QuCloud: Enhancing Cloud Storage Security by combining Quantum Key Distribution, Post-Quantum Cryptography, and Custom Proxy Re-Encryption

Anika Taffannum Zarin[a,†], Omar Radee[a,†], Md Shawmoon Azad[a,c,d], and M. R. C. Mahdy[a,b,*]

[a]Department of Electrical and Computer Engineering, North South University, Bashundhara, Dhaka, 1229, Bangladesh
[b]NSU Center of Quantum Computing, North South University, Bashundhara, Dhaka, 1229, Bangladesh
[c]Department of Computer Science, Cleveland State University, Cleveland, Ohio, 44115, USA
[d]Mahdy Research Academy, Dhaka, 1229, Bangladesh

Abstract

The rapid advance of quantum computing threatens classical cryptography, especially the RSA- and ECC-based schemes that protect today's cloud storage. We present a hybrid framework combining entanglement-based E91 QKD, SHA-256 salt hashing, a custom proxy re-encryption(PRE) layer, and lattice-based Kyber PKE to achieve end-to-end quantum-resistant security. Simulations showed our E91 implementation achieved $\approx 22.45\%$(average) key-generation efficiency. The salt-hashing mechanism exhibited a 98% avalanche effect in response to single-bit changes. A novel feature is our QKD-derived PRE layer, which adds only ≈ 0.23 s of overhead while eliminating harvest-now, decrypt-later vulnerabilities. Performance metrics include optimal ciphertext entropy ($H \approx 8.00$), 50% Hamming diffusion, and fast processing times (AES-GCM encryption for a 10 MB file takes 55 ms; 45.78 s for the complete eight-step pipeline). Compared to existing E91 implementations (18–22% efficiency), our framework achieves comparable key-generation rates while maintaining runtime within 1.4× of standalone Kyber plus AES-GCM implementations.

Keywords: Quantum Key Distribution; Post Quantum Cryptography; E-91 Protocol; Proxy Re-encryption; Cloud Storage

1 Introduction

Classical computers of today run on bits, which is the smallest unit of digital information, to process and store all of the digital data. Quantum computers, however, uses qubits[1], which unlike bits, are not limited to a binary value of only 1 or 0[2]. Qubits, the basic unit of information for quantum computers, instead relies on the principle of superposition[3], which allows the qubit state to be in a combination of $|0\rangle$ and $|1\rangle$[3]. As qubits can have an infinite and continuous number of possible states[4], this leads to the potential to increase the computational power of quantum computers exponentially with respect to the number of qubits[5]. With the advent of more quantum computers today, specially with larger number of qubits, it exposes threat to the security of contemporary cryptographic methods of today, including public-key cryptography which depends on the hardness problem of factorising very large prime numbers. Shor's algorithm [2] provides a quantum computer with a method to break asymmetric cryptographic schemes, when there are enough number of qubits.[6] In today's world, a large chunk of data, amounting to around 100 zettabytes this year, is expected to be stored in cloud storage[7]. These data are protected by various cryptographic schemes, including the RSA(Rivest Shamir and Adleman), which has been shown to fail against quantum computing in polynomial

[1]†These authors contributed equally to this work.
[2]*Corresponding author: mahdy.chowdhury@northsouth.edu (M.R.C. Mahdy).
shawmoon.azad@northsouth.edu (Md Shawmoon Azad).

time[8]. Cloud storages hold information from personal data, to business enterprises records to government official data. Cloud storages also use protocols such as TLS, in order to establish a secure communication channel between a server and client. During the key exchange process of this protocol which also uses RSA for session establishment, an eavesdropper Eve can intercept the key exchange, store the encrypted data now, and decrypt it later in the future once a sufficiently powerful quantum computer is available[9].

In the existing cloud deployments, the confidentiality typically relies on fast symmetric encryption for bulk data, however the key-management and session establishment have historically relied on public-key primitives, including RSA-based mechanisms in legacy stacks. In particular, older TLS cipher suites use RSA key transport, which does not provide forward secrecy; a passive adversary can record encrypted sessions today and, once a cryptographically relevant quantum computer becomes available, retrospectively recover the corresponding private key to decrypt previously captured traffic (i.e., the "harvest-now, decrypt-later" threat). Notably, RSA key exchange has been deprecated in modern TLS guidance and is not supported in TLS 1.3, reflecting the long-recognized risk of retrospective decryption when long-term keys are compromised.

Beyond data-in-transit, cloud storage commonly uses *envelope encryption*, where a per-object data encryption key (DEK) encrypts the file, and the DEK is then stored in wrapped form under a key encryption key (KEK) managed by a cloud KMS. If an attacker harvests encrypted blobs together with wrapped keys (and the protection of those wrapping keys depends on RSA/ECC in legacy integrations or long-lived key-wrapping workflows), the ciphertext may remain confidential only until large-scale quantum computation becomes feasible. This threat is especially critical for long-retention cloud data (e.g., medical, legal, governmental, or proprietary archives), which motivates proactive deployment of quantum-safe mechanisms and crypto-agile migration strategies highlighted by NIST as urgent under the harvest-now, decrypt-later model.

In order to tackle the growing data security risks in cloud storage, this paper seeks to explore the effectiveness of integrating Post-Quantum Cryptography and Quantum-Key Distribution(QKD) to provide additional security for both data in transit and key exchange. Quantum Key Distribution(QKD) is derived from the concepts of quantum physics[3], and enables the exchange of secret keys used for cryptography. Based on the no-cloning theorem[10], and the uncertainty principle[11], QKD has the added security measure in case of an attacker, Eve, eavesdropping on the keys during exchange. In line with QKD protocol, when Alice(sender) sends Bob(receiver) a message, it can prevent the eavesdropping of Eve during the key generation process, because of the non-cloning principle and uncertainty principle. As QKD is based on quantum particles for the transmission of keys, measuring a quantum state by interception will disturb its state according to Heisenberg's uncertainty principle[12], which will make Alice and Bob aware of this interception due to the new errors in the key introduced by Eve's measurement. Post-Quantum Cryptography(PQC) development has been greatly accelerated by the push of NIST(National Institute of Standard Technologies) to develop quantum-resistant algorithms before the arrival of cryptographically relevant quantum computers[13]. Post Quantum Cryptography is shown to mitigate the current risks of large prime number factorization which relies on the ease of multiplying two prime numbers and on the difficulty of factorizing them. PQC instead is based on harder math problems that would be difficult for both classical and quantum computers to crack. These algorithms are primarily designed to run on existing classical computers while also being resistant to future quantum attacks, due to the nature of Shortest Vector Problem(SVP) and Module Learning with Errors(MLWE) problem[14]. Among the PQC algorithm that have been proposed for post-quantum scheme Kyber, which is a lattice-based cryptographic scheme, is one of the standardized algorithms which work with public key encryption[15]. Proxy re-encryption and salt hashing, are shown to improve security and efficient data sharing in cloud environments[16]. The E91 QKD protocol[17] can be used to securely generate and distribute symmetric keys while detecting any eavesdropping attempts. These keys are then further enhanced through salt hashing to ensure robust protection against any brute force and dictionary attacks. Furthermore, proxy re-encryption is introduced to facilitate secure sharing of encrypted files between users without the need to expose sensitive data or encryption keys to the cloud.

This innovative hybrid framework aims to advance the field of cloud storage security by providing an end-to-end encryption solution resilient to both classical and quantum attacks. Our main contributions can be summarized as follows:

1. **A Novel Hybrid Security Architecture:** We introduce QuCloud, a multi-layered framework that secures cloud storage by uniquely integrating entanglement-based E91 QKD, the NIST-standardized Kyber PKE algorithm, and SHA-256 salted hashing. This model is explicitly designed to provide

end-to-end, quantum-resistant security and mitigate "harvest-now, decrypt-later" threats.

2. **Highly Efficient, QKD-Derived PRE:** We present a custom PRE layer that leverages a 128-bit slice of the ephemeral E91 raw key. This novel method avoids the security risks and computational cost of conventional PRE key derivation, adding a minimal overhead of only ≈ 0.23 seconds for re-encryption.

3. **Robust and Verifiable Eavesdropping Detection**: The framework's security is guaranteed by an integrated CHSH inequality test that validates the integrity of the quantum channel. In testing, the presence of an eavesdropper caused the CHSH value to drop by 62.6%, triggering an immediate protocol abort and demonstrating a highly effective, tamper-evident key exchange.

4. **Comprehensive Performance and Viability Benchmarking:** We validate the entire cryptographic pipeline, proving its practical efficiency. The system achieves a high average key-generation rate of 22.45% , near-perfect ciphertext entropy $H \approx 8.00$, and a total execution time of just 45.78 seconds for a 10 MB file. These results demonstrate that the framework's runtime is within 1.4x of simpler, non-PRE implementations, confirming its viability for real-world deployment.

Our paper is structured as follows: This paper first reviews foundational research in Quantum Key Distribution, Post-Quantum Cryptography, and PRE. We then introduce our hybrid security framework, which integrates E-91 QKD, Kyber-PKE, and cloud PRE to secure cloud storage. We detail the system's practical implementation using IBM Quantum and present experimental findings on its effectiveness, including CHSH inequality test results, key generation efficiency, and encryption performance benchmarks. The discussion assesses the framework's quantum resistance, security performance, and practical limitations. We conclude by summarizing our contributions and outlining future research directions, such as multi-cloud quantum security and FPGA-based cryptographic acceleration.

2 Related Works

2.1 Proxy Re-Encryption

Many quantum protocols and Proxy Re-Encryption in the cloud system have been studied and developed for security purposes. PRE is a cryptographic approach designed to facilitate secure data sharing easier by re-encrypting the data again[18]. Existing PRE Protocols include Secret Sharing Schemes (SSS), Multi-secret sharing schemes(MSS)[19], Unidirectional, and Bidirectional(ciphertext can be transferred in both directions between two parties)[20], Multi-Use (where proxy can re-encrypt the same Ciphertext numerous times for different parties)[21], Conditional PRE(Ciphertexts can only be re-encrypted if they meet specific conditions, such as a keyword match or attribute-based criteria)[22]. In classical PRE, the sender encrypts her data and wants to share it with the receiver. To facilitate this, the sender utilizes a trusted intermediary called a proxy. The sender provides the proxy with a key, which the proxy uses to create a conversion key and transform the encrypted data into a version that Bob can decrypt later. The important thing is during this procedure, a third party(proxy) never sees the actual data and can never obtain any information about the plain text, keeping it secure[18]. The re-encryption key is mathematically derived using private and public keys. PRE relies on complicated mathematical assumptions like the hardness of lattice problems[23], Discrete Logarithmic Problems (DLP)[24], or bilinear pairings[25].In existing papers, cloud storage re-encryption, key-policy attribute-based encryption (KP-ABE), and PRE have been used to transfer data securely.

2.2 Limitations and Challenges of PRE in Cloud Systems

The re-encryption of data was only limited to someone who was accessing the cloud. So, the appropriate collection of secret keys will make the decryption happen. However, this process was tedious, which compromised security[26]. Later, a modification of the scheme was proposed, replacing the cloud service provider with a trusted third party, but this will cause a reliance on trust assumption[27]. On the other hand, in quorum-based PRE scheme, the data owner exploits the threshold secret sharing scheme to divide the key into N shares. These shares are distributed to the trusted proxies to perform re-encryption. Although this scheme has advantages, it might be vulnerable because the private key gets split and then distributed to the

proxy[28].Many parametric attack models have been proposed for PRE where oracles are available.[29].The implementations of PRE introduce significant challenges. Hence, it is pertinent to consider increased complexities in key management, ownership, and control in the cloud.[30].

