Received: 11 May 2025, Accepted: 12 May 2025, Published: 15 May 2025
Digital Object Identifier: https://doi.org/10.63503/j.ijssic.2025.112

Research Article

Software Based Implementation of a Hybrid Quantum Secure Cryptosystem for Secure Communication and Data Protection

Sristi J[1], Andrew Vivan X[1*], A Ashoka[1], Y B Tulasi[1], Dr. Manjunath C R[1]

[1] Department of Artificial Intelligence and Machine Learning, CMR University, Bengaluru, India

sristij03@gmail.com, andrewvivanx@gmail.com, aashoka198@gmail.com, tulasiyadav9908@gmail.com, dr.crmanjunath@gmail.com

*Corresponding author: Andrew Vivan X, andrewvivanx@gmail.com

ABSTRACT

Current cryptographic systems and information security methods are largely based on the computational intractability and impracticability of reversing complex cryptographic algorithms and protocols. The foundation of secure communication and data protection strategies that we rely on a daily basis is provided by this computational difficulty. However, quantum computers with sufficient processing power could breach these basic security assumptions, making it possible to recover the cryptographic keys used to safeguard sensitive data and compromise widely used key exchange techniques. A viable strategy to counter the threat of quantum attacks and maintain standard security guarantees is hybrid systems. The software-based hybrid cryptographic framework that guarantees forward secrecy and contemporary cryptographic authentication while also integrating quantum-resilient key establishment schemes, namely, quantum key distribution simulation and a post-quantum key encapsulation mechanism, in a manner that also combines their security properties to generate a hybrid key that remains secure as long as at least one underlying component remains uncompromised is experimentally implemented in this paper. We demonstrate how this approach can safeguard both real-time secure communication and long-term data protection against current threats as well as future quantum adversaries.

Keywords: *Cryptography, Data Protection, Hybrid Quantum Cryptography, Post-Quantum Cryptography, Quantum Key Distribution, Quantum Threat, Secure Communication.*

1. Introduction

With the evolution of digital times, characterized by exponential increase in data generation, transmission, and storage, the challenges of securing communication and protecting data have grown more challenging. Rapid digital transformation calls for efficient techniques to preserve information in the long term since the amount of information to be secured is projected to relentlessly increase year by year [1]. Protecting confidential data from unauthorized access and capture efforts is essential, especially when it involves personal, financial, corporate, or governmental information. Cryptography aids information security and protects data defence throughout applications ranging from private communications to critical infrastructure. It ensures security services such as confidentiality, integrity, authenticity, and non-repudiation of information using the latest cryptographic algorithms and protocols which is the cornerstone of digital security by safeguarding information from unauthorized access, tampering, interception and other forms of cyber threats [2]. Recent research [3, 4] highlights how quantum computing postures noteworthy dangers to current state-of-the-art information protection mechanisms widely used for secure communication and data protection, which rely on symmetric,

asymmetric, and hash function schemes. These cryptographic schemes are built around the infeasibility of solving some mathematical problems with conventional computing methods within a reasonable timeframe, also known as the computational hardness of specific problems. However, when equipped with a sufficient number of qubits, quantum computers will be capable of employing quantum algorithms to solve these problems exponentially faster than classical computers. This shift in the paradigm of computation would render existing cryptographic systems vulnerable to vast quantum-enabled assaults. With sufficient quantum capabilities, these devices could then retroactively decrypt information classified as sensitive and securely transmitted in today's environment. This means that information believed to be secure today could ultimately be exposed [5]. This paper presents an experimental implementation of a software-based hybrid cryptographic framework that combines contemporary cryptographic authentication with quantum-resilient key establishment techniques in response to the quantum threat. At the core of this hybrid framework is quantum key distribution (QKD), which leverages simulated quantum mechanics to provide a superior source of entropy for randomness in key generation. QKD simulation also enhances security by enabling the detection of any eavesdropping attempts. In the meantime, the computationally challenging post-quantum key encapsulation mechanism is Crystals-Kyber. By combining these security features, this hybrid framework makes sure that cryptographic keys are robust and unpredictable, providing defence against attacks made possible by quantum computing. The design and simulation of the system's operation are the main topics of this paper. Testing and analysis are then used to assess the system's performance.

2. Background

The basic purpose of cryptographic systems is to guarantee safe data transfer even, when possible, attackers are present. This is achieved by converting readable data, known as plaintext, into ciphertext, an encrypted (unreadable) format that hides the original data. The ciphertext can only be restored to its original readable state by an authorized recipient who has the right decryption mechanism (key and algorithm) [6]. In order to improve security, modern cryptographic techniques rely on computational complexity, employing incredibly long keys and sophisticated algorithms. Modern cryptographic techniques fall into two major categories:

i. **Symmetric Cryptography:** This technique, also referred to as private-key cryptography, uses a shared "secret key" that is only known by the parties involved in the communication. Using this secret key, the sender encrypts plaintext into ciphertext, and the recipient uses the same secret key to decrypt the ciphertext back into plaintext. Although this approach is computationally efficient, it is still difficult to distribute and manage the secret key securely because the entire system's security is compromised if the key is lost, stolen, or intercepted during transmission [7].

ii. **Asymmetric Cryptography:** This technique, which is also referred to as public-key cryptography, employs a pair of keys; the recipient's "public key" is used to encrypt the data. The message can only be decrypted using the matching "private key," which is safely kept by the intended recipient. With a widely shared public key and a private key for each user, this method solves the key distribution issue that symmetric cryptography entails and guarantees that only the intended and authorized recipient can access the original data [8].

The resilience of modern cryptography depends on factors such as key size and the computational power required to test all potential keys and break the algorithm. The concept of a “computationally secure scheme” implies that while it is theoretically possible to crack such a system, it remains practically impossible if the cost of breaking it outweighs the value it protects [9].

3. Quantum Computing and It's Threat to The Current State-Of-The-Art in Information Protection

The exponential growth of computational capabilities in recent decades has made it necessary to continuously improve cryptographic protocols in order to preserve their security margins against traditional computational attacks. Modern protocols are designed to withstand attackers using traditional computing techniques. The emergence of quantum computing, a technology whose capabilities and processing power far surpass those of existing paradigms, presents a new challenge as classical cryptographic architectures approach their physical limits. With the ability to compromise cryptographic systems that have long been thought to be impenetrable, this new technology poses a serious threat and opens up a new channel for cryptanalysis [10, 11].

"Quantum Computing" refers to a class of technologies that, by drawing on concepts from quantum mechanics like superposition, entanglement, and interference. In quantum computing, a qubit can exist in a superposition of states, in contrast to a classical bit, which can only exist in one of two states (0 or 1) and is manipulated individually in classical computing. In order to speed up computation, a qubit can simultaneously represent 0, 1, or any combination of the two states. By connecting qubits in a way that makes their states dependent on one another regardless of distance, entanglement further increases computing power. Interference enables the probability amplitudes of qubit states to add constructively (strengthening each other) or destructively (cancelling each other out), which enables quantum algorithms to amplify correct solutions and reject incorrect ones [12]. This superposition, combined with quantum entanglement and interference, enables quantum computers to process and investigate multiple values simultaneously. Through application of these principles of quantum mechanics, quantum computation allows computations to be computed in parallel and can solve some computationally challenging problems exponentially faster than classical computers ever could [13], thereby directly impacting the security provided by modern cryptography.

