Shor's algorithm could be used to break public-key cryptography schemes, such as

* The RSA scheme
* The Finite Field Diffie-Hellman key exchange
* The Elliptic Curve Diffie-Hellman key exchange

RSA is based on the assumption that factoring large integers is computationally intractable. As far as is known, this assumption is valid for classical (non-quantum) computers; no classical algorithm is known that can factor integers in polynomial time. However, Shor's algorithm shows that factoring integers is efficient on an ideal quantum computer, so it may be feasible to defeat RSA by constructing a large quantum computer. It was also a powerful motivator for the design and construction of quantum computers, and for the study of new quantum-computer algorithms. It has also facilitated research on new cryptosystems that are secure from quantum computers, collectively called post-quantum cryptography.

## The Procedure

### The Goal

Our goal is to factorize number **N**: given an odd composite number , find its integer factors. A complete factoring algorithm is possible if we're able to efficiently factor arbitrary into just integers greater than 1, since if either are not prime then factoring algorithm can in turn run on those until only primes remain.

### Preliminary Steps

Using **Euclid's algorithm**, we can compute the Greatest Common Divider (**GCD**) between two integers efficiently. In particular, this means we can check efficiently whether is even, in which case 2 is trivially a factor.  
Let us thus assume that **N** is odd for the remainder of this discussion.

Afterwards, we can use efficient classical algorithms to check if Nis a prime power. For prime powers, efficient classical factorization algorithms exist, hence the rest of the quantum algorithm may assume thatNg  is not a prime power.

If those easy cases do not produce a nontrivial factor ofN , Shor's algorithm proceeds to handle the remaining case.

### The Quantum Algorithm

We pick a random integer2≤a<N . A possible nontrivial divisor ofN can be found by computinggcd(a,N) , which can be done classically and efficiently using the **Euclidean algorithm**. If this produces a nontrivial factor (meaning gcd(a,N)≠1 ), the algorithm is finished, and the other nontrivial factor is Ngcd(a,N) **.**

If a nontrivial factor was not identified, then that means thatN  and the choice ofa  are **coprime**, soa is contained in the multiplicative group of integers moduloN , having a multiplicative inverse modulo**N** .

Thus, a has a multiplicative orderr moduloN , meaning and is the smallest positive integer satisfying this congruence.

The quantum subroutine findsr . It can be seen from the congruence thatN  dividesar−1 , writtenN∣ar−1 . This can be factored using the difference of squares:N∣(ar/2−1)(ar/2+1) **.**

Since we have factored the expression in this way, the algorithm doesn't work for oddr  (becausear/2  must be an integer), meaning the algorithm would have to restart with a newa . Hereafter we can therefore assumer  is even. It cannot be the case thatN∣ar/2−1 , since this would implyar/2≡1modN , which would contradictorily imply that r2would be the order ofa , instead of r .

At this point, it may or may not be the case thatN∣ar/2+1 . If it is not true thatN∣ar/2+1 , then that means we are able to find a nontrivial factor ofN . We computed=gcd(N,ar/2−1) .

If  d=1, then that meansN∣ar/2+1  was true, and a nontrivial factor of N cannot be achieved froma , and the algorithm must restart with a new a . Otherwise, we have found a nontrivial factor of N , with the other being Nd , and the algorithm is finished.

For this step, it is also equivalent to compute **gcd(N,ar/2+1),**  it will produce a nontrivial factor if gcd(N,ar/2−1)  is nontrivial, and will not if it's trivial (**where N∣ar/2+1** ).

It has been shown that this will be likely to succeed after a few runs. In practice, a single call to the quantum order-finding subroutine is enough to completely factorN  with very high probability of success if one uses a more advanced reduction.

## Quantum Advantage

The key to Shor's algorithm's power lies in its use of quantum superposition, which provides a significant advantage over classical computing methods. Here's how:

* Parallel Evaluation: In a quantum computer, Shor's algorithm can evaluate a specific function for multiple inputs simultaneously. This is fundamentally different from classical computers, which must evaluate functions for different inputs one at a time.
* Superposition: The algorithm places a quantum register into a superposition of states, effectively representing all possible inputs at once. This can be thought of as exploring multiple computational paths in parallel.
* Efficient Period Finding: By leveraging this superposition, the algorithm can efficiently find the period of a function, which is crucial for both factoring and solving discrete logarithms.
* Quantum Fourier Transform: The algorithm uses a quantum Fourier transform to extract the period information from the superposition state. This operation is particularly efficient on a quantum computer.
* Amplification of Useful Information: The quantum nature of the algorithm allows it to amplify the likelihood of measuring the useful period information, making the process much more efficient than classical alternatives.