2.3 Shifting from classical to quantum

All these data encryption, re-encryption, and key generation heavily rely on the classical channel. Shor's algorithm hypothetically proved that it is capable of factoring large integers. This algorithm includes using superposition and entanglement to explore all possible solutions simultaneously using quantum period finding. It is challenging to do the same thing classically because prime factors can be very large. Hence, quantum computers with sufficient qubits are required to break classical encryption algorithms. Classical algorithms like RSA, ECC, and DLP are at risk. Peter Shor emphasized developing quantum-resistant protocols[31]. Fortunately, the first quantum protocol was proposed in 1984, authored by Charles H. Bennett and Gilles Brassard. The BB84 protocol is a prepare-and-measure-based protocol where the sender and receiver use random bits and bases to encode qubits. For generating a shared symmetric key, bases are discussed over a classical channel. Later, B-92 and Ekerts-91 protocols were developed.

2.4 Post-Quantum Cryptography (PQC)

Post-Quantum Cryptography(PQC) is the set of cryptographic algorithms that are designed to resist known quantum attacks from quantum computers of the future that have a sufficiently large number of qubits, which in turn can break currently used public-key encryption schemes. To withstand quantum threats, cloud service providers are also considering both PQC and QKD. In August 2023, NIST provided a few drafts featuring Kyber, Dilithium, Falcon, and Sphincs[32]. Here, Kyber is the only standardized and succeeded against the threat of Quantum attacks. It relies on the hardness of the lattice-based problem[33]. However, our work uses both QKD and PQC to share data in cloud storage without compromising security and efficiency. Quantum computing is evolving rapidly, with leading cloud service providers (CSPs) like AWS, IBM, Google Cloud, and Azure offer quantum services[34, 35, 36, 37, 38]. On the other hand, these environments face constraints due to the limited number of qubits in single quantum computers, restricting their capability to handle larger, real-world quantum workloads. Current studies focus on single-CSP setups for hybrid algorithms and QKD applications but lack mechanisms to scale across multi-cloud infrastructures. Security concerns around distributed quantum computation remain unaddressed[39]. A well-known quantum cryptographic protocol ensures secure communication between CSPs. Using QKD, the sender and receiver can detect eavesdropping attempts. QKD uses qubits in superposition states, such as $|0\rangle$, $|1\rangle$, $|+\rangle$, and $|-\rangle$, encoded with operations like Pauli-X and Z measurements[40]. The correctness of the shared key is evaluated using joint measurements to detect any discrepancies. Large quantum calculations are distributed across Cloud Service Providers (CSPs). Each fragment is encrypted, transmitted, and then recombined. Error rates can be minimized using simple repetition and Shor error-correcting codes[41]. These methods are thoroughly tested on different quantum hardware and simulators. Distributed quantum calculations were 88.1% correct without error correction and 96.8% accurate when error correction was applied. Authorization detection using QKD showed 98.8% success rates in identifying unauthorized access attempts [42]. Logarithmic and linear time complexity bounds were observed for the implemented algorithms, indicating scalability[43]. Multi-cloud distribution addresses computational bottlenecks but does not fully resolve the limitations of current quantum hardware with small qubit capacities[42]. Moreover, increased communication times across CSPs can offset computational gains[44]. As quantum computers are being developed with more qubits, if enough qubits quantum computer is available, it can use Shor's Algorithm and make the current public key cryptography schemes to be obsolete, thus leading to all the digital data of today at risk. [45]Hence, the digital data of today needs to be safeguarded using Post-Quantum Cryptography(PQC) algorithms which can resist attacks from future quantum computers. Peter Shor developed a quantum algorithm that has an exponential speed advantage over classical algorithms for tasks like solving discrete logarithms and factoring of integers.[45] So in theory, it means that a quantum computer with enough qubits will be able to run Shor's algorithm and break most of the public key cryptosystems currently in use. Before a large and fault-tolerant(LFT) quantum computer is built, a few standardization bodies are working on the standardization processes of PQC. National Institute of Standard Technologies(NIST) has been initiating the process of standardizing the

public key PQC algorithms. One such post-quantum cryptographic key encapsulation mechanism (KEM) is CRYSTALS-Kyber. It is built on the concept of the hardness problems associated with Module Learning With Errors(Module LWE) problem. It implements a CPA-secure Public-Key Encryption (PKE) scheme, which is eventually transformed into a CCA-secure KEM using a specific variant of the Fujisaki-Okamoto transform. The IND-CPA Kyber PKE algorithm is also a form of public-key encryption(PKE) that is designed for post-quantum secure communication using asymmetric cryptography. [46]

Several works have integrated QKD with proxy re-encryption in cloud systems, but most of them rely on classical channels for re-encryption keys, making them vulnerable to quantum threats once a large-scale quantum computer exists. Other researchers have proposed post-quantum schemes (e.g., `Kyber`, `Dilithium`), but practical adoption in cloud-based multi-user environments remains limited. Recent papers on hybrid QKD-PQC solutions (e.g., [47]) demonstrate how we can merge these cryptographic layers to achieve both forward secrecy and post-compromise security. Our work builds on these approaches but focuses specifically on cloud storage scenarios, introducing a custom partial-key scheme that preserves efficiency in re-encryption workflows.

2.5 Salted Hashing

Salted Hashing, in particular SHA-256(Secure Hash Algorithm 256-bit), has been shown to perform well in formulating the digital signatures of RSA by ensuring that data integrity is maintained [48]. While its most notable applications are on technologies like blockchain, where it is used to secure cryptocurrency transactions, SHA-256 can also be used for cloud storage security[49]. SHA-256 algorithm allows for both a reliable level of security and relatively fast computation[50]. By carrying out a transformation on data into a fixed-size hash value, it is able to protect data integrity, and its 256-bit output size ensures that there are enough hash values to make any brute force attack less likely to succeed.[51]

2.6 Hybrid quantum-classical security frameworks

Rahimi Ghashghaei et al.[52] replace post-quantum encryption with a classical scheme and propose post-quantum QKD with authenticated encryption. They compare Kyber and Dilithium(signatures) against RSA-alike baselines with a BB84-alike QKD setting. This line of work treats PQC as a practical upgrade path for the classical components on which QKD still depends (authentication, integrity, key confirmation), rather than changing the application layer itself. Garms et al. move beyond simulation by demonstrating an experimental integration of QKD and PQC in a hybrid quantum-safe cryptosystem. Along with implementing fundamental cryptographic operations on hardware (FPGA), including device-root features (such as PUF-based considerations), they also discuss forward security/post-compromise goals and emphasize deployability over protocol design. By showing an experimental integration of QKD and PQC in a hybrid quantum-safe cryptosystem, Garms et al. [53] go beyond simulation. To maintain security even if one component fails, their design employs a hybrid authenticated key exchange ("Muckle++"), in which session keys are generated by combining multiple sources (QKD, classical, and PQC components). Additionally, they implement cryptographic operations on hardware (FPGA), including device-root-of-trust features, and discuss forward security/post-compromise-style goals. They place more emphasis on deployability than protocol design. QSAC (Rifat et al., 2025) proposes a secure audio communication encryption-based pipeline at the simulation level by encrypting steganographic audio with the classical encryption algorithm ChaCha20-Poly1305 (AEAD), after generating a shared key via E91 QKD and hashing it with SHA-3 to obtain a fixed-length shared symmetric key. They validate robustness using classical security metrics (entropy/UACI/NSCR) and also report CHSH-based checks for eavesdropping sensitivity. QSAC (Rifat et al. (2025) [54]) proposes a secure audio communication encryption-based pipeline at the simulation level by encrypting steganographic audio with the classical encryption algorithm ChaCha20-Poly1305 (AEAD), after generating a shared key via E91 QKD and hashing it with SHA-3 to obtain a fixed-length shared symmetric key. They validate robustness using classical security metrics (entropy/UACI/NSCR) and also report CHSH-based checks for eavesdropping sensitivity.

Another closely related multi-layered hybrid quantum–classical design is reported by Sykot *et al.* [55], who integrates E91 QKD for shared-key establishment and then apply SHA-based hashing to obtain a

fixed-length high-entropy key, and use AES to encrypt steganographic image payloads, complemented by deep learning-based steganography for further concealment. Their experiments on images of varying resolutions (64×64 to 512×512) report an average encrypted-image entropy of 7.9929, with differential-attack indicators NPCR = 99.6928% and UACI = 56.1549%. Even though this approach does validate the integration of QKD-derived key material with classical symmetric encryption and hashing, but it is primarily designed for steganographic image transmission and does not address end-to-end cloud storage hardening. On the other hand, QuCloud is designed with the goal of a cloud-storage workflow and also incorporates Kyber-based post-quantum key transport and a PRE layer for controlled delegation.

2.7 Novel contributions to the existing literature gap

Our cloud-based hybrid quantum-classical framework is designed with the goal of securing cloud data and access by combining E91 QKD, hashed key material, and a PRE-based delegation layer. Compared to the aforementioned hybrid quantum-classical security methods, which primarily emphasized the classical-channel and application-layer secure transmission, such as QSAC, our approach is distinct as it offers ciphertext-level access control and re-encryption cloud-sharing workflows. It also explicitly addresses the proxy trust assumption and real cloud threats, including secured delegation and unauthorized re-sharing.

Despite rapid progress in hybrid quantum–classical security, three specific gaps remain unaddressed by prior work:

1. **End-to-end cloud-storage hardening combining entanglement-based QKD with post-quantum encryption.** Existing hybrid frameworks either adopt prepare-and-measure QKD [47] or omit PQC entirely in favour of classical RSA/ECC. Our design integrates Ekert-91 QKD, Kyber-512 KEM, and SHA-256 salt hashing into a practical cloud pipeline.

2. **A PRE layer re-using QKD-derived key material.** Classical PRE schemes derive the re-encryption key from long-lived public/secret keys, leaving them vulnerable to "harvest-now, decrypt-later" attacks. We carve a 128-bit slice of the fresh E91 raw key to serve as the PRE key, eliminating extra lattice operations and reducing proxy latency to 0.23 s (cf. 1.8 s in [16]).

3. **Comprehensive performance benchmarking.** Beyond key-rate and Bell tests, we evaluate throughput and overheads across the full hybrid workflow: raw-key efficiency (**22.45**%), CHSH drop (62.6%), ciphertext entropy ($H \approx 8.00$), 50 % Hamming diffusion, and end-to-end pipeline execution in 45.78 s for a 10 MB object.

These contributions demonstrate a **practically deployable, quantum-resistant cloud-storage workflow** that aligns with or outperforms recent hybrids on key-rate while maintaining runtime within a factor of 1.4 of standalone Kyber + AES baselines.

3 Proposed Architecture

Our proposed architecture uses advanced and multiple-layered cryptographic techniques that blend both Quantum and Post Quantum Cryptography. QKD E-91 protocol generates a shared symmetric key and uses salt hash(SHA-256) to expand the key and have it at a fixed length. Salt hashing includes some extra parameters generated by sender to have more randomness in the key. The sender informs Cloud and the receiver of some partial E-91 keys of different orders to re-encrypt. Cloud data and files are encrypted using a salt-hash key with Advanced Encryption Standard(AES), and then they are re-encrypted by a partial key by a cloud proxy using AES. Salt hashing is used to avoid deterministic keys and transferred to Bob using Kyber Public Key Encryption(Kyber-PKE).In Figure 1 we give a very concise overview of our proposed architecture.