3.1 Quantum Algorithms and Their Challenge to Asymmetric, Symmetric Cryptography, and Hash Functions

Asymmetric cryptosystems are secure based on computationally difficult mathematical problems. Two examples that underpin the theory are RSA, based on the difficulty of factorization of large integers (for example, products of two large primes), and schemes such as Diffie-Hellman and Elliptic Curve Cryptography (ECC), based on the hardness of the Discrete Logarithm Problem (DLP). The classical solution to these problems has sub-exponential or exponential time complexity, making them intractable using classical computation. Shor's quantum algorithm [14], in 1994, basically changed this paradigm by coming up with polynomial-time solutions for these problems. Shor's algorithm factors large integers (and thus breaks RSA) and solves the DLP over multiplicative groups (e.g., Diffie-Hellman) and additive groups (e.g., ECC), and hence breaks of these popular asymmetric cryptosystems, making them useless [15].

Symmetric cryptosystems derive their security from the computational difficulty of exhaustively searching through possible secret keys or values. One such system is the Advanced Encryption Standard (AES), where the biggest risk is a thorough key search. Brute-force attacks on AES keys are impractical in classical computing because of the large number of possible outcomes. To find the right answer to a search problem, all potential solutions must be methodically tested. Grover's quantum algorithm [16], introduced in 1996, designed for such search tasks. In classical computing, searching an unsorted space of N elements such as a cryptographic key space requires $O(N)$ operations in the worst case, as each element must be checked sequentially. Quantum computers, using Grover's algorithm, accomplish this task in $O(\sqrt{N})$ quantum operations, offering a significant speed advantage for large search spaces by

decreasing the search area over classical approaches. This quadratic speedup becomes even more pronounced as N grows larger and could dramatically reduce the time required to discover an AES key. For example, AES-256, with $N = 2^{256}$ possibilities, a classical attack would need 2^{256} operations. Grover's algorithm reduces the complexity to about $2^{256/2} = 2^{128}$ operations reducing the strength equivalent to AES-128 bit variant [17].

Hash functions such as SHA-2/SHA-3, required to maintain data integrity and digital signatures, will similarly be compromised. The Brassard-Hoyer-Tapp (BHT) algorithm [18] that appeared in 1997, combines aspects of the classical birthday attack with Grover search affords a theoretical scaling of $O(2^{n/3})$ for finding hash collisions. For instance, SHA3-256, normally offering 128-bit security, would be limited to about 85-bit security against quantum attacks [19]. This impending quantum threat requires research into new techniques to secure cryptographic processes in the post-quantum world. Of these alternatives, quantum key distribution (QKD) and post-quantum cryptography (PQC) present complementary strategies to attain resilient security.

4. Overview of Quantum Key Distribution

A technique for distributing keys that safely transmits encryption keys by utilizing the characteristics of individual light particles (photons) and the concepts of quantum mechanics. Photons, which are only moving particles that cannot be precisely replicated without changing or destroying their original state, are used in these systems to encode the keys. Since it alters their quantum state, any attempt to intercept or eavesdrop on these photons can be detected. Unauthorized interception is very challenging due to the intrinsic properties of these photons. Building on these basic characteristics, the BB84 protocol [20], which describes a systematic procedure for creating a secure shared key, is one of the most promising applications of quantum key distribution (QKD). By creating a random bit sequence made up of 0s and 1s, the sender starts the process. Bits are selected with equal probabilities, which serves as the foundation for the cryptographic key. As indicated in Table 1, the sender simultaneously prepares the quantum states for each bit in the sequence by choosing an encoding basis at random from a set of four potential polarization states. Next, a corresponding quantum state (qubit) is assigned to each bit. By embedding the information in the photons' quantum states, this encoding makes sure that it can only be accessed under particular measurement circumstances [21].

Table 1: Polarization states and their basis representations.

Basis	**State**	**Polarization**
Computational Basis (+)	$\|0\rangle$	Horizontal polarization
	$\|1\rangle$	Vertical polarization
Diagonal Basis (x)	$\|+\rangle$	45° polarization
	$\|-\rangle$	135° polarization

The prepared qubits are sent to the recipient through a quantum communication channel as part of the sender's quantum state transmission process. By maintaining the photons' quantum properties, this specialized channel enables the photons to be received by the receiver in the same polarization states as when they were sent. The receiver independently and at random chooses one of two possible measurement bases (diagonal or computational) for each qubit after receiving the qubits. Since each photon's basis selection is random, it is impossible for an eavesdropper to predict how the photons will be measured. A crucial component of the protocol's security is the receiver's measurement uncertainty, which prevents an eavesdropper from obtaining valuable information without being detected.

Depending on the basis selected, the measurement collapses the quantum state into a classical bit value (0 or 1). Following transmission and measurement, the sender and recipient compare the bases used for encoding and measuring each qubit in a public discussion known as Basis Reconciliation. Crucially, they don't reveal the actual bit values; they just reveal the basis choices [22]. As indicated in Table 2, they only keep the bits where their bases line up and discard the ones where they don't. The "raw key" is a shorter, shared bit sequence that is produced by this sifting process.

Table 2: Possible outcomes of photon exchanges between the sender and receiver in the BB84 protocol.

Sender's Bit	**Sender's Basis**	**Sender's Polarization**	**Receiver's Basis**	**Receiver's Measurement**	**Outcome**
0	+	Horizontal	+	0	0 (Kept)
0	+	Horizontal	x	Equal chance of 0 or 1	Discarded
1	+	Vertical	+	1	1 (Kept)
1	+	Vertical	x	Equal chance of 0 or 1	Discarded
0	x	45°	+	Equal chance of 0 or 1	Discarded
0	x	45°	x	0	0 (Kept)
1	x	135°	+	Equal chance of 0 or 1	Discarded
1	x	135°	x	1	1 (Kept)

However, disparities between their keys are introduced by transmission flaws or possible eavesdropping. One party's sifted key (for example, the sender's) is selected as the reference, and any discrepancies with the receiver's key are identified as errors. This is one of the additional post-processing steps that the sender and receiver take to guarantee the security and accuracy of their raw keys. Low-density parity-check (LDPC) codes and interactive error correction procedures are two methods used to fix these errors. These methods involve exchanging parity-check bits in order to identify and fix errors. The Quantum Bit Error Rate (QBER), which is computed as stated in (1), is used to evaluate the key's security.

$$QBER = \frac{number\ of\ erroneous\ bits}{total\ shifted\ key\ bits} \tag{1}$$

For the BB84 protocol, the protocol is terminated if the QBER surpasses a threshold because high errors suggest potential eavesdropping or too much noise in the quantum channel [23]. Because any attempt by an eavesdropper to measure the quantum states disturbs them and introduces detectable errors, this ensures security. Privacy amplification is applied to mitigate any information leakage ℓ_{EC} from the error correction process in order to further improve security. Both the sender and the recipient compress their verified key using a two-universal hash function. The final secure key length l is determined as specified in equation (2),

$$l = n - \ell_{EC} - s \tag{2}$$

Where n is the sifted key length, ℓ_{EC} accounts for the information revealed during error correction, and s is a security parameter ensuring negligible knowledge for an eavesdropper [24]. Despite offering superior security, QKD has significant drawbacks that prevent it from being widely used. QKD necessitates specialized hardware and infrastructure, like quantum channels or optical fibres, which are expensive and challenging to implement globally. Additionally, it is limited by distance, necessitating the use of quantum repeaters for communication over long distances. Furthermore, QKD is not feasible for large-scale networks because it is best suited for direct, point-to-point communication. Real-world

deployment is complicated by environmental noise, hardware flaws, and high maintenance costs [25]. Because of these difficulties, a different strategy is required, and post-quantum cryptography (PQC) is the solution. PQC is a crucial addition to QKD because of its scalability, flexibility, and affordability, which help to solve the more general issues with quantum-safe encryption.