This quantum advantage enables Shor's algorithm to solve factoring and discrete logarithm problems exponentially faster than the best known classical algorithms. It's this speed-up that poses a significant threat to current cryptographic systems based on the hardness of these mathematical problems.

The ability to perform these parallel computations and efficiently extract periodic behaviour is what gives quantum computers their power in this context, and it's why preparing for quantum-resistant cryptography is crucial for future security.

# Related Work

## Open Quantum Safe Project

The Open Quantum Safe (**OQS**) project is an open-source project that aims to support the transition to quantum-resistant cryptography. OQS is part of the Linux Foundation’s Post-Quantum Cryptography Alliance.

OQS consists of two main lines of work: **liboqs**, an open-source C library for quantum-resistant cryptographic algorithms, and prototype integrations into protocols and applications, including the widely used OpenSSL library.

While many other advanced cryptographic primitives that need to be updated to have quantum resistance, **OQS** focus is currently on post-quantum KEMs and signature schemes in the NIST PQC standardization project.

## Liboqs

liboqs is an open source C library for quantum-safe cryptographic algorithms. It provides:

* a collection of open source implementations of quantum-safe key encapsulation mechanism (KEM) and digital signature algorithms
* a common API for these algorithms
* a test harness and benchmarking routines

liboqs is updated according to NIST releases and decisions and its not fully developed yet.

## Quantum safe TLS

Liboqs integrated into forks of BoringSSL and OpenSSL3 and a standalone **OQS** provider for OpenSSL3 to provide prototype post-quantum key exchange and authentication and cipher suites in the TLS protocol.   
Researchers looking to try additional post-quantum algorithms can easily add more algorithms that follow the **OQS API**. We can use their modified implementations to prototype quantum-resistant cryptography in applications that rely on OpenSSL (such as Apache httpd, nginx, haproxy, curl, or OpenVPN) or on BoringSSL (such as Chromium).  
Currently this provider fully enables quantum-safe cryptography for KEM key establishment in TLS1.3 including management of such keys via the OpenSSL (3.0) provider interface and hybrid KEM schemes. Also, QSC signatures including CMS and CMP functionality are available via the OpenSSL EVP interface. Key persistence is provided via the encode/decode mechanism, X.509 data structures, and PKCS#12 for bundling a private key with its corresponding X.509 certificate.

# Expected Achievements

We will implement post-quantum TLS, a hybrid key exchange protocol designed to be resistant to quantum attacks, and ensure its compatibility across both old and new firmware. By integrating quantum-resistant algorithms like CRYSTALS-Kyber and CRYSTALS-Dilithium, we aim to secure communications even in the presence of quantum computing threats. The ultimate goal is to make this advanced security protocol accessible to all devices—whether they're current technologies like PCs, phones, and gaming consoles, or older systems that rely on cryptography—ensuring comprehensive protection in the quantum era.

# QSSL

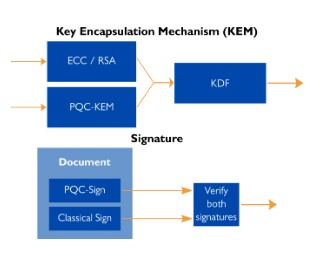
Our project aims to enhance the traditional TLS protocol by incorporating a hybrid key exchange mechanism. Hybrid key exchange combines multiple key exchange algorithms, ensuring security even if all but one of the algorithms are compromised. This approach is particularly motivated by the transition to post-quantum cryptography.  
One of the algorithms we’ll use will be quantum-resistant, ensuring protection against potential quantum attacks. This is critical for making the TLS protocol secure in the face of future quantum computing threats. While NIST’s Post-Quantum Cryptography Standardization project, initiated in 2016, is still ongoing, it has yet to finalize its recommendations, meaning the cryptographic landscape is subject to change.

Figure 11

Our project is experimental and seeks to determine whether existing technology—such as computers, smartphones, gaming consoles, and other devices that currently rely on cryptographic security—can be made resilient to quantum attacks. Essentially, we aim to explore if "older" technologies can be adapted to withstand quantum threats.

We plan to demonstrate our solution through a small application simulating bank access. During this process, the user and the bank server initiate a handshake and employ hybrid key exchange to generate a quantum-resistant key. Upon completing the handshake, the user will gain access to their account.