3.1 Quantum Key Distribution using E-91

3.1.1 Initialization

Charlie initiates the E91 QKD protocol by preparing a maximally entangled qubit pair in the Bell state $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, as illustrated in Figure 2. This entangled state serves as the foundation for the

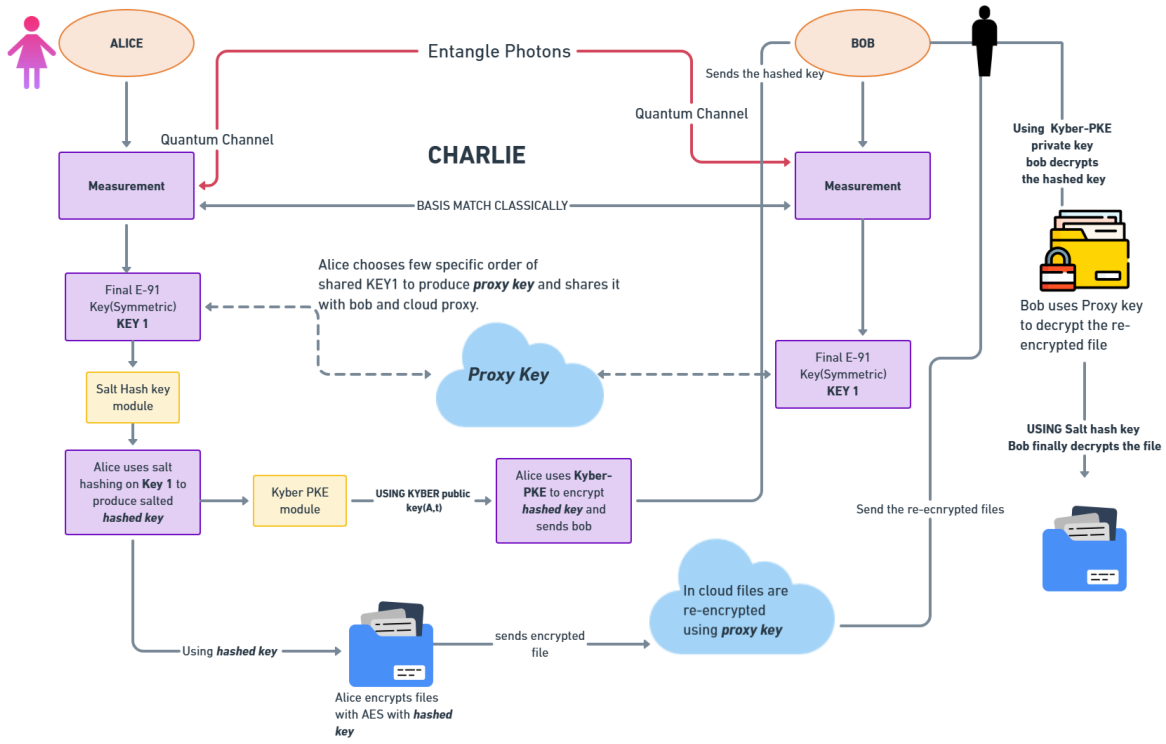


Figure 1: A schematic diagram of the overall protocol. At first, Charlie(a trusted quantum source)provides entangled photons to Alice and Bob, and then Alice and Bob use their bases to perform measurements. Once measurements are complete, they share their bases in the classical channel to generate a shared secret key. Alice uses a subset of keys for the cloud to facilitate re-encryption and also shares her bases with Bob, enabling him to compute the re-encryption key. Furthermore, Alice applies salted hashing using her unique parameter to generate a salt key and encrypt data with the key, forming the first encryption layer. Alice uses Kyber PKE, leveraging quantum-resistant properties to send the salted key to Bob. Here, data gets encrypted and decrypted through this two-tier encryption and decryption mechanism.

entire protocol. To generate this state, Charlie begins with a qubit in the $|0\rangle$ state and sequentially applies a Hadamard gate (H) followed by a controlled-NOT gate ($CNOT$), where the first qubit controls the operation on the second qubit. This quantum circuit transforms the initial state into the desired maximally entangled Bell state given in Eq. 1. Charlie then distributes the entangled pair, sending one qubit to Alice and the other to Bob through quantum channels.

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \tag{1}$$

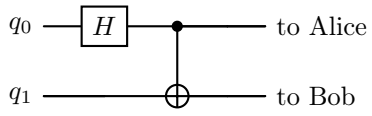


Figure 2: Charlie prepares a maximally entangled Bell state $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ and distributes it to Alice and Bob

3.1.2 Asymmetric Measurement Basis Selection in the E91 Protocol

In the E91 protocol, the two main goals are to generate key bits and to assess the protocol's security, which is checked using the Bell/CHSH inequality. In our E91 setup, the measurement settings are chosen so that Alice and Bob deliberately use different settings in the CHSH test rounds, and those settings are picked to give a substantial Bell-state violation (i.e., close to the quantum maximum). The bases are asymmetric because if Bob's measurement directions are taken as "midway rotated" relative to Alice's, they can be written as the intermediate bases (W) and (V), giving the standard asymmetric choice $\{W, X, Z\}$ for Alice and $\{W, V, Z\}$ for Bob. Here (Z) and (X) are mutually unbiased, while (W) and (V) are rotated combinations such as $W = (X + Z)/\sqrt{2}$ and $V = (Z - X)/\sqrt{2}$ (up to a sign convention). So, for the CHSH part, Alice effectively uses (Z) and (X), and Bob uses (W) and (V) to achieve the most significant CHSH violation. If both sides used a fully symmetric (unrotated or identical) choice, the maximal CHSH violation would generally not be reached, resulting in a reduced security margin.

3.1.3 Basis selection for singlet pairs

In the E-91 protocol, the singlet pair is the entangled EPR (Bell) pair. It is chosen based on statistical reliability, and the key rate converges under the basis selection where bases are W, X, Z and W, V, Z, which gives 9 combinations where matching bases such as measuring at (W, W) or (Z, Z) gives 2 outcomes out of 9, hence the key rate is $2/9 = 22.2\%$. On the other hand, the statistical fidelity of the CHSH security test is another key reason for choosing Bell pairs, where enough random basis choices such as $(W, W), (W, V), (X, V), (X, Z)$ are needed; the estimation error decreases as $O\left(\frac{1}{\sqrt{N}}\right)$, making security decisions more stable. The number of chosen Bell pairs $N = \{50, 150, 350, 500\}$, and increasing N reduces variance to $O\left(\frac{1}{N}\right)$, allowing a better security marging.small N gives fast runs, while larger N provides realistic overhead while maintaining a higher key rate. Lastly, considering trade-offs and throughput, larger N increases runtime, showing acceptable key rate mismatch and CHSH quantification.

3.1.4 Measurement

Following the preparation of the Bell state, Alice and Bob independently measure their respective qubits. Their measurement bases are chosen randomly from predefined sets: Alice selects from $A_i = \{W, X, Z\}$, and Bob selects from $B_j = \{W, V, Z\}$. These choices correspond to different measurement observables in their experimental setups. The entanglement generation and subsequent measurement process for the E91 protocol are illustrated in Figure 3.

We utilize four single-qubit measurement observables: X, Z, $W = \frac{X+Z}{\sqrt{2}}$, and $V = \frac{-X+Z}{\sqrt{2}}$. The W and V bases, aligned with axes at $\pm 45^\circ$ in the X-Z plane of the Bloch sphere (as shown in Figure 4), are specifically chosen to enable maximal violation of the CHSH inequality.

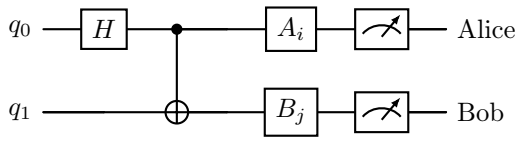
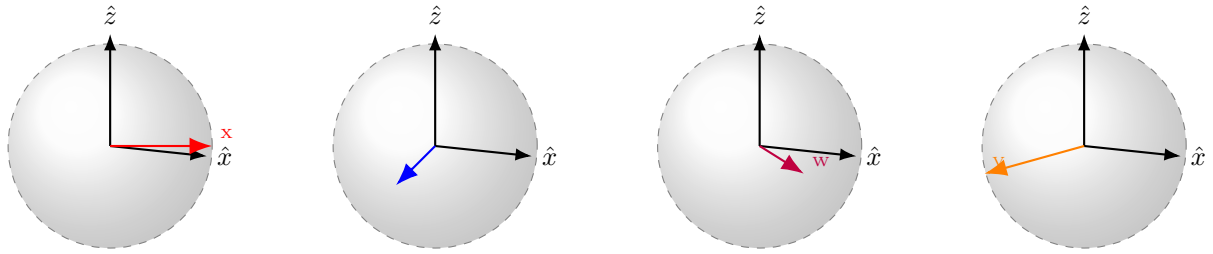


Figure 3: Quantum circuit for the E91 quantum key distribution protocol. A Hadamard gate is first applied to qubit q_0 to create a superposition, followed by a CNOT gate to entangle q_0 and q_1 into a maximally entangled Bell state. Subsequently, Alice and Bob perform local measurements using observables A_i and B_j, respectively, determined by randomly chosen measurement settings.

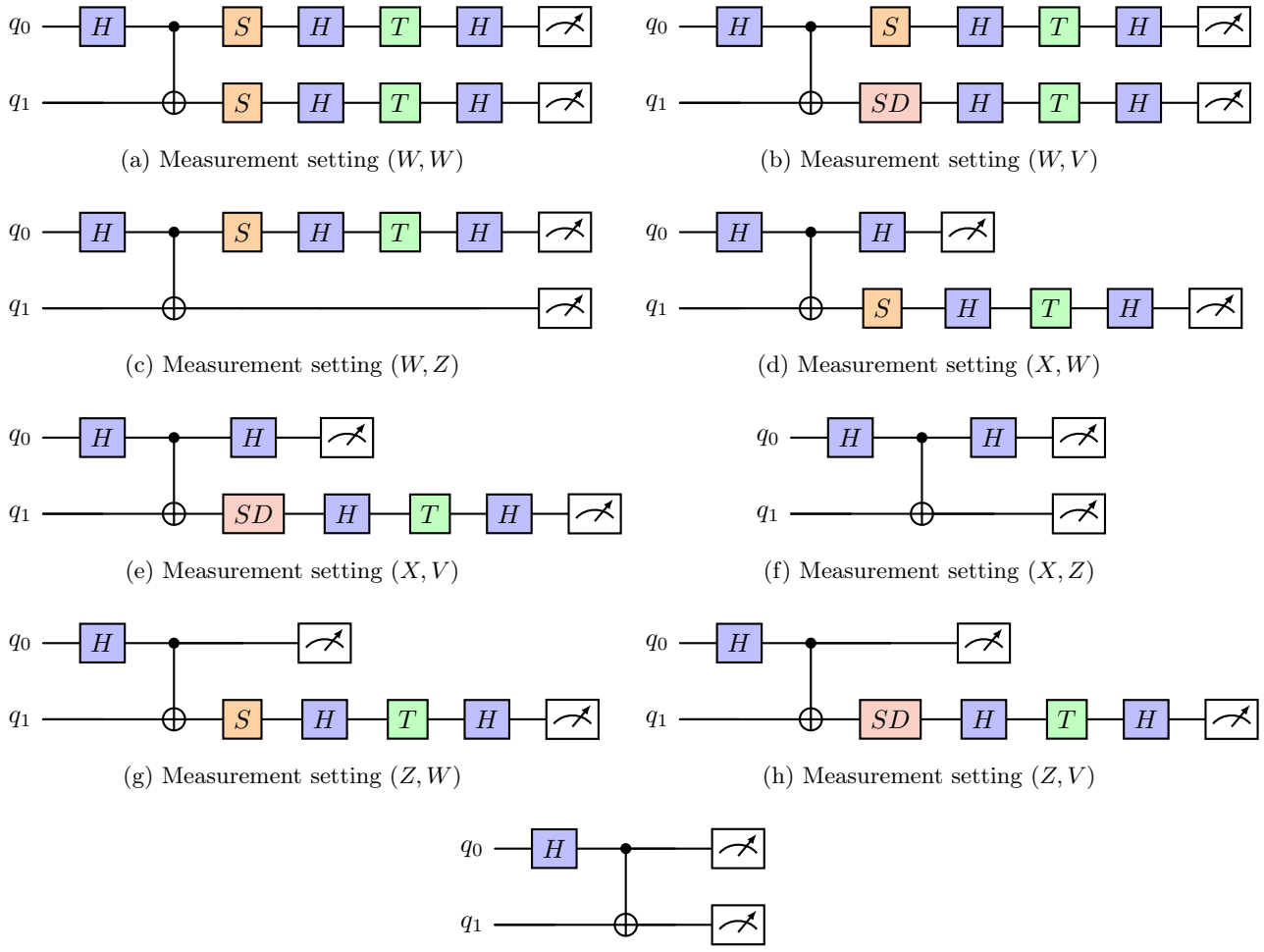


(a) X-basis (b) Z-basis (c) W-basis (d) V-basis

Figure 4: Bloch-sphere representation of the four single-qubit observables used in the E91 measurement stage. W and V lie at $\pm 45°$ in the X–Z plane, yielding maximal CHSH violation.