5. Overview of Post-Quantum Cryptography

Quantum-resistant cryptographic techniques are being actively standardized by the National Institute of Standards and Technology (NIST). Unlike to QKD, post-quantum cryptography (PQC) is software-based, and its security is based on mathematical algorithm problems that are computationally challenging and thought to be unsolvable even by a large-scale quantum computer. PQC can withstand attacks enabled by quantum computer and integrates easily into current digital infrastructure without the need for specialized hardware [26]. An outline of these completed mathematical families can be found below.

i. **Lattice-based cryptography:** Lattice-based cryptography derives its security from the hardness of solving computational problems in mathematical lattices. This family includes CRYSTALS-Kyber, a key encapsulation mechanism (KEM) for secure key exchange [27], and CRYSTALS-Dilithium and FALCON, which provide digital signatures for authentication [28]. Most lattice-based key establishment algorithms are simple, efficient, and highly parallelizable. Additionally, some systems offer provable security under worst-case hardness assumptions, providing a stronger guarantee than average-case security.

ii. **Hash-based cryptography:** Hash-based cryptography is based on the security properties of cryptographic hash functions. One of the selected algorithms in this family is SPHINCS+, which offers a stateless hash-based digital signature scheme [29]. This approach ensures long-term security against quantum attacks without relying solely on the security of lattice-based methods.

iii. **Code-based cryptography:** Code-based cryptography leverages the hardness of decoding general error-correcting codes, a problem believed to remain intractable even for quantum computers. One of the selected algorithms in this category is HQC (Hamming Quasi-Cyclic), a key encapsulation mechanism (KEM) based on quasi-cyclic codes without relying on hidden trapdoors, which serves as a backup key encapsulation mechanism (KEM) alongside CRYSTALS-Kyber [30].

In terms of speed and performance, PQC systems are currently less effective than contemporary cryptography systems. They frequently need more memory, bandwidth, and processing power, particularly for large-scale applications. The intricacy of quantum-resistant algorithms and the requirement for strong security measures against quantum threats are the causes of this [31]. Threats to post-quantum cryptography (PQC) continue despite the theoretical difficulty of the underlying mathematical issues. Beyond the hypothetical prospect of massive quantum computers, the quantum threat also includes real-world developments in quantum or classical algorithms that target particular PQC schemes. For instance, quantum algorithms have been proposed to break multivariate and isogeny-based PQC schemes, some of which were excluded from standardization efforts. This underscores that quantum threats to PQC are dynamic and evolving, necessitating continuous monitoring and evaluation. Relying solely on a single PQC algorithm carries the risk that unforeseen vulnerabilities will surface and expose data once the algorithm is implemented in real-world settings. By combining PQC with well-known classical primitives (like RSA), hybrid cryptosystems provide layered security, so that a single flaw wouldn't bring down the entire system. However, because hybrid PQC-classical systems still rely on mathematical presumptions that are susceptible to algorithmic or quantum advances, they do not overcome the fundamental drawbacks of computational cryptography. Furthermore, PQC schemes are susceptible to implementation errors such as fault injection vulnerabilities, side-channel

attacks, and changing standards, which call for flexible migration plans and constant attention to handle both theoretical and real-world risks [32, 33, 34].

6. Hybrid Quantum-Secure Cryptographic Framework

In this section, we present our software-based hybrid quantum-secure cryptosystem implementation, detailing the selected cryptographic primitives and the experimental setup. The limitations of QKD and relying only on PQC or even hybrid PQC-classical systems are not enough for long-term security in the post-quantum era, as was covered in earlier chapters. Our design expands upon similar frameworks that incorporate the idea of authenticated key exchange (AKE) and key combiners, as suggested in [35, 36, 37, 38, 39]. In particular, their method blends quantum-resistant and classically secure schemes. For secure communication and strong data protection, our method makes use of a comprehensive, tiered framework that combines post-quantum cryptography (PQC), quantum key distribution (QKD), and traditional cryptographic authentication.

6.1 Experimental Setup and Implementation Workflow

In order to provide a user-friendly web interface for initiating and managing various cryptographic convention scripts, our test setup makes use of Django, a Python-based web framework operating on a Windows-based operating system. The application is modularized into five distinct modules: quantum key distribution (QKD) simulation, kyber key exchange (KKE), key management system (KMS), quantum secure communication and encryption/decryption framework. These modules are made with C/C++ for the performance-critical parts and Python 3 for the high-level logic. As shown in Figure 1, the Django application uses sub-process calls to launch specific Python scripts for every cryptographic operation. Django does not store or process any experimental data that is sent or received in relation to key distribution, key exchange, or communication, even though it controls script execution, termination, and interfaces with a secured SQLite database to store cryptographic keys and user credentials. The cryptographic operations run as independent processes, each operating on unique ports, except for those within a specific module. Through the use of a Python TCP socket channel, QKD simulation, KKE, and secure communication modules enable end-to-end connections between sender and recipient parties. A VPN point-to-point tunnelling protocol (PPTP) is used exclusively to create the connection in cross-network configurations, where the sender and the recipient are in different locations and connected to different subnets. In nearby organize arrangements (sender and recipient are in same area associated to same subnet), direct TCP socket connections are permitted, provided network isolation and firewall policies permit it.

Assume that the communicating entities are sender (S) and receiver (R). In order to ensure that only verified parties can move forward with secure key establishment for both BB84 quantum key distribution (QKD) and kyber key exchange (KKE), our framework starts with a mutual authentication mechanism, as illustrated in Algorithm 1.

Algorithm 1 Pre-shared Key (PSK) Hash Validation Based Authentication

1:	S $\rightarrow R$: $H_S = SHA\text{-}256\ (PSK_S)$
2:	$H_R \leftarrow SHA\text{-}256\ (PSK_R)$
3:	$\boldsymbol{if}\ H_R \neq H_S\ \boldsymbol{then}$ "$Abort$" $\boldsymbol{else}\ Authenticate = $ "$Success$"
4:	$\boldsymbol{return}\ Legitimate\ connection$

A pre-shared key (PSK) is used by both the sender and the recipient. The pre-shared secret is subjected to a one-way SHA-256 hash, and the hash value is transmitted from the sender to the recipient via the

socket channel for authentication. In order to compare the received value with the hash of its stored PSK, the receiver independently calculates it. Authentication is successful if the hashes match, guaranteeing that any further communications are secure and authentic; if they don't, the protocol terminates.