For now, we will utilize the algorithms selected by NIST in 2022, as newer algorithms are still in early development stages with limited information available. Our approach include the following technologies:

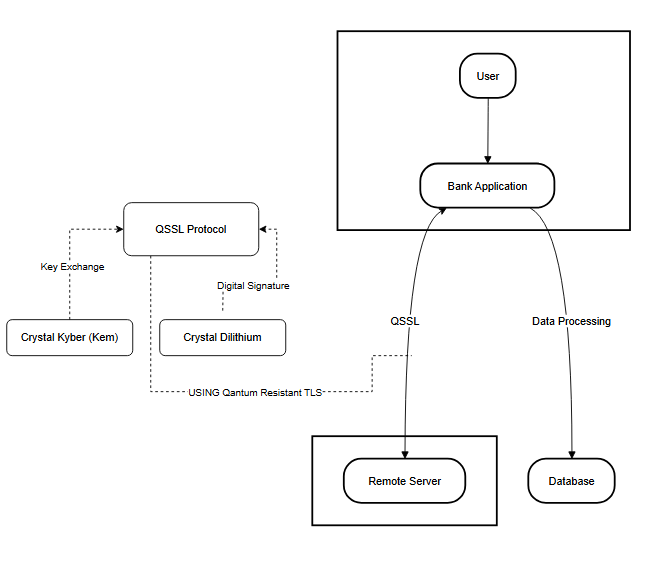
* **RSA**, **ECC, AES** – OpenSSL
* **Key Encapsulation Mechanism (KEM)** – CRYSTALS-Kyber
* **Digital Signature** – CRYSTALS-DILITHIUM

The application’s GUI will be developed in C# (.Net), while the backend will be built in C using the liboqs library.

## Product Diagram and GUI

In this section, we describe our work and a "proof of concept" draft of the proposed GUI for the system. We designed our front-end system by having a .Net application using C#. Figure **12** shows an overview of our banking application system incorporating our QSSL protocol for secure communication.

Figure 12



### Overview

1. **User Interaction**:

* The user interacts with the bank application through a graphical user interface. This bank application is responsible for handling the user's requests, such as logging in or accessing account details.

1. **Bank Application**:

* The bank application serves as the main point of interaction between the user, the remote server, and the database. It processes the user’s data and sends/receives information from both the server and the database.
* The communication between the bank application and the remote server is secured using QSSL (Quantum-Safe Secure Socket Layer), which ensures that the connection is protected from quantum-based attacks.

1. **QSSL Protocol**:

* The QSSL Protocol is a core part of securing the communication between the bank application and the remote server. It ensures both parties authenticate and agree on secure encryption keys.
* This protocol leverages hybrid key exchange, which involves using a combination of classical and quantum-resistant algorithms.

1. **Key Exchange:**

* CRYSTALS-Kyber is used for the key exchange process during the TLS handshake. This is a post-quantum cryptographic algorithm chosen for its resistance to quantum attacks.

1. **Digital Signature**:

* CRYSTALS-Dilithium is used for generating digital signatures during the TLS handshake. This ensures the authenticity of the communication and helps prevent tampering by verifying the identities of the parties involved.

1. **Remote Server & Database**:

* The remote server handles external requests and serves as the counterpart to the bank application. It communicates with the bank application over a secure QSSL connection.
* The database processes and stores the user’s data (such as account information) separately from the remote server, with data processed securely via the bank application.

1. **Quantum-Resistant TLS**:

* The overall TLS handshake and subsequent data transmission use quantum-resistant cryptography to defend against future quantum computer attacks, making this protocol especially secure.

This diagram effectively illustrates how the bank application leverages hybrid quantum-resistant cryptography (QSSL) for secure communication, using the post-quantum algorithms - CRYSTALS-Kyber and CRYSTALS-Dilithium