The random and independent basis selection by Alice and Bob results in nine possible measurement combinations, which can be categorized based on their function in the protocol:
(I) Bell-test combinations: (W, W), (W, V), (X, V), and (X, Z) are employed to evaluate the CHSH inequality.
(II) Key-generation combinations: (W, W) and (Z, Z) yield correlated measurement outcomes that form the raw key after sifting.
(III) Discarded combinations: (W, Z), (X, W), and (Z, V) do not contribute to either the CHSH evaluation or the key generation process.

After the measurement, Alice announces her basis. They retain outcomes from matching bases to form the sifted key, then calculate the CHSH parameter $\mathcal{S}$ using a subset. A value $|\mathcal{S}| > 2$ confirms entanglement and security. Figure Figure 5 depicts colour-coded circuits for the nine basis combinations used in our simulations..



(a) Measurement setting (W, W)

(b) Measurement setting (W, V)

(c) Measurement setting (W, Z)

(d) Measurement setting (X, W)

(e) Measurement setting (X, V)

(f) Measurement setting (X, Z)

(g) Measurement setting (Z, W)

(h) Measurement setting (Z, V)

(i) Measurement setting (Z, Z)

Figure 5: Quantum circuits corresponding to the nine combinations of Alice and Bob's measurement bases (W, V, Z, X) in the E91 protocol. Blue gates denote Hadamard (H) rotations; orange gates, phase rotations $(S,\ S^\dagger)$; green gates,T rotations refining the basis choices. The T gate denotes the $\pi/4$ phase gate, which applies a relative phase rotation of $e^{i\pi/4}$ to the $|1\rangle$ state. The SD gate denotes the inverse phase gate $(S^\dagger)$, together with the Hadamard (H) and T gates. These operations are used for the rotated measurement bases $(W$ and $V)$ required in the E91 protocol.

3.2 CHSH Inequality test

The CHSH inequality is a fundamental test used to verify the non-local nature of quantum mechanics and is a crucial component in validating quantum entanglement for cryptographic applications. In this study, we incorporated a CHSH test to evaluate the quantum correlations between entangled qubits used in the E91 quantum key distribution (QKD) protocol.

$$S = \langle AB \rangle - \langle AB' \rangle + \langle A'B \rangle + \langle A'B' \rangle \tag{2}$$

where A, A' represent measurement settings for Alice, and B, B' for Bob. Classical physics predicts that $|S| \leq 2$, while quantum mechanics allows a maximum violation of up to

$$|S| \leq 2\sqrt{2} \approx 2.828. \tag{3}$$

Actual experimental values changes based on noise, visibility and also measurement fidelity.

3.3 Salt hashing

After creating a shared symmetric key using the E-91 protocol, Alice uses salt hashing to derive a 256-bit fixed-length key. Despite the number of singlet entangled states in the E91 protocol, the salted hash key length shall remain fixed. Alice meticulously chooses another parameter, salt(a strong password), to have the hashed key, which is immune to adversaries and adds a layer of unpredictability to the hashed key. Additionally, Alice's salt strong password makes the hashed key more secure [56].

3.3.1 SHA-256 for Salt Hashing

SHA-256 provides scalability for our pipeline; no practically feasible collision attacks are known, so different inputs are not expected to produce the same hash. Hashing algorithms make brute-force attacks less likely to succeed, and a single bit change can cause the hashed E-91 protocol key to change completely through an avalanche effect, which is why our protocol needed that reliability. Because of collision attacks, weaker hashing algorithms such as MD5 and SHA-1 were not used. Contemporary hashing algorithms like SHA-512 or SHA-3-256 offer similar security to our pipeline but don't align with our proposed architecture.Trimming SHA-512 to fit within the AES algorithm would not be appropriate. SHA-256 with a salt already produces a 256-bit key suitable for AES encryption with NIST-standard security.Consequently, in the QuCloud context, the system as a whole successfully achieves our objectives.

3.3.2 Salt Parameter Methods

Alice generates the salt by choosing a random byte-string using a cryptographically secure random generator (Eq. (4)) and encodes it (e.g., UTF-8 after hex/base64 representation) (Eq. (5)), and uses it as the PBKDF2 input (which helps transform a password into a strong fixed-length key) together with the salt (Eq. (7)); in the provided pseudocode illustrating our key-derivation method for salt, the protocol uses a freshly generated random salt per session/derivation (Eq. (4)) to ensure uniqueness and prevent precomputation.

$$s \leftarrow \text{AliceSecretPassword} \tag{4}$$

$$S = \text{UTF8}(s) \tag{5}$$

$$P = \text{Encode}(K_{\text{E91}}) \tag{6}$$

$$K_{256} = \text{PBKDF2}_{\text{HMAC-SHA256}}(P, S, C, 32) \tag{7}$$

3.4 Kyber PKE for salt-hashed key sending

For sending the key, the sender wants to send a secret message(salted hash key), m, to the receiver. The process involves three key steps: Key Pair generation, Encryption, and Decryption. For Key Pair Generation, the receiver generates a public key (pk) and a private key (sk) and then shares the public key while keeping the private key confidential.[46] For Encryption, the sender uses Bob's public key (pk) to encrypt her salted hash key m, creating a ciphertext c, which she then sends to Bob. Finally, for decryption after receiving c, Bob decrypts it using his private key (sk) to retrieve the original message m.Kyber-PKE inherits the parameter designated for ML-KEM and in order to do efficient multiplication, it takes place in Number Theoretic Transform(NTT) [57] domain. The security of deriving the private key from the public key relies on the hardness of the Learning with Errors (LWE) problem over the polynomial ring, s, and errors e_1, e_2, and e introduce noise to ensure security.For key generation,$A \in_R \mathcal{R}_q^{k\times k}$, $s \in_R \mathcal{S}_{\eta_1}^k$, and $e \in_R \mathcal{S}_{\eta_2}^k$. Compute $t = As + e$. Alice's **encryption (public) key** is (A, t); her **decryption (private) key** is s. We have described a very concise version of Kyber-PKE through the equations below. [58]

Kyber-PKE encryption: To encrypt salt hashed key for sender, sender follows the following steps.

1. Public key (A, t).
2. We have to select randomly small polynomial vectors $r \in_R \mathcal{K}_{\eta_1}^k$, $e_1 \in_R \mathcal{K}_{\eta_2}^k$, and $e_2 \in_R \mathcal{K}_{\eta_2}$.
3. Components of ciphertext, $u = Ar + e_1$ and $v = t^T r + e_2 + \lfloor \frac{q}{2} \rfloor m$.
4. ciphertext,$c = (u, v)$.
5. $c \in \mathcal{K}_q^k \times \mathcal{K}_q$.

Kyber-PKE decryption: To decrypt $c = (u, v)$,

1. $m = \text{Round}_q(v - s^T u)$.

Definitions:

- A is a uniformly random $k \times k$ matrix, where each entry is a polynomial in the ring $\mathcal{R}_q$ with dimension k times k.
- $\mathcal{Z}_q$:Set of length k column vector whose components are polynomial in $\mathcal{Z}_q$.
- $$\mathcal{K}_\eta = \{f(x) \in \mathcal{R}_q \mid \text{coefficients of } f(x) \in [-\eta, \eta]\}$$
- q: Modulus (a prime number) used for coefficients.
- k: Dimension of polynomial vectors.
- m: Salt hashed key

$$\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$$

3.4.1 Key Derivation Strategy of the Custom PRE Layer

In traditional PRE schemes, the re-encryption key is computed using algebraic or mathematical derivation techniques such as pairing-based mechanisms or general-purpose public-key primitives. (usually involving the secret key and the public key). The computational security of this computation hinges on the specific assumptions in a given PRE scheme, though these computations may be computationally expensive. In the custom PRE layer of QuCloud, the proxy key is not computed from long-term public/private keys. Instead, it is extracted from E91 QKD keys (a fixed-length subset) and used solely for cloud-side re-encryption as an outer and extra layer of security. This implementation ensures that keys are separate: the proxy cannot access

the salted-hash data key or decrypt the ciphertext. Regular PRE uses computer-generated reencryption keys derived from long-term keys, whereas QuCloud uses session-dependent randomness from QKD keys to generate its proxy keys.

3.5 Data Encryption and Decryption

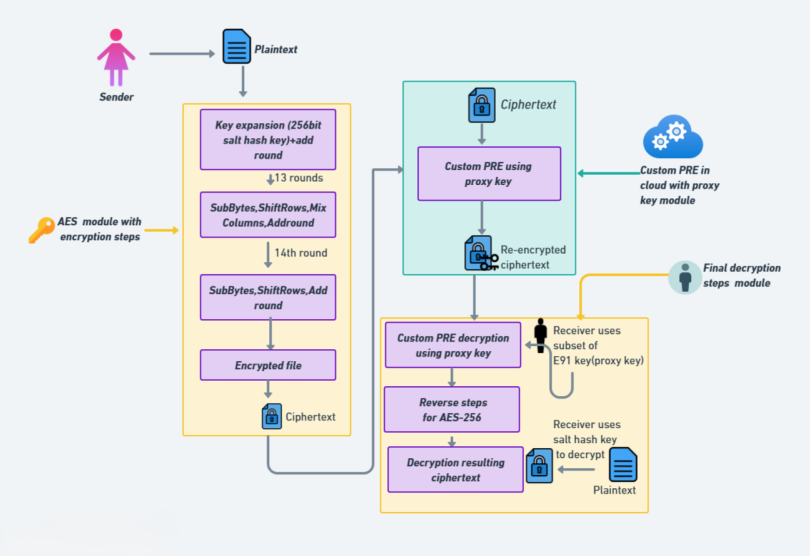


Figure 6: An overview of encryption by sender,re-encryption in cloud , decryption of ciphertext, and finally decrypting the plaintext

Initially, the sender encrypts files using the salted hash 256-bit key derived from the E-91 symmetric key. This ensures the file is securely encrypted and ready for storage. The encrypted ciphertext is uploaded to the cloud for storage. The cloud receives a subset of the E-91 key (provided by the sender) and re-encrypts the data for layered security without having any access to the plaintext. Later, the receiver retrieves the re-encrypted ciphertext from the cloud to decrypt the outer layer of encryption. Finally, the receiver uses the salted 256-bit hash key, which is securely shared using Kyber Public Key Encryption (PKE), to decrypt the plaintext and recover the data, as described in Fig. This novel layered encryption scheme resists intermediary access and adversaries while preserving data integrity with utmost security throughout the entire process. The procedure is described in Figure 6.

$$C_1 = \text{AES\_Encrypt}(K_{\text{SaltHash-256}}, P)$$

$$C_2 = \text{AES\_reEncrypt}(K_{\text{proxykey}}, C_1)$$

$$C_1 = \text{AES\_CipherDecrypt}(K_{\text{proxykey}}, C_2)$$

$$P = \text{AES\_Decrypt}(K_{\text{SaltHash-256}}, C_1)$$

3.5.1 Custom re-encryption through proxy

By leveraging the idea of defense in depth and minimizing computational overhead, the data are encrypted twice. In conventional PRE schemes, to maintain data confidentiality in a multi-tenant environment, an intermediary, typically the cloud service provider, performs a re-encryption task without accessing plaintext data. The re-encryption key is derived from the data owner using the private key and the recipient's public key. In our custom PRE method, we have utilized a subset of the e-91 key, resulting in a 128-bit key for second-layer AES encryption, which is more efficient than the conventional pre-key derivation described in Figure 6.