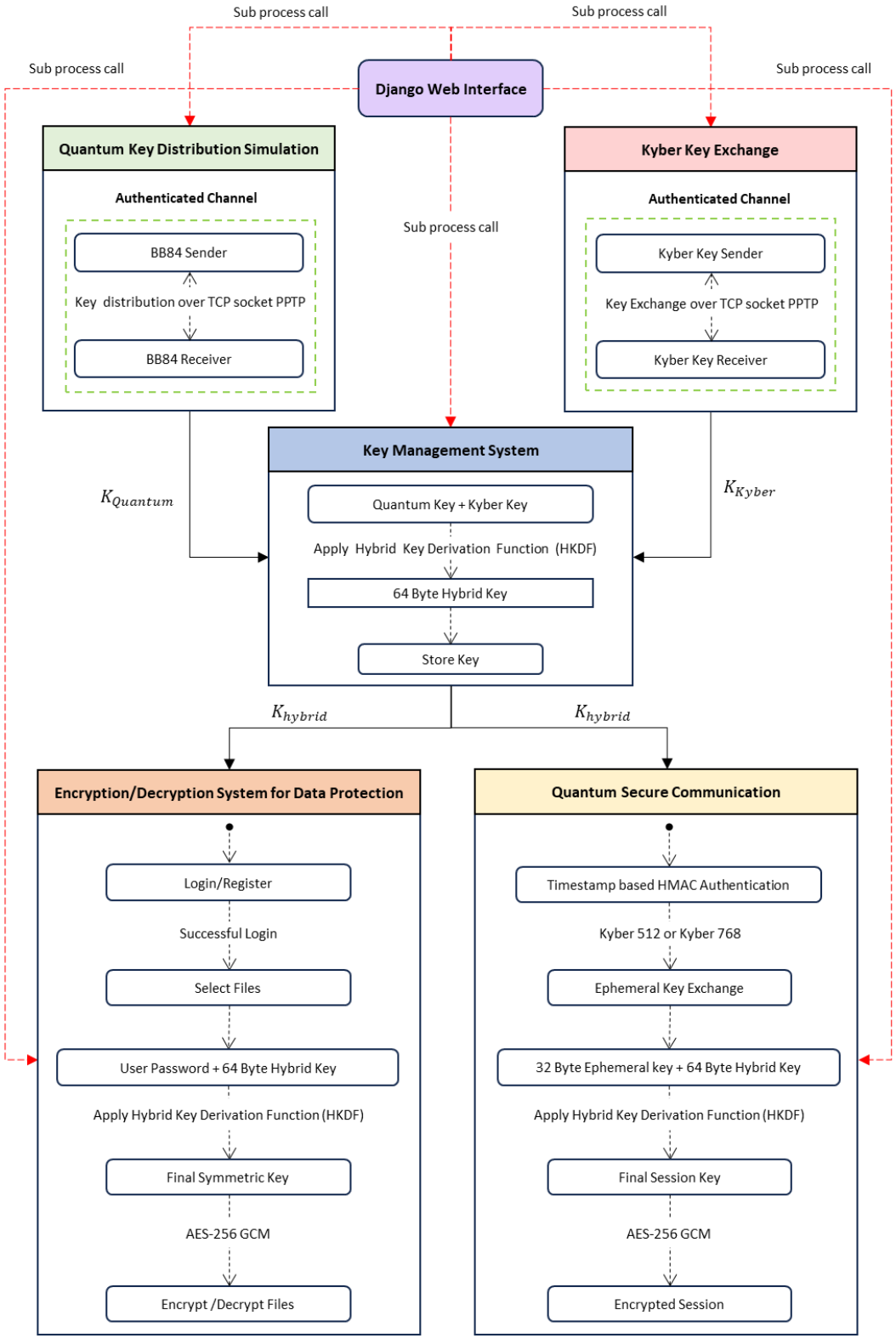

Figure 1: Experimental Setup of the Software-Based Hybrid Quantum-Secure Cryptosystem

Algorithm 2 describes how to start the BB84 protocol simulation in our hybrid framework after it has been authenticated. The open-source quantum computing framework Qiskit, created and maintained by IBM, has been chosen as the source for quantum key generation and offers a full suite of tools for creating and modelling quantum circuits. The higher entropy needed for key generation, specifically in bit values and basis selection, is accomplished using Aer, a high-performance quantum circuit simulator included in the qiskit framework, even though our simulation does not use any physical quantum channel or hardware. Aer creates quantum states (such as qubits in superposition and quantum measurement) in a regulated software environment by simulating quantum operations using classical computing resources.

Algorithm 2 BB84 Quantum Key Distribution (QKD) simulation using Qiskit-Aer

1:	$n \leftarrow number\ of\ qubits$
2:	$bits_S \leftarrow Generate\_Random\_Bits.Aer(n)$
3:	$basis_S \leftarrow Generate\_Random\_Basis.Aer(n)$
4:	$Q_C \leftarrow Encode.Aer(bits_S, basis_S)$
5:	$Send\ (Q_C)\ to\ R$
6:	$basis_R \leftarrow Generate\_Random\ \_Basis.Aer(n)$
7:	$measured_{bits_R} \leftarrow Measure.Aer(Q_C, basis_R)$
8:	$Public\ Compare\ (basis_S, basis_R)$
9:	$matching\_indices \leftarrow \{i \mid basis_S[i] = basis_R\ [i]\}$
10:	$K_{raw} \leftarrow \{measured\_bits_R[i] \in matching\_indices\ \}$
11:	$K_{subset} \leftarrow first\ 20\%\ of\ K_{raw}$
12:	$H_S^{sub} \leftarrow SHA\text{-}256\ (K_{subset})$
13:	$H_R^{sub} \leftarrow SHA\text{-}256\ (K_{subset})$
14:	$\boldsymbol{if}\ H_S^{sub} = H_R^{sub}\ \boldsymbol{then}\ K_{quantum} \leftarrow K_{raw}\ \boldsymbol{else}\ "Abort"$
15:	$\boldsymbol{return}\ K_{quantum}$

Instead of depending on traditional pseudo-random number generators, the sender must produce a genuinely random bit sequence to act as the raw key in order to guarantee randomness. The sender builds quantum circuits in chunks using Qiskit's Aer simulator. This method guarantees computational viability by handling larger numbers of qubits efficiently, with each chunk operating on 16 qubits. Each qubit undergoes a uniform superposition operation created by the sender using Hadamard gates. In a similar manner, the sender determines how the qubits will be encoded and measured by selecting a random measurement basis for each qubit (computational '+' or diagonal 'x'). To get the random bits, the sender then uses measurement to collapse them. The resulting bit sequence is guaranteed to be genuinely random due to the intrinsic unpredictability of quantum measurement. Making use of the generated random bits and bases, the sender encodes each classical bit into a qubit as represented in table 3, by constructing a quantum circuit.

Table 3: Quantum state preparation process on bit values via basis and gate operations.

Bit	Basis	Gate Operations Applied	Final Qubit State
0	+	None	$\|0\rangle$
0	x	H	$\|+\rangle$
1	+	X	$\|1\rangle$
1	x	X → H	$\|-\rangle$

The ground state $|0\rangle$ for every qubit is where the encoding starts. A Pauli-X gate is used to flip the qubit to the $|1\rangle$ orthogonal state, thereby encoding the bit value, if the classical bit to be encoded is 1. If not, it stays in the orthogonal state of $|0\rangle$. A Hadamard gate is then used to rotate the qubit into the superposition state if the diagonal (x) basis is chosen as the basis for the qubit. This guarantees that the encoded bit is either 0 or 1 and that the qubit is prepared in either the $|+\rangle$ or $|-\rangle$ state, depending on the basis chosen. A string of encoded qubits is created during the encoding process by converting the classical information (bit and basis) into quantum states. The sender serializes each prepared qubit by logging the order of gate operations used during encoding in a structured JSON description, as this simulation does not use true quantum channels. The transmission of quantum states in a classical format over an unprotected network environment is then simulated by sending this abstract description of the quantum state over a typical TCP socket channel. The receiver uses the Aer simulator to independently generate random measurement bases after receiving the sender's description. The sender's flip and rotation sequence are not repeated by the recipient. Rather, the receiver measures each qubit to collapse it to a specific 0 or 1, records the measurement results as the receiver's candidate key bits, and applies the selected measurement basis directly to the received qubit states in chunks. This measurement procedure accurately mimics the entropy needed for raw key generation, particularly in bit values and basis selection, since it is genuinely random and unaffected by the sender's encoding operations. The receiver receives a complete sequence of measurement results by repeating this process for every chunk in batches. The sender and recipient publicly compare their selected bases following the measurement procedure. The raw quantum key (K_{raw}) is formed by keeping only the measurement results where their bases match. The security of this BB84 protocol simulation depends on the fact that any attempt to measure or alter this quantum circuit description in transit would introduce detectable errors, even though an eavesdropper could intercept it. The first 20% of (K_{raw}) is utilized as a key subset (K_{subset}) for verification in order to confirm the raw key's integrity and identify possible eavesdropping. A SHA-256 hash of this subset is calculated by both parties, and the results are then shared. If the computed hashes match, it confirms that the entire raw key is secure, and the remaining 80% of K_{raw} is accepted as the final quantum key ($K_{quantum}$). If the hashes do not match, it indicates a potential security breach, and the final key is aborted.