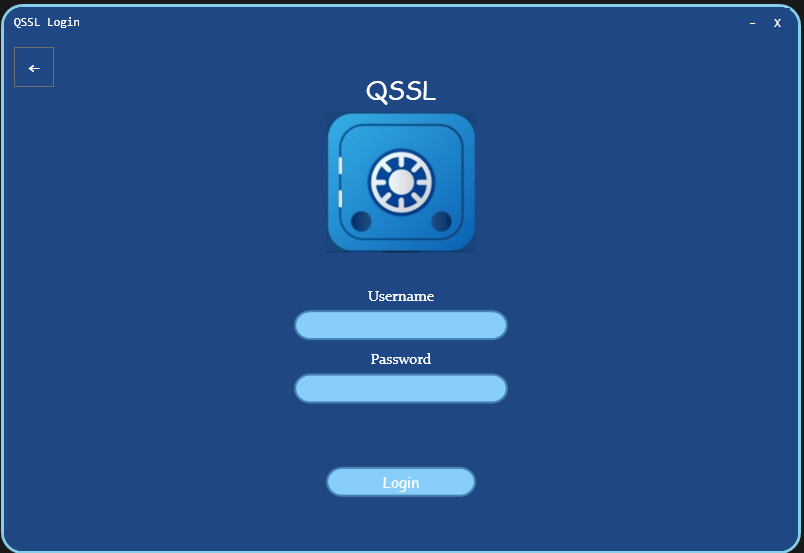
**Main Screen:**



Figure 13

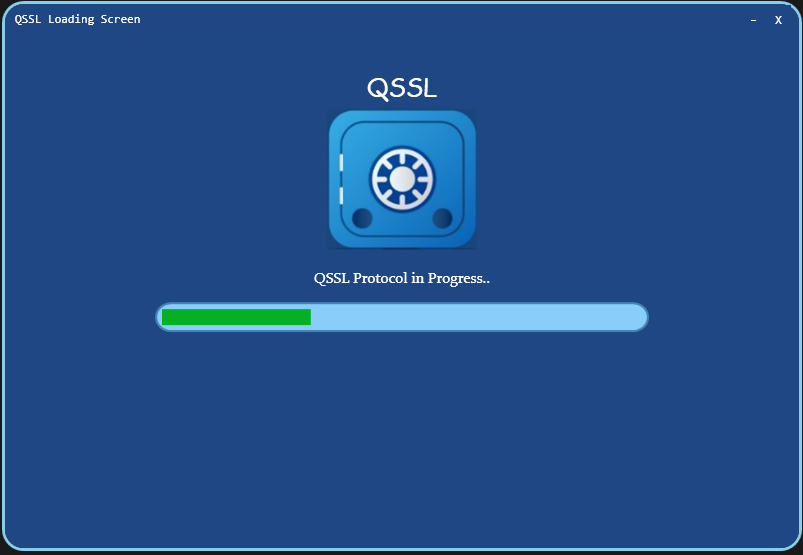
**Login Screen**:

Figure 14



**QSSL progress**:

Figure 15



**Successful Login:**

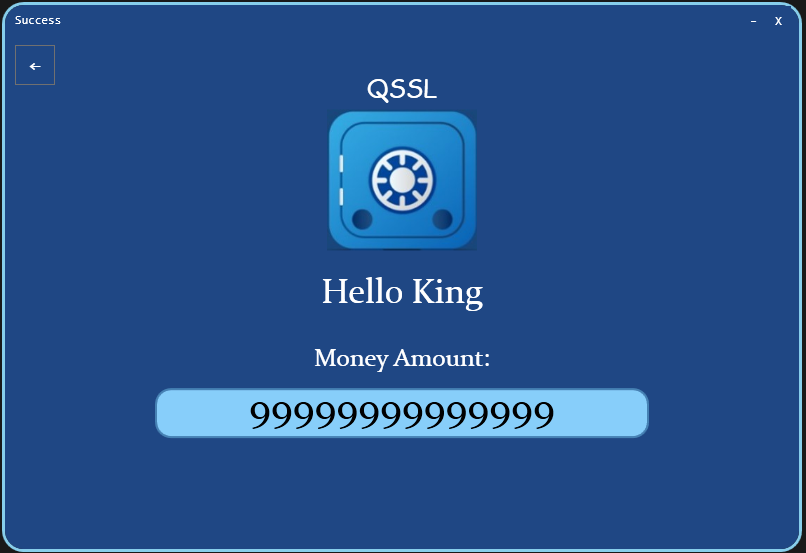


Figure 16

### QSSL Protocol

This diagrams illustrates a the Quantum Safe Secure Sockets Layer handshake process between a client and a server, and the process of delivering secret messages between the two parties.