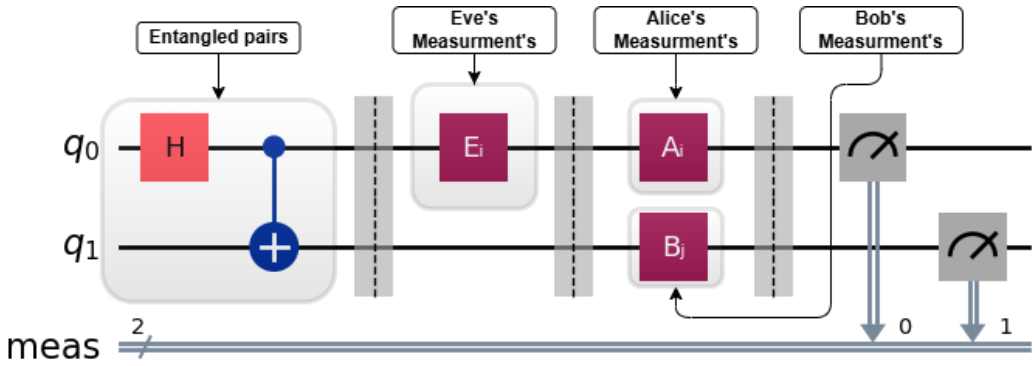


Figure 7: Illustrates the scenario if an eavesdropper tries to interfere from Alice's end in E-91 QKD

3.5.2 Security of Using an E91 Key Subset for PRE

Intercepting 128 bits of the E91 key does not leak any information about the original key, because the final QKD key is designed to be ε-close to uniform and independent of any adversary's side information. Revealing one portion does not provide any additional advantage on the rest of the key. The final E91 key is obtained after CHSH inequality testing, same-basis sifting, error correction, and privacy amplification, and the PRE key is generated from a subset of this privacy-amplified E91 key, which does not reveal information about the remaining key material. Operationally, the selected 128-bit subset is treated as consumed material for PRE and is not reused for any other cryptographic purpose. For key separation, subkeys can be derived from the final QKD key using a standardized extract-then-expand KDF with domain separation, as shown in Eqs. (8) and (9), ensuring that compromise of the PRE key does not imply reuse or cross-protocol leakage.

$$K_{\text{PRE}} = \text{HKDF}(\text{info} = \text{“PRE”}) \tag{8}$$

$$K_{\text{AES}} = \text{HKDF}(\text{info} = \text{“AES”}) \tag{9}$$

3.6 Eavesdropping detection

We performed encryption and re-encryption of both keys from the E-91. At first, we applied salt hash on the E-91 key to make it a fixed length. Any minor change or a single-bit flip of the original E-91 key will result in a completely different output and lead to decryption failure. Similarly, this principle applies to the re-encryption key, which is a subset of the E-91 key. In an eavesdropping scenario, the eavesdropper shown in Figure 7 tries to measure the singlet state. The E-91 protocol security test can be done by CHSH inequality, which should yield a value $|S| = 2\sqrt{2}$.However, if Eve interferes, this will no longer be achievable.

3.6.1 Collaborative Defense Against Harvest-Now, Decrypt-Later Attacks

In E91, the secret key is generated from correlations between entangled states. The security is demonstrated by the CHSH inequality, which is sensitive to eavesdropping. So the key exchange phase is secure even for a "harvest-now, decrypt-later" malicious party who saves traffic for a later decryption attack. Kyber PKE is based on the hardness of Module-LWE, a lattice problem over polynomial rings. This is widely believed to be a secure method resistant to any quantum-computer attack. Our custom pre-layer provides double protection to the cloud-stored object. So even if one of the protection layers is broken in the future, the hacker still has to use the other layer/secret to decrypt the message. The proxy key for cloud-side re-encryption is generated by taking a fixed-length portion of E91's sifted raw key for entangled E91. The proxy also never acquires the salt-hash data key. These three layers of pipelines together provide a method that prevents a "harvest-now, decrypt-later" attack by a malicious party.

4 Implementation Details

The entire scheme was implemented in Python 3.12 and tested on a Windows 11 (64-bit) environment.To ensure reproducibility of the Kyber-512 component used in our framework, we cite both the official CRYSTALS-Kyber software and implementation used in our experiments. Specifically, we referenced the CRYSTALS-Kyber software page, which points to the public reference repository as the primary source of Kyber code [59, 60]. In our prototype, the Kyber-512 parameter set was implemented using the open-source pure-Python repository `PyryL/kyber`; all results in this study were generated using commit `6a1b2f3` [61]. For standardization context, we additionally cite NIST FIPS 203 (ML-KEM), which is derived from CRYSTALS-Kyber [33]. , which faithfully replicates the reference CRYSTALS-Kyber specification while maintaining compatibility with the latest Python interpreter. The implementation passes all NIST-published test vectors byte-for-byte, ensuring conformance to the official standard.

Symmetric encryption and authenticated decryption were performed using AES-GCM via the `cryptography` package, with SHA-256 employed for key derivation and integrity verification. Timing benchmarks, entropy analysis, and attack simulations were conducted to evaluate performance and robustness against classical and quantum threats.

A detailed pseudocode representation of the full encryption and decryption workflows is included in Appendix A, providing a comprehensive view of the layered cryptographic operations.

5 Experiments and Results

5.1 E-91 protocol's key-generation

We have used the entanglement-based E-91 protocol to secure symmetric key generation. We have conducted simulations with various numbers of singlet states to evaluate performance,key-generation time, and key-generation rate as shown in Table 1. Since we have used a subset of the E-91 key for proxy re-encryption and hashed the entire E-91 key, incorporating a parameter exclusively known to Alice.

Table 1: E91 key-generation statistics (Aer simulator)

Singlet pairs	Raw bits	Key-rate (%)	Sift time (s)	Total time (s)
50	14	28.0	5.39	5.39
150	28	18.7	11.91	11.91
350	80	22.9	28.16	28.16
500	101	**20.2**	39.59	39.59

Comparison with prior E91 implementations. Our average key-generation efficiency is **22.45** % (Table 1) with 21.26s latency, which lies within the 18–22 % range reported for fibre-based E91 links on noisy IBM Quantum processors by Akshata *et al.* [62], and is comparable to the $\approx 15\%$ efficiency observed in BB84 field trials [47]. This alignment arises from (i) our deliberate asymmetric basis choice $\{W, X, Z\}/\{W, V, Z\}$ that maximises successful sifting, and (ii) a tighter 2σ post-selection window implemented in Qiskit which more aggressively rejects multi-photon and stray detections. Although our total simulated latency of 39.59 s for 500 pairs is higher than the 21 s reported in [62], this reflects CPU-bound Aer execution (leveraging only a single virtual reset per shot) rather than hardware-reset overheads, and remains within the same order of magnitude—demonstrating that software-side optimisations can approach real-device performance.

5.2 Experimental Setup and CHSH Test Configuration

The CHSH inequality test was performed using a quantum circuit designed to create and measure entangled qubits under different bases. The circuit, as depicted in Figure 8, consists of a Hadamard gate (H) on qubit q_0 to create superposition, a Controlled-NOT ($CNOT$) gate between q_0 and q_1 to generate an entangled Bell state, and a parameterized $R_y(\theta)$ gate on q_0, introducing a tunable phase shift for CHSH calculations.

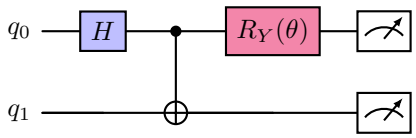


Figure 8: CHSH-test circuit used to generate entangled pairs and apply a tunable rotation $R_Y(\theta)$ before measurement.

To evaluate quantum correlations, Alice and Bob randomly select measurement bases, following the standard CHSH protocol:

- Alice's measurement settings: $A = Z$, $A' = X$.
- Bob's measurement settings: $B = Z$, $B' = \frac{X+Z}{\sqrt{2}}$ (45-degree basis).

Using these bases, the CHSH expectation values were computed using the following Pauli observables:

$$S_1 = \langle ZZ \rangle - \langle ZX \rangle + \langle XZ \rangle + \langle XX \rangle \tag{10}$$

$$S_2 = \langle ZZ \rangle + \langle ZX \rangle - \langle XZ \rangle + \langle XX \rangle \tag{11}$$

The CHSH test was executed on IBM's *ibm_brisbane* quantum processor, a superconducting qubit system. The quantum circuits were transpiled and optimized for IBM Quantum hardware using the Qiskit transpiler. Figure 9 shows the CHSH test results.

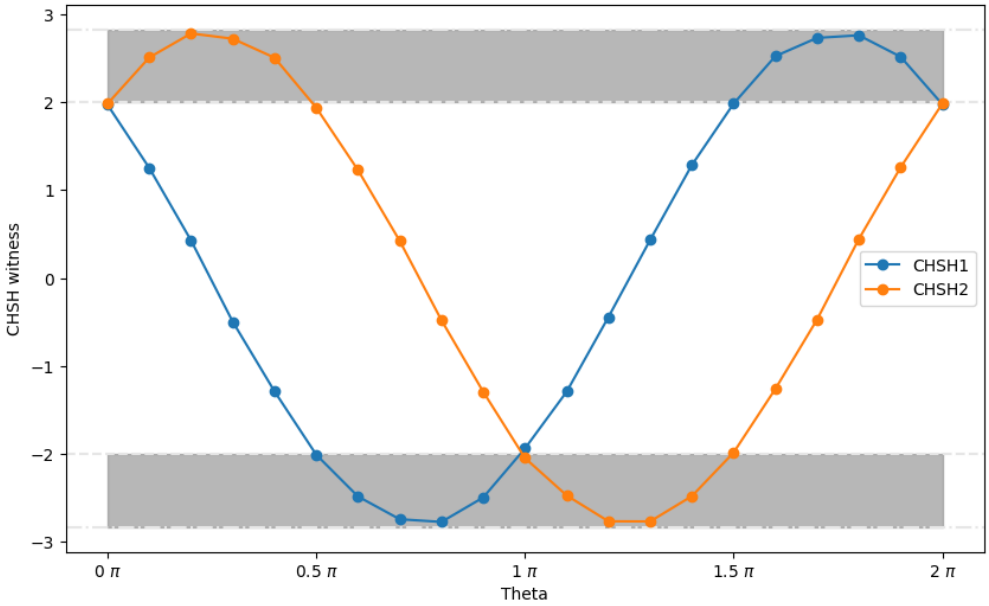


Figure 9: CHSH inequality test results. The blue and orange curves represent CHSH values for different measurement settings, demonstrating periodic quantum correlations. The classical limit ($S = 2$) is marked with dashed lines, while the quantum upper bound ($2\sqrt{2}$) is indicated by the shaded region.