The system uses the CRYSTALS-Kyber-1024 variant for post-quantum key establishment, building on the previously established authenticated connection. As stated in Algorithm 3, our framework's kyber key exchange (KKE) module facilitates quantum-resilient key exchange between the sender and the recipient. The sender initiates the key exchange by generating a key pair, consisting of a public key pk_S and a secret key sk_S.

Algorithm 3 Kyber Key Exchange (KKE)

1:	$(pk_S, sk_S) \leftarrow generate\_keypair()$
2:	$Send\ (pk_S)\ to\ R$
3:	$(c, Ss_R) \leftarrow encapsulate(pk_S)$
4:	$Send\ (c)\ to\ S$
5:	$Ss_S \leftarrow decapsulate(c, sk_S)$
6:	$K_{kyber} \leftarrow Ss \leftarrow Ss_S = Ss_R$
7:	$\boldsymbol{return}\ K_{kyber}$

The sender transmits pk_S to the receiver, who uses pk_S to perform an encapsulation operation. This encapsulation generates a ciphertext c and a shared secret Ss_R, computed internally by the receiver. The ciphertext c is then sent back to the sender, who uses c and their secret key sk_S to perform a

decapsulation operation, resulting in their own shared secret Ss_S. The kyber algorithm ensures that $Ss_S = Ss_R$, establishing a common shared secret K_{kyber} known only to both parties. An adversary without knowledge of sk_S cannot recover the shared key from pk_S and c thanks to this one-round exchange, which is based on the hardness of the Module-Learning-With-Errors (MLWE) assumption and provides IND-CCA2 security, guaranteeing confidentiality and integrity even against adaptive chosen-ciphertext attacks. In order to prevent information leakage and guarantee post-quantum security, this shared secret is obtained using lattice-based operations that include noise injection and reconciliation.

Algorithm 4 Hybrid Key Derivation Function (HKDF)

1:	$K_{quantum} \leftarrow from\ QKD$
2:	$K_{kyber} \leftarrow from\ KKE$
3:	$K_{combined} \leftarrow K_{quantum} \,\|\|\, K_{kyber}$
4:	$salt \leftarrow a\ fixed, 32\ byte\ secure\ salt\ value$
5:	$parameters \leftarrow \{time_{cost} = 4, memory_{cost} = 102400, parallelism = 8, hash_{len} = 64\}$
6:	$K_{hybrid} \leftarrow Argon2id(K_{combined}, salt, parameters)$
7:	$\boldsymbol{return}\ K_{hybrid}$

Both the quantum key ($K_{quantum}$) and the kyber key (K_{kyber}) are sent to the key management system (KMS) after they are acquired. Using a hybrid key derivation function (HKDF) developed in Algorithm 4, the KMS is the primary component in charge of overseeing hybrid key generation. A single combined key ($K_{combined}$) is created by concatenating these two keys. To maintain security and guarantee deterministic behaviour, a fixed salt value is added. The Argon2id key derivation function is used to process the combined key and salt. To achieve a well-balanced trade-off between security and computational efficiency, the Argon2id function is configured with particular parameters that have been carefully chosen. a four-iteration time cost, boosting resistance to even quantum enabled brute-force attacks. Significant memory-hardness is provided by a memory cost of 102400 KB, which makes it challenging for attackers to use specialized hardware like GPUs or ASICs. By optimizing CPU core utilization, a parallelism level of 8 increases hashing efficiency. By avoiding collisions, the 64-byte hash length guarantees an output size that is adequate for cryptographic strength. Together, these parameters help create the hybrid key (K_{hybrid}), which successfully combines the computational security of post-quantum cryptography with the information-theoretic security of QKD. In our hybrid framework, this hybrid key can now be utilized for symmetric encryption in the data protection module and quantum secure communication.

In our hybrid framework, as shown in Algorithm 5, the quantum secure communication module starts by obtaining a 64-byte hybrid key (K_{hybrid}) from the key management system (KMS). In order to confirm the sender's and recipient's legitimacy, this key is first used in a timestamp-based mutual HMAC authentication procedure. The sender sends the receiver the current timestamp (T_1) to begin the authentication process. The receiver instantly determines if $T_1$is within the permitted time drift of ± 30 seconds. The protocol is terminated by the receiver if the timestamp T_1 is not current. Following successful validation, the receiver replies with its own timestamp (T_2) and an HMAC of T_1, which is calculated using the shared hybrid key. The sender makes two checks after receiving this response. The sender first confirms that T_2 is new (that is, within the permitted time drift); next, the sender confirms T_1's HMAC. The sender terminates the protocol if either of these checks is unsuccessful. A final HMAC check can be carried out by the recipient if the sender responds with an HMAC of T_2. The receiver

aborts if the final HMAC does not match. Authentication is deemed successful only if all HMACs match and timestamp validations are successful on both ends. This mutual authentication method ensures resistance to replay attacks while verifying that both parties have the same hybrid key and are time-synchronized.

Algorithm 5 Quantum Secure Communication Protocol

$T_{now} \leftarrow$ *current unix timestamp*
$K_{hybrid} \leftarrow 64$ *byte Hybrid Key from KMS*
$S \rightarrow R: T_1$
R *verifies* $|T_1 - T_{now}| \leq \pm 30$ *seconds* $\rightarrow$ *else abort*
$R \rightarrow S: HMAC\left(K_{hybrid}, T_1\right), T_2$
S *verifies* $|T_2 - T_{now}| \leq \pm 30$ *seconds* $\rightarrow$ *else abort*
S *verifies* $HMAC\left(K_{hybrid}, T_1\right)$ *matches* $\rightarrow$ *else abort*
$S \rightarrow R: HMAC\left(K_{hybrid}, T_2\right)$
R *verifies* $HMAC\left(K_{hybrid}, T_2\right)$ *matches* $\rightarrow$ *else abort*
$K_{ephemeral} \leftarrow Ephemeral\ Key\ Exchange(S, R)$
$K_{combined} \leftarrow K_{hybrid} \,||\, K_{ephemeral}$
$salt \leftarrow$ *a fixed*, 32 *byte secure salt value*
$parameters \leftarrow \{time_{cost} = 4, memory_{cost} = 102400, parallelism = 8, hash_{len} = 32\}$
$K_{session} \leftarrow Argon2id(K_{combined}, salt, parameters)$
$Ciphertext \leftarrow AES\text{-}GCM\text{-}256.Encrypt(K_{session}, Data, nonce)$
$Data \leftarrow AES\text{-}GCM\text{-}256.Decrypt(K_{session}, Ciphertext, nonce)$
return Secure Channel Established

After mutual authentication is successful, the Kyber algorithm is used to exchange ephemeral keys. Two security variants are supported by this exchange: Kyber512 for standard security and Kyber768 for advanced security. To guarantee strong forward secrecy, each ephemeral key ($K_{ephemeral}$) is generated specifically for a single session and then discarded right away. Lastly, a combined key ($K_{combined}$) is created by concatenating the hybrid key (K_{hybrid}) and ephemeral key ($K_{ephemeral}$). The final 32-byte session key ($K_{session}$) is obtained by passing this into the hybrid key derivation function (HKDF). This session key is used as input for Advanced Encryption Standard (AES) algorithm, the "De facto standard" for fast encryption of large amounts of data and is widely recognized by major authorities as quantum-safe. Utilizing 256-bit variant of AES in Galois/Counter Mode (GCM), for both encryption and decryption of all data exchanged across LAN or WAN to secure the overall communication.

By applying the Hybrid Key Derivation Function (HKDF) and concatenating the user's password with the hybrid key (K_{hybrid}), the encryption/decryption system in our hybrid framework obtains the symmetric key. To choose files, users must first register or log in. The files are then safely encrypted and kept, as seen in Figure 1. Only with the correct user password can decryption be accomplished; if the password is incorrect, decryption will fail. Long-term data at rest is protected by this method, which makes sure it's safe even if it's intercepted or accessed without permission.