Figure 17

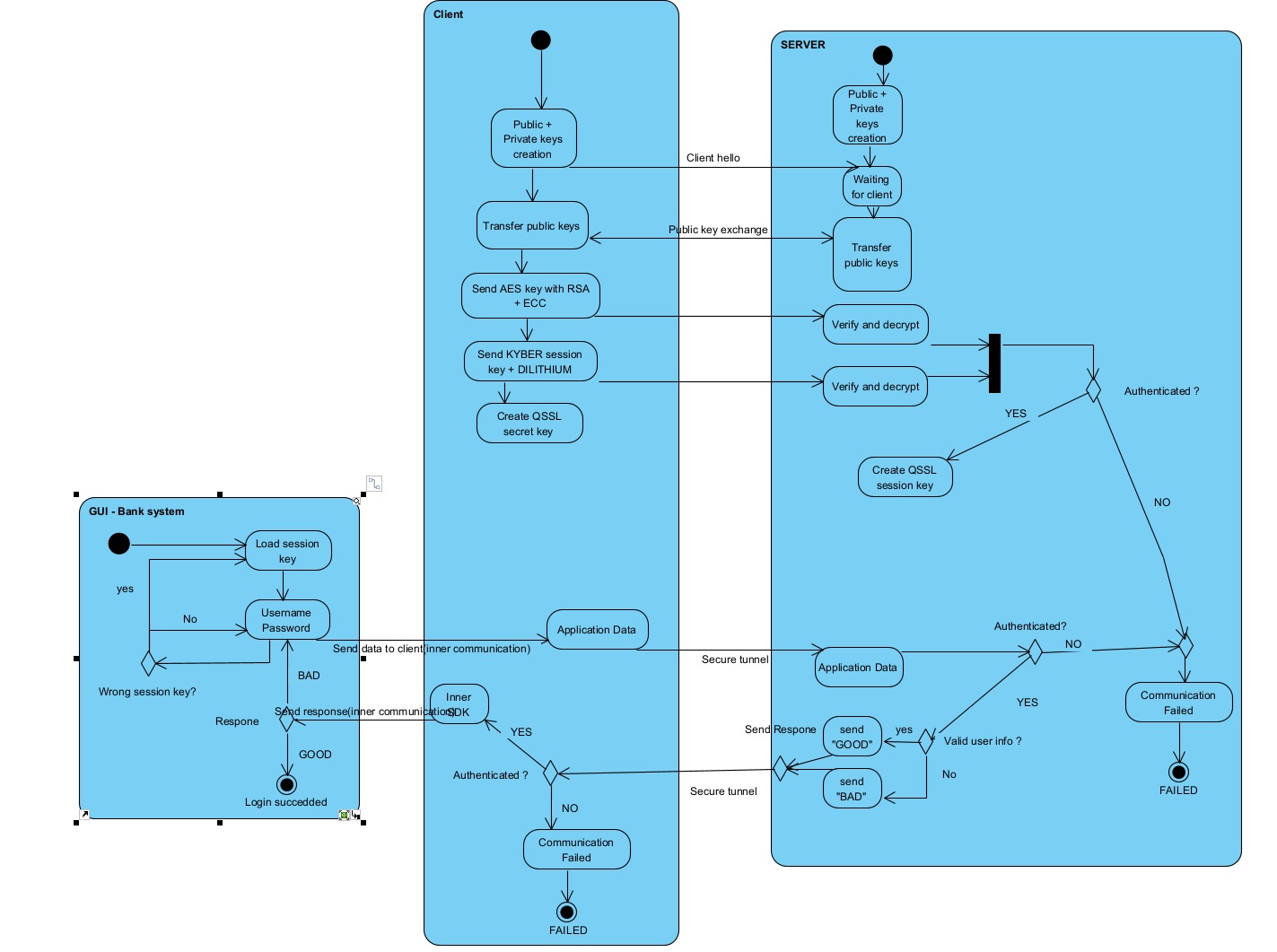
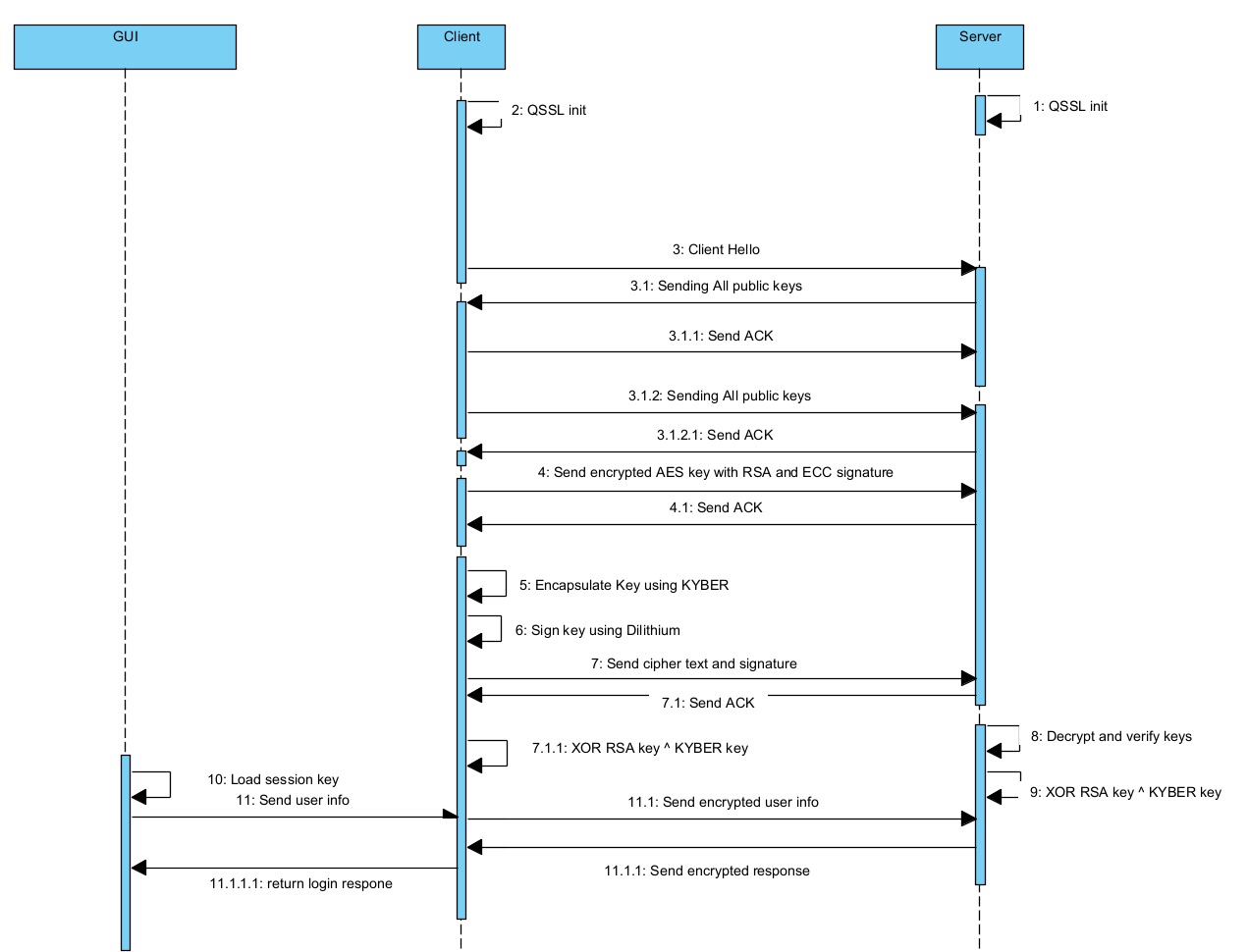