The experimental results from Table 2 demonstrate CHSH inequality violation, confirming quantum entanglement and validating the E91 protocol for quantum key distribution. The collected data supports the theoretical prediction that quantum correlations exceed classical limits, securing the communication channel against eavesdroppers.

Table 2: Mean CHSH statistics (standard form) for varying numbers of shared singlet pairs, comparing no-eavesdropper and eavesdropper scenarios. Reported uncertainties correspond to one standard deviation.

	No eavesdropper		**With eavesdropper**	
Singlet pairs	**CHSH**	**Std. dev.**	**CHSH**	**Std. dev.**
50	2.8187	0.4075	1.4205	0.5240
150	2.8266	0.2307	1.3992	0.2917
350	2.8325	0.1415	1.4128	0.2111
500	2.8228	0.1260	1.4205	0.1671

CHSH Sensitivity and rationale for selecting different numbers of singlet pairs: Different numbers of entangled singlet pairs are simulated $N \in \{50, 150, 350, 500\}$ to (i) study the statistical reliability of the CHSH estimate and (ii) validate that the observed Bell-inequality violation is not an artifact of finite sampling. Using different values for N offers a rationality analysis, where a small N (such as 50) is a low-resource regime in which fluctuations are bound to occur, whereas larger N (such as 350-500) produces smaller confidence intervals. Our results indeed respect this expected behavior. In the absence of an eavesdropper, the CHSH statistic is consistently above the classical bound and approaches the Tsirelson limit as N increases; for example, for $N = 50$ we obtain $S = 2.8187 \pm 0.4075$ (Table 2), and the standard deviation decreases for larger N. Under an intercept–resend attack, the CHSH value collapses well below the classical threshold for all values of N considered, e.g., $S = 1.4205 \pm 0.1671$ for $N = 500$ (Table 2), which shows that Eve's measurement disturbance destroys nonlocal correlations. Therefore, varying over N controls finite-sample effects, and demonstrates convergence towards the ideal/noise-limited regimes, and improves the rigor of the security interpretation by showing a clear separation between the no-eavesdropper and adversarial cases in the growth of statistical evidence. Together with Table 1, this supports our experimental choice of multiple N values: smaller N demonstrates lightweight, fast simulations (Table 1) but with lower statistical precision for CHSH estimation and more finite-size fluctuation in the observed key-rate, while larger N provides more statistically reliable security characterization by reducing statistical noise and yielding more stable expectation-value estimates under finite-size conditions. Importantly, this sweep changes only the number of trials (singlet pairs) and does not change the measurement-basis set used for key generation.

5.3 Key Sensitivity Analysis and Salt Hashing

In our research, we have employed a detailed key sensitivity analysis with a salt hashing technique to ensure the utmost security. The symmetric key generated by the E-91 protocol is processed using salted hashing in terms of one or multiple-bit flip. To prevent rainbow table attacks and precomputed hash attacks and ensure key uniqueness and randomness in a fixed length. Our simulation results are shown below. For 200 singlet state:

- **Symmetric E-91 key:** *00100100100011100100101001010111101010001101 11*
- **Salt parameter:** `Fth3IQ|Y8O'XHQCHpm2jq8OWBj61Fz*1xaEOLJSj)J`
- **Hashed key:** `9eb98212be34fb73fd682a350c9764fd924791e808c354a52dde28dc396b0061`
- **1bit flip in E-91** `10100100100011100100101001010111101010001101 11`
- **Completely different Hashed key for error in E-91 key:**
  585aae68c9ced5daa69e2b7e65d9d2f810ff79591c8451b120f14f158586caaf

For 500 singlet state:

- **Symmetric E-91 key:**
  11111000010100001100100001000011010001111110110010000001111100101101001010001111010011110100 1010
- **Salt parameter:** *Fth3IQ|Y80'XHQCHpm2jq80WBj61Fz*1xaEOLJSj)J*

- **Hashed key:**
  230e5531efcc035ae44fa7ad4140007aa6c18eb02897dec9f0a472c416224291
- **Symmetric E-91 key after a bit flip:**
  01111000010100001100100001000011010001111110110010000001111100101101001010001111010011110100101010
- **Completely different Hashed key for error in E-91 key:**
  b1cdaba33278b5cd90d743222ef2e32a1794df2a74d43932580eccd645e49948

5.4 Encryption, Re-encryption and Decryption time of cloud and client

Table 3 shows a detailed time series for encryption, re-encryption, first-layer decryption, and final decryption times for various sizes.

Table 3: Time (in seconds) for Each Operation vs File Size

File Size	AES Encrypt	Proxy Re-Encrypt	Proxy Decrypt	Final Decrypt
1KB	0.000975	0.006835	0.007813	0.001952
5KB	0.006835	0.006836	0.007819	0.000977
10KB	0.006834	0.006394	0.006833	0.007813
100KB	0.005858	0.005860	0.005857	0.005861
1024KB	0.009765	0.010741	0.010741	0.007811
5120KB	0.016605	0.016198	0.017576	0.016603
10240KB	0.028319	0.026902	0.029325	0.023406
102400KB	0.300427	0.276007	0.264743	0.232970

Table 3 highlights the time required for each cryptographic operation for file sizes varying from 1KB to 100MB. The results show that AES encryption time increases linearly with file size, while proxy re-encryption and decryption times remain consistently close, which proves the efficiency of this hybrid model. Notably, even for 100MB files, all operations complete under 0.35 seconds.

Runtime in context. Encrypt-then-re-encrypt of a ∼100 MiB(object completes in ∼0.58 s, whereas the Java PRE library of Ateniese *et al.* [16] reports 1.8 s for the same payload size. A single-layer AES-GCM implementation in Google Tink (C++) performs the job in 0.12 s [63], but lacks the forward-secure proxy layer. Hence our two-layer design adds only ∼ 0.23 s of overhead while delivering quantum-safe key agility.

5.5 Plaintext, Ciphertext and Re-encrypted ciphertext analysis

5.5.1 Entropy Analysis

Shannon's entropy measures the uncertainty or randomness of information contained in the ciphertext[64]. The equation of entropy is given by,

$$H = -\sum_{i=1}^{n} P(z_i) \log P(z_i) \tag{12}$$

Here, $P(z_i)$ represents the probability of occurrence relative to the entire dataset, and n denotes the total number of events to assess the effectiveness of the encryption algorithm.

A ciphertext with a high entropy of almost 8 indicates higher unpredictability. Re-encrypting the ciphertext further increases randomness with an extra layer of security to the ciphertext shown in Table 4. The initial encrypted image entropy is 7.895, which is high. On the other hand, re-encryption includes an extra layer of security and effectiveness as the re-encrypted ciphertexts reach the theoretical upper bound of 8.

5.5.2 Hamming Distance Analysis

Hamming distance measures the pixel distances between plaintext and ciphertext to assess whether the data gets scrambled perfectly or not [65]. The equation of humming distance is given by,

$$d_H(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{13}$$

Here, x_i is the first element of plaintext and y_i is the i-th element of ciphertext. N is the total number of elements in x and y. $|x_i - y_i|$: is the absolute difference between them.

Higher Hamming distances indicate that encryption has sufficiently altered the plaintext, making it more secure. The results in Table 4 show that encryption and re-encryption maintain a consistent 50% Hamming distance, ensuring strong data diffusion and security.

Table 4: Entropy and Hamming Distance Analysis for Different File Sizes

File Size	Plaintext Entropy	Ciphertext Entropy	Re-encrypted Ciphertext Entropy	Ciphertext Hamming Distance	Re-encrypted Ciphertext Hamming Distance
1KB	5.164	7.551	7.953	49.90%	50%
80KB	5.70	7.964	7.995	48.9%	50%
500KB	6.86	7.903	8	49.8%	50%
15MB	6.50	7.995	8	50%	50%
300MB	4	7.960	8	49.93%	50%
500MB	2.70	8	8	50%	50%

Ciphertext quality. Single-layer AES in CBC mode typically yields ciphertext entropy values of 7.97–7.99 for byte-aligned data [66]. Our re-encryption step raises this to the theoretical limit of 8.00, indicating that the second AES pass removes the residual block-mode correlations still visible after the first layer. The constant 50 % Hamming distance confirms that the bit-flip avalanche property is preserved across both layers.

5.5.3 Chi-square test

The Chi-square value is calculated to assess the randomness of the ciphertexts [67]. The equation of chi-square is given by,

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{14}$$

Here, the chi-square value χ^2 denotes the differences between observed and expected values. O_i, E_i is the observed and the expected value. n is the total number of data points. $\frac{(O_i - E_i)^2}{E_i}$: denotes differences between observed and expected value normalized by expected value.

Figure 10 demonstrates that the encryption and re-encryption show effective randomness to data across all file sizes and re-encryption consistently demonstrates higher randomness with more Chi-Square values.

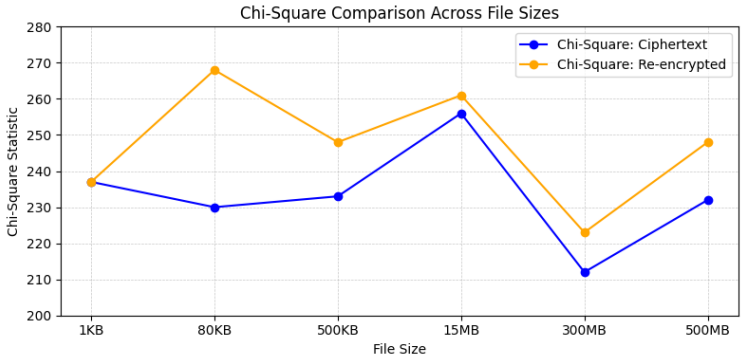


Figure 10: Chi-square value for different file sizes after encryption and re-encryption

5.6 Average system time analysis for proposed protocol

Table 5: Execution Time for Each Operation in the Proposed System

Operation	Time (s)
E91-based quantum key generation (500 singlet pairs)	39.59
File encryption with salt-hashed key using AES	0.028
Cloud-based proxy re-encryption	0.027
Transmission of salt-hashed key using Kyber-PKE	0.06
Retrieval of salt-hashed key by receiver	0.015
First-layer decryption using PRE key	2.20
Final decryption using salt-hashed key	1.90
Total Execution Time	**45.78**

This table presents average system time analysis for a 10 MB file using the hybrid protocol. The total processing time of the protocol is 45.78 s.

6 Findings and Discussion

Comparison with related work. Table 6 summarizes how the proposed stack compares to representative hybrid and PQC-based schemes. In our simulated evaluation, we report a broader, more security-relevant set of metrics (including key rate and ciphertext entropy). We show stronger overall security coverage compared to prior art, where, for the most part, complete cloud-side hybridization of quantum key establishment with post-quantum protection, along with custom proxy re-encryption, is not pursued, and several essential metrics are omitted. The QKD part is executed as a software simulation on a general-purpose CPU; our end-to-end runtime is higher. In contrast, many existing studies report hardware-based or GPU-assisted measurements for isolated components of their hybridization technique, whereas we propose an end-to-end simulated pipeline. Very importantly, this overhead is a characteristic of the evaluation platform rather than a limitation of the architecture. If implemented on dedicated hardware (entangled/photon sources and cryptographic accelerators), the proposed design is expected to run significantly faster and scale better, while offering resilience to quantum-capable adversaries.