7. Results and Discussions

The experimental outcomes of our software-based hybrid quantum-secure cryptosystem implementation are shown in this section. The evaluation of computational efficiency, key generation

and establishment rates, authentication mechanisms, and overall system performance in secure communication contexts was made possible by the controlled software-based environment in which all tests were carried out. Two laptops running Windows operating system made up the experimental testbed. The receiver had an AMD Ryzen 5 3450U processor with 7.89 GB of usable RAM, and the sender had an Intel Core i5 1235U processor with 15.3 GB of usable RAM. Both same-network and cross-network configurations were used for system-to-system communication. The average internet speed during testing was around 8 Mbps ($\pm$2 Mbps), regardless of the network configuration. This bandwidth is representative of a normal home broadband connection and provides a useful starting point for assessing how well the system performs in actual network scenarios.

By carrying out 1,000 authentication rounds between the sender and the recipient, the test results for pre-shared key (PSK) hash validation assess the authentication process's timing and performance benchmarks.

Table 4: PSK hash validation-based authentication benchmarks.

Entity	**Network Type**	**PSK Length (bytes)**	**Average Connection time (μs)**	**Average Authentication Processing time (μs)**	**Average Total Round Time (μs)**
Sender	Same Network	32	217.50	26.12	244.67
Receiver			201.59	7.88	213.27
Sender	Cross Network		238.64	32.72	276.84
Receiver			224.23	9.38	239.51

As detailed in Table 4, transitioning from same-network to cross-network environments increased average connection time, which measures the time needed to establish a network connection between the sender and receiver, by 9.7% for the sender and 11.2% for the receiver. This time includes the TCP handshake and any network-level delays during the connection setup phase. The computational overhead of the authentication mechanism, which is raised by 25.3% for the sender and 19.0% for the recipient, is reflected in the authentication processing time, which includes the time spent on the authentication logic itself, such as hashing the PSK, comparing the hashes, and validating the authentication request. While the receiver authenticated instantly upon hash matching, the sender's authentication time was typically longer because they had to wait for the receiver's authentication to be completed. Overall, the full authentication round trip grew by 13.2% on the sender side and by 12.3% on the receiver side.

The benchmarking between the sender and receiver involved 1,000 rounds with 300 simulated qubits to evaluate the performance of the BB84 protocol simulation. The aggregated key metrics are shown in Tables 5 and 6. The sender's bits-per-basis generation time, quantum transmission time, and basis comparison time all increased by 2.3%, 34.6%, and 22.4%, respectively, when switching from same-network to cross-network environments. While the overall protocol time increased by 9.3%, the sender's key generation throughput increased by 16.6%. Quantum state receiving time, measurement time, basis comparison time, key generation throughput, and overall protocol time all increased by 24.5%, 8.8%, 19.4%, and 19.6%, respectively, on the receiver side. However, these increases aside from network latency depend on the computational efficiency of generating quantum random bits and measurement bases using quantum circuits. This process includes the overhead of initializing quantum registers,

applying gate operations and preforming measurement using Aer simulator. Quantum state transmission time, on the other hand, is influenced primarily by network latency and captures the duration required to transmit the quantum state (i.e., circuit operations) from the sender to the receiver, including the overhead of data serialization and deserialization.

Table 5: Benchmarking sender-side BB84 quantum key establishment simulation.

Entity	**Network Type**	**Average Bits/Basis Gen Time (μs)**	**Average Quantum Transmission Time (μs)**	**Average Basis Comparison Time (μs)**	**Average Key Generation Throughput ($bits/\mu s$)**	**Average Total Protocol Time (μs)**
Sender	Same Network	713476.28	597.16	33.07	3.97	2772853.18
Sender	Cross Network	730116.81	803.76	40.49	4.63	3032039.52

Table 6: Benchmarking receiver-side BB84 quantum key establishment simulation.

Entity	**Network Type**	**Average Quantum State Receiving Time (μs)**	**Average Quantum State Measurement Time (μs)**	**Average Basis Comparison Time (μs)**	**Average Key Generation Throughput ($bits/\mu s$)**	**Average Total Protocol Time (μs)**
Receiver	Same Network	8409.37	1094627.97	29.84	3.31	1192704.43
Receiver	Cross Network	10468.70	1191467.76	35.64	4.32	1426241.22

Using the Kyber-1024 variant between the sender and the recipient, key-pair generation, encapsulation, and decapsulation times over 10,000 rounds each were measured in order to assess the computational efficiency and key establishment rates of the Kyber Key Exchange (KKE). For every iteration of the benchmark process, a new key pair was generated. The sender then decapsulated each ciphertext after the recipient benchmarked encapsulation across 10,000 randomly generated ciphertexts using the sender's public key.

Table 7: Average execution times of KKE operations using Kyber-1024.

Entity	**Network Type**	**Average Key Pair Generation time (μs)**	**Average Encapsulation time (μs)**	**Average Decapsulation time (μs)**	**Average Shared key Establishment time (μs)**
Sender	Same Network	150.50	-	472.25	1350.19
Receiver		-	251.30	-	1316.74
Sender	Cross Network	152.61	-	506.61	1642.77
Receiver		-	339.43	-	1626.18

In order to guarantee accurate performance metrics, the shared secret was verified against the original during this process, which measured the efficiency of encapsulation and decapsulation. Table 7 displays the averages that were obtained by averaging all timing samples across all iterations. According to the findings, switching to a cross-network environment increased the sender's key-pair generation time by 1.4%, the decapsulation time by 7.3%, and the overall shared-key establishment time by 21.7%. Both the shared-key establishment time and the receiver's encapsulation time increased by 23.5% and 35.1%, respectively. These percentage increases demonstrate how network conditions and processing power affect KKE performance.

Different output lengths were used to benchmark the hybrid key derivation function's (HKDF) performance. The procedure starts with a 32-byte kyber key and a 256-bit quantum key, which are concatenated and run through the HKDF. Prior to going through the HKDF, these two keys are concatenated. The core benchmark runs 1,000 timed iterations for each hash length (16, 32, 64 bytes), measuring the total elapsed time. We aggregated all timing samples as average values across all iterations and the resulting averages are presented in Table 8.

Table 8: Performance metrics of HKDF in hybrid key generation processes.

Hybrid Key Size (bytes)	**Hybrid Key Size (bits)**	**Hybrid Key Generation Time (μs)**
16-bytes	128-bits	77614.89
32-bytes	256-bits	78469.48
64-bytes	512-bits	79988.59

According to the findings, the key-generation time increased by 1.1% when the HKDF output size was increased from 16 bytes to 32 bytes and by another 1.9% when the output size was increased from 32 bytes to 64 bytes. Overall, there was about a 3.1% slowdown when going from 16 to 64 bytes. These modest increases imply that the system can manage larger key sizes with little degradation in performance, demonstrating that the HKDF process is computationally efficient and not significantly affected by changes in key size.

Connection establishment time, which measures the amount of time needed to establish a network connection between the sender and the recipient, together with the time needed for the TCP handshake and any network-level delays, mutual authentication using HMAC-SHA256, ephemeral key exchange with the Kyber-768 variant, and final 32-byte session key derivation time using HKDF, was used to assess the performance of the quantum-secure communication protocol.

Table 9: Timing benchmarks for final session key establishment in secure communication module.