Figure 18

# References

Our GIT Repository - [https://github.com/raztibi/QsslProject](https://github.com/raztibi/PhaseA)

Introduction - <https://www.technologyreview.com/2019/01/29/66141/what-is-quantum-computing/>

CRYSTAL Project – <https://pq-crystals.org/index.shtml>

Falcon - <https://falcon-sign.info/>

SPINCHS - <https://sphincs.org/>

NIST Post Quantum Project - <https://csrc.nist.gov/projects/post-quantum-cryptography>

Digital signature terms - <https://blog.cryptographyengineering.com/euf-cma-and-suf-cma/>

Open Quantum Safe Project - <https://openquantumsafe.org/>

Liboqs repository - <https://github.com/open-quantum-safe/liboqs>

RSA - <https://en.wikipedia.org/wiki/RSA_(cryptosystem)>

Diffie Helman - <https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange>

SSL/TLS - <https://en.wikipedia.org/wiki/Transport_Layer_Security#SSL_1.0,_2.0,_and_3.0>

Elliptic Curve Cryptography- <https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/>

Shor's Algorithm - <https://en.wikipedia.org/wiki/Shor%27s_algorithm>

Qubits and Quantum PCs - RAFAEL confidential inner resources

CRYSTAL-Kyber Algorithm - <https://medium.com/@hwupathum/crystals-kyber-the-key-to-post-quantum-encryption-3154b305e7bd>

CRYSTAL – Dilithium Algorithm - <https://eprint.iacr.org/2017/633.pdf>

CRYSTAL – Dilithium - <https://blog.cloudflare.com/post-quantum-signatures/>

OUR LOGO – DALL-E, ChatGPT