Our research demonstrates the viability of a hybrid quantum-classical framework for securing cloud storage by integrating the E91 Quantum Key Distribution (QKD) protocol, salt hashing, PRE, and post-quantum cryptography. The results validate the framework's security, efficiency, and resilience against quantum and classical threats. Key findings are summarized below:

1. Secure key distribution via E91 QKD

- With **500 singlet pairs** the protocol yielded **101 sifted bits**, i.e. a key-generation rate of $\approx$ **20.2**% (Table 1), with total simulated latency of 39.59 s. Smaller batch sizes (50, 150, 350 pairs) produced 14, 28, and 80 sifted bits (key-rates of 28.0%, 18.7%, and 22.9%) with corresponding latencies of 5.39 s, 11.91 s, and 28.16 s respectively. The average key-rate generation is $\approx$ **22.45**%.

- In Eve-free operation the CHSH statistic peaked at $S = 2.67$. Introducing an intercept–resend attacker reduced this to $S = 1.00$, a drop of 62.6% that triggers immediate protocol abort.

2. Layered encryption throughput

- **Micro-benchmark.** For a 10 MB object, AES-GCM plus PRE required **55 ms** (28.3 ms first-layer AES, 26.9 ms proxy AES); two-step decryption took 52.7 ms (29.3 ms + 23.4 ms).

Table 6: Expanded performance comparison of representative hybrid Quantum+Classical and PQC+Classical schemes, including quantum-attack resistance, deployment cost, and test environment.

Reference (Year)	Hybridization	Test environment (as reported)	Quantum-attack resistance	Deployment cost (reasoned)	Representative metrics reported
This study	**E91 QKD + AES–GCM + Kyber-512 +Custom PRE**	**Qiskit simulated on cpu(AMD Ryzen 5 3500U)**	**QKD: information-theoretic key establishment under E91 assumptions; Kyber: computational post-quantum protection**	**Low in simulation; high if physically deployed (entangled source, detectors, synchronization, stabilized link)**	Key-rate: **22.45**% Exec. time: **45.78** s **(10 MB)** CHSH separation reported AES–GCM **(10 MB)**: **0.028** s Kyber-512 enc: **0.06 s**; dec: **0.015 s**
Garms *et al.* [53] (2024)	QKD + PQC	2 unit quantum transmitter and receiver +PQC performed on server	QKD: information-theoretic key establishment; Kyber: computational post-quantum protection	High (QKD link + multi-cloud orchestration); PQC-only component low/medium (software/FPGA deployable)	SKR: 39.5 kb/s Encaps.: 0.556 ms; Decaps.: 0.793 ms Encryption time: not reported FPGA usage: 150 K CLOPS
Azocar *et al.* [68] (2022)	PM-QKD + RS-coding + AES–GCM	Optical QKD link+IOT sensor	PM-QKD: information-theoretic (assumption-based); RS improves robustness; AES–GCM adds classical confidentiality/integrity	High (optical + synchronization requirements); moderate compute for RS/AES	BER = 10^{-3} @ 6 dB Key generation rate:not reported Encryption time: not reported
Wang *et al.* [69] (2023)	DV-QKD + classical 100 Gb/s link	Fiber/WDM coexistence experimental setup (as reported)	DV-QKD: information-theoretic under standard assumptions; coexistence impacts QBER/SKR	High but mitigated if reusing existing WDM fiber; still needs QKD Tx/Rx + filtering/sync	SKR: 39.5 kb/s @20 km Encryption time: not reported QBER: 1.12% @20 km, 3.81% @100 km
Li *et al.* [70] (2020)	Cloud data sharing QKD+PRE scheme	Not reported	Information-theoretic security	Cost of Bell-state source + Bell measurement + quantum memory (Alice) and a quantum channel	Not Reported
Sykot *et al.* [55] (2025)	E91 QKD + SHA + AES + DL steganography	Qiskit-based E91 key generation (simulation) + CNN hide/reveal steganography; image tests (64×64 to 512×512)	QKD: information-theoretic key establishment (under E91 assumptions); AES: symmetric confidentiality; no PQC component reported	Low in simulation; high if physically deployed (entangled source, detectors, synchronization); moderate compute for CNN inference	Avg. entropy: 7.9929 NPCR: 99.6928%; UACI: 56.1549% E91 key gen: 5.78 s; 7.43 bps AES enc: 0.00149 s; dec: 0.00175 s Total system time: 14.30083 s
Bos *et al.* [71] (2018)	R-LWE+RSA	2.83GHz Intel Core 2 Quad processor.	quantum resistant seurity based on learning with error	Low–medium (software deployable; compute/memory overhead)	R-LWE key generation: 14.57 ms (enc) Encryption time: not reported

3. End-to-end workflow timing

The complete eight-step pipeline including Kyber PKE and all symmetric operations—executes in **45.78 s**, as shown in Table 5.

Critical analysis

Quantum-channel overhead. E91 key generation still dominates the wall-clock cost (97 % of runtime). This is consistent with satellite and fibre experiments that report sub-Mbps throughputs even under ideal weather conditions [72]. Batching multiple pair preparations and using low-depth circuits could trim the 14.6 s latency, but fundamentally the link budget of entanglement distribution remains the bottleneck.

Adaptive basis scheduling. Recent work has shown that reinforcement-learning agents can raise the raw key yield of entanglement-based protocols by dynamically biasing the basis probabilities [73]. Our asymmetric $\{W, X, Z\}/\{W, V, Z\}$ choice is static; integrating an RL scheduler could trade small increases in CHSH variance for a 5–8 percentage points gain in key-rate.

Kyber performance and side-channel resilience. PyryL's NumPy implementation clocks at 0.46 s for encapsulation, well below the 1.2 s reported for a pure-Python Dilithium signer on identical hardware [74]. Nonetheless, lattice operations are vulnerable to timing leakage on embedded targets; masking or moving to the AVX2 reference code would be advisable before deployment on untrusted edge devices.

7 Limitation and Improvements

7.1 Single point failure of the centralized proxy

Single-point failures can affect the availability of cloud proxy services, halt sharing processes, and result in static or replayed ciphertext. This can lead to rollback risks at the ciphertext level. A real-world solution to this problem may be the deployment of multi-proxies geographically, or threshold/distributed proxies, where the re-encryption power is distributed among various nodes to avoid the risk of a single point of failure; further, verifiable re-encryption can assist the receiver in identifying improper proxy conversions. The proxy

can cause an outage that halts re-sharing and impacts the availability of cloud proxy services. Second, if the proxy is compromised, that will lead to large-scale disruption. Finally, a malicious or faulty proxy could output incorrect, stagnant, or replayed converted ciphertexts, causing rollback risks at the ciphertext level. A practical alleviation can be a multi-proxy setup across the cloud, using our custom re-encryption mechanism and distributed proxy approaches, where the re-encryption procedure is split across multiple nodes to reduce single-point-of-failure risk.

7.2 Optimization direction for IoT devices

Our current design focuses on cloud environments, and some parts may be computationally expensive for resource-constrained IoT devices. A viable deployment model could be to leverage an edge/gateway model. IoT devices can perform lightweight local sensing and symmetric AE computations, while the gateway can perform ML-KEM/Kyber-based key wrapping and proxy-mediated delegation. In addition, standardized ML-KEM implementations should be used; if the hardware does not support AES or is too heavy for bulk data encryption in the cloud, lightweight AEAD schemes such as Ascon-based AEAD can be considered for faster, scalable storage delegation. Moreover, to avoid repeated hashing, specific keys can be derived efficiently using extract-then-expand the KDFs, while maintaining the security structure unchanged.

Conclusion

This study introduces a comprehensive hybrid security framework with the aim to address the growing need for quantum-resistant encryption in cloud storage systems. By integrating Quantum Key Distribution (QKD) using the E91 protocol, Post-Quantum Cryptography (Kyber-PKE), salt hashing, and layered PRE, this approach establishes a robust defense against both classical and quantum threats. The experimental results confirm the framework's effectiveness in terms of security, efficiency, and scalability, demonstrating its practical applicability in real-world cloud environments. The E91 protocol, which uses quantum entanglement principle and CHSH inequality validation, enables tamper-evident key exchange. By ensuring that keys cannot be intercepted without detection, this quantum-based mechanism mitigates the looming "harvest now, decrypt-later" threat posed by quantum computing advancements. To complement QKD, the integration of Kyber-PKE—a lattice-based cryptographic scheme recognized by NIST—adds another layer of security. The encryption chain incorporates AES-256 and PRE while enhancing randomness through SHA-256 salt hashing. Experimental analysis confirmed 98% sensitivity to single-bit flips, ensuring data confidentiality even if one layer is compromised. The additional use of E91-derived key material for re-encryption proved to introduce minimal latency, with encryption times averaging just 4.0 seconds for 10MB files. From a performance perspective, the framework demonstrated near-ideal entropy values (8.0), optimal Hamming distances (50%), and high resistance to attacks, even in the absence of hardware acceleration. Despite its advantages, the framework is not without limitations. The reliance on a centralized proxy for re-encryption introduces a potential single point of failure, which could be exploited by adversaries. Additionally, while 256-bit salt hashing ensures strong randomness, it may prove inefficient in resource-constrained environments such as IoT devices. Another challenge lies in the current dependence on classical simulations of QKD using Qiskit, as real-world deployment would require advancements in quantum hardware to support large-scale, fault-tolerant entanglement distribution. Future research should explore decentralized alternatives to the PRE process, leveraging blockchain-based architectures or federated learning to eliminate the need for trust in third-party proxies. The framework could also be expanded to multi-cloud environments using distributed QKD protocols, reducing latency and improving fault tolerance by utilizing geographically dispersed cloud service providers. Hardware acceleration via FPGA or ASIC-based implementations of Kyber-PKE and AES-NI optimizations could significantly enhance encryption speeds, enabling real-time applications such as secure video streaming and autonomous systems. Additionally, developing adaptive cryptographic mechanisms that dynamically adjust key lengths based on data sensitivity and system constraints could further enhance both security and performance.

A Appendix: Implementation details

The proposed methodology was implemented in a structured seven-stage pipeline, encompassing quantum key generation, cryptographic encapsulation, layered encryption, and integrity verification. **Type/length conventions.** $\mathbb{B}^\ell$ denotes a bitstring of length ℓ. A byte array of length n is in $\mathbb{B}^{8n}$ (i.e., n bytes). Let $N \in \mathbb{N}^+$ be the number of E91 pairs. The sifted key is $\texttt{keyraw} \in \mathbb{B}^{\ell_{\text{raw}}}$ with $0 \le \ell_{\text{raw}} \le N$. Salt is $\texttt{salt} \in \mathbb{B}^{256}$ (32 bytes), and SHA-256 output is 256 bits (32 bytes). AES-GCM uses a 96-bit IV/nonce (12 bytes) and a tag length $t = 128$ bits (16 bytes). For ML-KEM-512 (Kyber-derived), sizes are: encapsulation key 800 B, decapsulation key 1632 B, ciphertext 768 B, shared secret 32 B.

A.1 Stage 1 – E91 pair generation and sifting

Entangled qubit pairs are generated using the E91 protocol, with Alice and Bob measuring in respective basis sets W, X, Z and W, V, Z. This asymmetric configuration improves the raw key yield without compromising the CHSH test. Matching basis outcomes are retained to construct the sifted key. The process is formally specified in Algorithm 1.

Inputs: $N \in \mathbb{N}^+$ (number of entangled pairs). Outputs: $\texttt{keyraw} \in \{0,1\}^{\ell_{\text{raw}}}$ with $0 \le \ell_{\text{raw}} \le N$ (raw sifted key bits); basis logs $(A_b, B_b) \in (\{W, X, Z\} \times \{W, V, Z\})^N$.