Entity	**Network Type**	**Average Connection time (μs)**	**Average Authentication time (μs)**	**Average Ephemeral Key Exchange Time (μs)**	**Final Session Key Derivation Time (μs)**
Sender	Same Network	358.25	142.56	912.02	129748.28
Receiver		215.39	255.13	526.78	128640.70
Sender	Cross Network	373.76	160.80	1024.77	135521.01
Receiver		220.21	291.56	579.55	132708.26

Table 9 shows the average times for each phase based on an aggregate of more than 1000 iterations. The findings showed that switching to a cross-network configuration increased processing times by a moderate percentage. Connection time increased by 4.3%, authentication time by 12.8%, key exchange time by 12.4%, and session key derivation time by 4.4% for the sender. The receiver experienced a 2.2% increase in connection time, a 14.3% increase in authentication time, a 10.0% increase in key exchange time, and a 3.2% increase in session key derivation time. Since the receiver completes the final HMAC verification, as outlined in Algorithm 5, the mutual authentication time on the receiver side is typically longer. Since the sender decrypts the ciphertext to obtain the session key, while the recipient does so during encapsulation, the sender's ephemeral key exchange time is also usually longer.

Table 10: Performance metrics for secure data transmission between sender and receiver.

Network Type	**Original Data Size ($bytes$)**	**Average Data Encryption Time (μs)**	**Encrypted Data Size ($bytes$)**	**Average Data Transmission time (μs)**	**Average Data Decryption time (μs)**
Same Network	10485760 ~ 10 Mb	71482.48	18641400	764874.07	71185.75
	52428800 ~ 50 Mb	340388.37	93206800	13558543.83	332458.25
	104857600 ~ 100 Mb	714252.20	186413560	51581528.87	705429.30
Cross Network	10485760 ~ 10 Mb	80366.53	18641400	851783.23	79186.70
	52428800 ~ 50 Mb	359014.40	93206800	13628053.47	361850.70
	104857600 ~ 100 Mb	735369.87	186413560	53955467.73	718725.00

Table 10 describes the timing metrics for safe data transfer between sender and recipient over various network types. The system's processing times increase moderately when switching between same-network and cross-network environments. The average data encryption time, transmission time, and decryption time all rise by about 12.5%, 11.4%, and 11.4%, respectively, for a 10 MB file. The encryption time increases by approximately 5.5%, the transmission time by only 0.5%, and the decryption time by 8.8% for a 50 MB file. Encryption rises by 3.0%, transmission by 4.6%, and decryption by 1.9% at the 100 MB scale. These differences imply that although cross-network configurations do result in extra latency, especially during transmission phases, the overall effect stays within a reasonable range. Notably, the encryption and decryption times remain proportionally balanced between sender and receiver, indicating that the cryptographic load is well-distributed.

Table 11: Encryption and decryption speeds of large data files for Data-at-Rest security.

Original Data Size (GB)	**Average Data Encryption Time (ms)**	**Average Data Decryption time (ms)**
1	1530.32	1494.58
10	18749.22	18143.31
25	49876.21	44245.56
50	90867.62	88867.62
100	183762.54	180222.71

As shown in Table 11, the encryption and decryption times increase linearly with data size increases from 10 GB to 25 GB, 50 GB, and even 100 GB. With encryption and decryption times growing proportionately with data size, the system exhibits consistent performance characteristics.

The performance benchmarks mentioned above was computed after quick, unrecorded warm-up runs to stabilize any just-in-time optimizations, we utilized a high-resolution performance counter to record exact timestamps before and after each operation, converting the elapsed nanoseconds to microseconds for readability, and we turned off Python's garbage collector to prevent intermittent pauses.

The overall performance of this software-based hybrid quantum secure cryptosystem is significantly influenced by network conditions. Our simulations above results demonstrate how important network infrastructure is to efficacy and efficiency. Entities benefit from more reliable and consistent communication channels when they are on the same network, which lowers latency. The increase in transmission time during cross-network environments, however, is not due to inefficiencies in the cryptographic protocol itself, but rather to anticipated network latency. However, even under the more complicated cross-network configurations, the system showed resilience and retained good computational efficiency in terms of key establishment, hybrid key generation, and data transmission speeds. This implies that the system can manage the difficulties presented by various network circumstances while preserving its security and performance characteristics.

8. Conclusions

The study's conclusion emphasizes how urgently cryptographic systems must be modified for the upcoming quantum computing era, where conventional approaches will face existential challenges. The study shows how the mathematical underpinnings of existing encryption schemes could be compromised by sufficiently sophisticated quantum computers, making sensitive data susceptible to retroactive decryption. This work proposed a hybrid cryptographic framework to overcome this difficulty. The practical use of this research to protect data and communications from quantum threats, both short-term and long-term, is what makes it significant. The framework's simulation-based methodology, which generates the required entropy for quantum key generation by simulating quantum properties instead of real quantum channels or hardware, further emphasizes its practical implications. The system's overall security for safe communication and data protection is further improved by the authentication procedures used during key establishment and the forward secrecy attained through ephemeral key exchange. Nonetheless, the study admits a number of shortcomings: Performance benchmarks, especially in high-throughput settings, highlight possible bottlenecks in key generation and distribution processes. Therefore, future research should concentrate on maximizing the performance of hybrid systems through hardware acceleration and algorithmic enhancements. Lastly, to ensure interoperability across the world's digital infrastructure, post-quantum algorithms must be harmonized with current protocols through cooperative standardization efforts.

Funding source

None.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] Anant, L. Donchak, J. Kaplan, and H. Solle, “Risk Practice: The consumer-data opportunity and the privacy imperative,” New York, United States: McKinsey & Company, Feb. 2020. Accessed online on 2 Mar. 2025 at https://www.mckinsey.com/business-functions/risk-and-resilience/our-insights/the-consumer-data-opportunity-and-the-privacy-imperative

[2] M. Tarawneh, "Perspective Chapter: Cryptography – Recent Advances and Research Perspectives," IntechOpen, Dec. 2023. https://doi.org/10.5772/intechopen.111847

[3] J. Suo, L. Wang, S. Yang, W. Zheng, and J. Zhang, "Quantum algorithms for typical hard problems: a perspective of cryptanalysis," Quantum Information Processing, vol. 19, no. 178, pp. 1–24, Apr. 2020. https://doi.org/10.1007/s11128-020-02673-x

[4] A. Cintas Canto, M. Mozaffari Kermani, et al., "Algorithmic security is insufficient: A comprehensive survey on implementation attacks haunting post-quantum security," TechRxiv, May 2023. https://doi.org/10.36227/techrxiv.23071079.v1

[5] Gitonga, C. K. "The Impact of Quantum Computing on Cryptographic Systems: Urgency of Quantum-Resistant Algorithms and Practical Applications in Cryptography". European Journal of Information Technologies and Computer Science, vol. 5, no. 1, Jan. 2025. https://doi.org/10.24018/compute.2025.5.1.146

[6] S. M. Naser, "Cryptography: From the ancient history to now, its applications and a new complete numerical model," International Journal of Mathematics and Statistics Studies, vol. 9, no. 3, pp. 11–30, 2021. Accessed online on 5 Mar. 2025 at https://eajournals.org/ijmss/vol-9-issue-3-2021/cryptography-from-the-ancient-history-to-now-its-applications-and-a-new-complete-numerical-model/

[7] S. Duggal, V. Mohindru, P. Vadiya, and S. Sharma, "A comparative analysis of private key cryptography algorithms: DES, AES and Triple DES," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, no. 6, pp. 1373–1376, June 2014. Accessed online on 5 Mar. 2025 at https://www.researchgate.net/publication/265166061

[8] M. E. Hellman, "An overview of public key cryptography," IEEE Communications Magazine, vol. 40, no. 5, pp. 42–49, May 2002. https://doi.org/10.1109/MCOM.2002.1006971

[9] D. Pointcheval, "Computational security for cryptography," Oct. 2009. Accessed online on 6 Mar. 2025 at https://api.semanticscholar.org/CorpusID:14413913

[10] V. Gheorghiu and M. Mosca, "Benchmarking the quantum cryptanalysis of symmetric, public-key and hash-based cryptographic schemes," arXiv preprint arXiv:1902.02332, Feb. 2019. Accessed online on 6 Mar. 2025 at https://arxiv.org/abs/1902.02332

[11] G. Mone, "The quantum threat," Communications of the ACM, vol. 63, no. 7, pp. 12–14, July 2020. https://doi.org/10.1145/3398388

[12] H. A. Bhat, B. K. Kaushik, F. A. Khanday, F. Bashir, and K. A. Shah, "Quantum computing: Fundamentals, implementations and applications," IEEE Open Journal of Nanotechnology, vol. 3, pp. 1–17, June 2022. https://doi.org/10.1109/OJNANO.2022.3178545

[13] M. Zeeshan, S. Anayat, R. G. Hussain, and N. Rehman, "Processing power of quantum computer," International Journal of Scientific & Engineering Research, vol. 7, no. 8, Aug. 2016. Accessed online on 7 Mar. 2025 at https://www.researchgate.net/publication/308298584

[14] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," Proceedings 35th Annual Symposium on Foundations of Computer Science, pp. 124–134, Nov. 1994. https://doi.org/10.1109/SFCS.1994.365700

[15] Li, K., Yan, P.-G., & Cai, Q.-Y. (2020). Quantum computing and the security of public key cryptography. Fundamental Research, https://doi.org/10.1016/j.fmre.2020.12.001

[16] L. K. Grover, “A fast quantum mechanical algorithm for database search,” Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC), pp. 212–219, May 1996. https://doi.org/10.1145/237814.237866

[17] Nunnaguppala, L. S. C., Sayyaparaju, K. K., & Padamati, J. R. (2022, October). The impact of quantum computing on cybersecurity: Anticipation and countermeasures. International Journal for Innovative Engineering and Management Research, 11(10), 183–195. https://doi.org/10.48047/IJIEMR/V11/ISSUE10/21

[18] G. Brassard, P. Høyer, and A. Tapp, “Quantum algorithm for the collision problem,” in Encyclopedia of Algorithms, M. Y. Kao, Ed. Berlin, Germany: Springer, 2015, pp. 998–999. https://doi.org/10.1007/978-3-642-27848-8_304-2

[19] A. Hosoyamada and Y. Sasaki, “Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound,” Cryptology ePrint Archive, Paper 2020/213, 2020. Accessed online on 7 Mar. 2025 at https://eprint.iacr.org/2020/213

[20] C. H. Bennett and G. Brassard, “Quantum cryptography: Public key distribution and coin tossing,” Theoretical Computer Science, vol. 560, Jan. 1984. https://doi.org/10.1016/j.tcs.2011.08.039

[21] S. K. Reddy and B. C. Mohan, “Comprehensive analysis of BB84, a quantum key distribution protocol,” arXiv, Dec. 2023. Available: https://inspirehep.net/literature/2734849.

[22] K. Dajani, R. Owor, and Z. Okonkwo, “The relevance of quantum cryptography in modern networking systems,” Neural, Parallel, and Scientific Computations, vol. 18, Available: https://www.academia.edu/49718363/The_relevance_of_quantum_cryptography_in_modern_networking_systems.

[23] L. Bouchoucha, S. Berrah, and M. Sellami, "Influence of experimental parameters inherent to optical fibers on Quantum Key Distribution, the protocol BB84," Semiconductor Physics, Quantum Electronics & Optoelectronics, vol. 21, no. 1, 2018. DOI: https://doi.org/10.15407/spqeo21.01.073

[24] E. Kiktenko, A. Trushechkin, and A. K. Fedorov, "Symmetric Blind Information Reconciliation and Hash-function-based Verification for Quantum Key Distribution," Lobachevskii Journal of Mathematics, vol. 39, no. 7, pp. 877–883, May 2017. https://doi.org/10.1134/S1995080218070107

[25] N. Lütkenhaus, B. C. Sanders, M. B. Plenio, and W. Tittel, "Limitations on Practical Quantum Cryptography," Physical Review Letters, vol. 85, no. 7, pp. 1330–1333, Sep. 2000. DOI: https://doi.org/10.1103/PhysRevLett.85.1330

[26] D. Moody, R. Perlner, A. Regenscheid, A. Robinson, and D. Cooper, "Transition to Post-Quantum Cryptography Standards: Initial Public Draft," NIST Internal Report 8547, Nov. 2024. Available: https://doi.org/10.6028/NIST.IR.8547.ipd.

[27] "Module-Lattice-Based Key-Encapsulation Mechanism Standard," FIPS 203, National Institute of Standards and Technology, Gaithersburg, MD, Aug. 13, 2024. Available: https://doi.org/10.6028/NIST.FIPS.203.

[28] "Module-Lattice-Based Digital Signature Standard," FIPS 204, National Institute of Standards and Technology, Gaithersburg, MD, Available: https://doi.org/10.6028/NIST.FIPS.204.

[29] "Stateless Hash-Based Digital Signature Standard," *FIPS 205*, National Institute of Standards and Technology, Gaithersburg, MD, Available: https://doi.org/10.6028/NIST.FIPS.205.

[30] Status Report on the Fourth Round of the NIST Post-Quantum Cryptography Standardization Process, NIST IR 8545, March 2025. Available: https://doi.org/10.6028/NIST.IR.8545.

[31] Manish Kumar, Post-quantum cryptography algorithm's standardization and performance analysis, Array, vol. 15, p. 100242, 2022. DOI: https://doi.org/10.1016/j.array.2022.100242

[32] Wilfred W. K. Lin, Challenges of post-quantum cryptography, April 2023. Available at: https://www.researchgate.net/publication/370049812

[33] Khondokar Fida Hasan, Mir Ali Rezazadeh Baee (Senior Member, IEEE), Leonie Simpson (Senior Member, IEEE), Chadni Islam, Ziaur Rahman, Warren Armstrong, Praveen Gauravaram, and Matthew McKague, "A Framework for Migrating to Post-Quantum Cryptography: Security Dependency Analysis and Case Studies," IEEE Access, vol. 12, pp. 19193–19211, Jan. 2024. DOI: 10.1109/ACCESS.2024.3360412.

[34] William Barker, William Polk, and Murugiah Souppaya, Getting Ready for Post-Quantum Cryptography: Exploring Challenges Associated with Adopting and Using Post-Quantum Cryptographic Algorithms, NIST Cybersecurity White Paper, Gaithersburg, MD, Apr. 28, 2021. Available: https://doi.org/10.6028/NIST.CSWP.04282021

[35] N. Aquina, S. Rommel and I. T. Monroy, "Quantum secure communication using hybrid post-quantum cryptography and quantum key distribution," 2024 24th International Conference on Transparent Optical Networks (ICTON), Italy, 2024, Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/ICTON62926.2024.10648124

[36] Ricci, S., Dobias, P., Malina, L., Hajny, J., & Jedlicka, P. (2023). *Hybrid keys in practice: Combining classical, quantum and post-quantum cryptography*. IEEE Access. https://doi.org/10.1109/ACCESS.2024.3364520

[37] Petcher, A., & Campagna, M. (n.d.). Security of hybrid key establishment using concatenation. Available: https://eprint.iacr.org/2023/972.pdf

[38] Bindel, N., Brendel, J., Fischlin, M., Goncalves, B., & Stebila, D. (2019). Hybrid key encapsulation mechanisms and authenticated key exchange. Post-Quantum Cryptography: 10th International Conference, Chongqing, China, Springer. https://doi.org/10.1007/978-3-030-25510-7_12

[39] Garcia, C. R., Aguilera, A. C., Vegas Olmos, J. J., Monroy, I. T., & Rommel, S. (2024). Quantum-Resistant TLS 1.3: A Hybrid Solution Combining Classical, Quantum and Post-Quantum Cryptography. In 2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD) (pp. 246–251). IEEE. https://doi.org/10.1109/CAMAD59638.2023.10478407