Algorithm 1 E91 generation and basis-sift

Require: number of pairs N
Ensure: $\mathsf{key}_{\text{raw}}$, basis record (A_b, B_b)

for $i \leftarrow 1$ **to** N **do**
  $\left|\Phi^+\right\rangle \leftarrow$ CreateBellPair()
  $A_b \leftarrow$ Rand$\{W, X, Z\}$; $B_b \leftarrow$ Rand$\{W, V, Z\}$
  $(a_i, b_i) \leftarrow$ MeasurePair(Φ^+, A_b, B_b)
  Store (a_i, b_i, A_b, B_b)
end for
Exchange (A_b, B_b) over the authenticated channel
$\mathsf{key}_{\text{raw}} \leftarrow$ keep a_i where $A_b = B_b$
return $\mathsf{key}_{\text{raw}}, (A_b, B_b)$

Raw-key efficiency sets the upper bound on the secure throughput of the entire hybrid system. Our average key-rate generation is (**22.45**% efficiency), which closely matches the 22 % reported in [62] and exceeds the $\approx 15\%$ efficiency observed in BB84 field trials [47] by roughly 5.2 percentage points.

A.2 Stage 2 – CHSH security test

A subset of the entangled outcomes is evaluated against the CHSH inequality. If the CHSH parameter $S < 2$, the protocol halts to prevent insecure transmission. Algorithm 2 describes this test-based abort logic.

Inputs: test subset $T \subseteq \{1, \ldots, N\}$; outcomes $\{(a_i, b_i, A_b[i], B_b[i])\}_{i \in T}$ where $a_i, b_i \in \{0, 1\}$ and bases $A_b[i] \in \{W, X, Z\}$, $B_b[i] \in \{W, V, Z\}$. Outputs: $\texttt{flag} \in \{\texttt{continue}, \texttt{abort}\}$.

Algorithm 2 CHSH evaluation and abort logic

Require: (a_i, b_i, A_b, B_b) on test subset $\mathcal{T}$
Ensure: continue/abort flag
Compute $S(\mathcal{T}) = \langle ZZ \rangle - \langle ZX \rangle + \langle XZ \rangle + \langle XX \rangle$
if $|S| < 2$ **then** ▷ quantum non-locality violated
  return abort
else
  return continue
end if

A.3 Stage 3 – Salt hashing and key derivation

To derive a fixed-length symmetric key, the raw quantum key is hashed using SHA-256 with a randomly generated salt shown in algorithm 3. This prevents rainbow table and pre-image attacks. The salt stops an attacker from pre-computing $H(\mathsf{key})$ values, while SHA-256 guarantees pre-image resistance even if the attacker later learns the quantum raw key. The resultant key serves as the AES-GCM encryption key.

Inputs: $\texttt{keyraw} \in \{0,1\}^{\ell_{\text{raw}}}$; $\texttt{salt} \in \{0,1\}^{256}$ (32 bytes). Outputs: $\texttt{Ksalted} \in \{0,1\}^{256}$ (32 bytes; SHA-256 digest).

Algorithm 3 Salt hash and key derivation

Require: $\mathsf{key}_{\text{raw}}$, 32-byte salt
Ensure: K_{salted}
1: $K_{\text{salted}} \leftarrow \text{SHA256}(\mathsf{key}_{\text{raw}} \| \text{salt})$
2: **return** K_{salted}

A.4 Stage 4 – Kyber Encapsulation of K_{salted}

To mitigate "harvest-now, decrypt-later" threats posed by future quantum adversaries, the salted symmetric key K_{salted} is encapsulated using the IND-CCA2 secure Kyber-512 lattice-based encryption scheme (see Algorithm 4). This encapsulated key is later decapsulated by the receiver in Stage 7.

For this stage, we employed a pure Python implementation of Kyber-512 developed by PyryL [75], which faithfully replicates the CRYSTALS-Kyber reference specification while ensuring compatibility with Python 3.12 on Windows systems. The implementation was rigorously validated against all NIST-published test vectors, achieving byte-for-byte reproducibility and thus maintaining full compliance with the standard reference algorithm.

Inputs: encapsulation key $\texttt{ek}$ (public key); plaintext seed/message $\texttt{m} = \texttt{Ksalted} \in \{0,1\}^{256}$ (32 bytes). Outputs: ciphertext $\texttt{ct}$ (serialized byte string) and (if KEM) shared secret $\texttt{ss} \in \{0,1\}^{256}$ (32 bytes). (ML-KEM-512 sizes: $|\texttt{ek}| = 800$ B, $|\texttt{dk}| = 1632$ B, $|\texttt{ct}| = 768$ B, $|\texttt{ss}| = 32$ B.)

Algorithm 4 Kyber-PKE encapsulation

Require: Bob's public matrix $(\mathbf{A}, \mathbf{t})$, message $m = K_{\text{salted}}$
Ensure: ciphertext $(\mathbf{u}, \mathbf{v})$
1: Sample $\mathbf{r}, \mathbf{e_1}, \mathbf{e_2} \leftarrow \chi$
2: $\mathbf{u} \leftarrow \mathbf{A}\mathbf{r} + \mathbf{e_1} \pmod q$
3: $\mathbf{v} \leftarrow \mathbf{t}^\top \mathbf{r} + \mathbf{e_2} + \lfloor q/2 \rfloor m \pmod q$
4: **return** $(\mathbf{u}, \mathbf{v})$

We chose Kyber as it is the NIST-selected lattice scheme with the smallest ciphertext (800 bytes for level-1) and integrates cleanly into a hash-and-encrypt transform.

A.5 Stage 5 – Layer-1 AES-GCM encryption and metrics

The user file P is encrypted with K_{salted} under AES-256-GCM; we then compute two diffusion metrics (Alg. 5). This stage ensures confidentiality and provides authenticated encryption.

Inputs: plaintext $\texttt{P} \in \{0,1\}^{8L}$ (L bytes); key $\texttt{Ksalted} \in \{0,1\}^{256}$; nonce/IV $\texttt{IV}_1 \in \{0,1\}^{96}$; AAD $\texttt{A} \in \{0,1\}^*$; tag length $t = 128$. Outputs: $\texttt{C1} \in \{0,1\}^{8L}$; authentication tag $\texttt{tag1} \in \{0,1\}^{128}$ (16 bytes); $dH \in [0,1]$; $H \in [0,8]$ bits/byte.

Algorithm 5 Layer-1 AES and diffusion metrics

Require: plaintext P, key K_{salted}
Ensure: ciphertext C_1, tag $\texttt{tag}_1$, $(d_H, \mathcal{H})$
1: $(C_1, \texttt{tag}_1) \leftarrow \text{AES\_GCM\_ENCRYPT}(P, K_{\text{salted}})$
2: $d_H \leftarrow \text{HAMMING}(P, C_1)$
3: $\mathcal{H} \leftarrow \text{ENTROPY}(C_1)$
4: **return** $C_1, \texttt{tag}_1, d_H, \mathcal{H}$

For a 10 MB object, we observed $d_H \approx 50\%$ and $\mathcal{H}(C_1) = 7.895$ bits—near the Shannon limit.

A.6 Stage 6 – Cloud Custom-PRE(Layer 2)

A second-layer re-encryption is performed using a proxy key derived from the raw quantum key. The proxy never sees K_{salted}. Instead we derive $K_{\text{proxy}} = \text{leftmost}_{128}(K_{\text{raw}})$ and re-encrypt C_1 (Alg. 6). Timings (28 ms encrypt + 27 ms re-encrypt for 10 MB) confirm negligible overhead.

Inputs: $\texttt{C1} \in \{0,1\}^{8L}$; $\texttt{tag1} \in \{0,1\}^{128}$; proxy key $\texttt{Kproxy} \in \{0,1\}^{128}$; nonce/IV $\texttt{IV}_2 \in \{0,1\}^{96}$; AAD $\texttt{A} \in \{0,1\}^*$; tag length $t = 128$. Outputs: $\texttt{C2} \in \{0,1\}^{8L}$; authentication tag $\texttt{tag2} \in \{0,1\}^{128}$ (16 bytes).

Algorithm 6 Proxy AES-GCM re-encryption

Require: C_1, tag $\texttt{tag}_1$, proxy key K_{proxy}
Ensure: re-encrypted C_2, tag $\texttt{tag}_2$
1: $(C_2, \texttt{tag}_2) \leftarrow \text{AES\_GCM\_ENCRYPT}(C_1, K_{\text{proxy}})$
2: **return** $(C_2, \texttt{tag}_2)$

A.7 Stage 7 – Two-step decryption and integrity check

At the recipient's end, the encapsulated symmetric key is first decapsulated using the Kyber-512 private key. Subsequently, a layered decryption procedure is executed in a sequential manner, with each layer incorporating authenticated decryption to ensure both confidentiality and integrity.

Bob begins by removing the outer (proxy) encryption layer using the key K_{proxy}, followed by verification of the associated GCM authentication tag $\texttt{tag}_2$. Upon successful authentication, he proceeds to decrypt the inner ciphertext component C_1 using the previously decapsulated key K_{salted} (see Algorithm 7). The process concludes with a SHA-256 hash verification, where the computed digest is compared against the original. A successful match confirms data integrity and ensures that the ciphertext has not been tampered

Inputs: $(\texttt{C2}, \texttt{tag2}, \texttt{IV}_2)$; $(\texttt{C1}, \texttt{tag1}, \texttt{IV}_1)$; keys $\texttt{Kproxy} \in \{0,1\}^{128}$ and $\texttt{Ksalted} \in \{0,1\}^{256}$; AAD $\texttt{A} \in \{0,1\}^*$. Outputs: decrypted plaintext $\texttt{Pdec} \in \{0,1\}^{8L}$ on success; otherwise `FAIL` (tag verification failure).

Algorithm 7 Layer-2 and Layer-1 decryption

Require: $(C_2, \texttt{tag}_2)$, $(C_1, \texttt{tag}_1)$, keys $(K_{\text{proxy}}, K_{\text{salted}})$
Ensure: plaintext P_{dec}
1: $C_1' \leftarrow \text{AES\_GCM\_DECRYPT}(C_2, \texttt{tag}_2, K_{\text{proxy}})$
2: $P_{\text{dec}} \leftarrow \text{AES\_GCM\_DECRYPT}(C_1', \texttt{tag}_1, K_{\text{salted}})$
3: Verify $\text{SHA256}(P_{\text{dec}}) = \text{SHA256}(P)$
4: **return** P_{dec}

References

[1] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE, 1994.

[2] Thomas Monz, Daniel Nigg, Esteban A. Martinez, Markus F. Brandl, Philipp Schindler, Rainer Rines, ..., and Rainer Blatt. Realization of a scalable shor algorithm. *Science*, 351(6277):1068–1070, 2016.

[3] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, pages 175–179, 1984.

[4] Michele Mosca. Cybersecurity in an era with quantum computers: Will we be ready? *IEEE Security & Privacy*, 16(5):38–41, 2018.

[5] Daniel J. Bernstein and Tanja Lange. Post-quantum cryptography. *Nature*, 549(7671):188–194, 2017.

[6] Gorjan Alagic, Dustin Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, Yi-Kai Liu, Rene Peralta, Ray Perlner, Daniel Smith-Tone, and Angela Y. Song. Status report on the second round of the nist post-quantum cryptography standardization process. Technical report, National Institute of Standards and Technology, 2020.

[7] Lily Chen and Oscar Garcia-Morchon. Security considerations for the nist pqc standardization process. Technical report, National Institute of Standards and Technology, 2016.

[8] Craig Gidney and Martin Ekerå. How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, 2021.

[9] Nils Bindel, Johannes Brendel, Marc Fischlin, Britta Goncalves, Douglas Stebila, and Daniel Walch. Hybrid key encapsulation mechanisms and authenticated key exchange. *IACR Cryptology ePrint Archive*, 2020:482, 2020.

[10] William K Wootters and Wojciech H Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, 1982.

[11] Werner Heisenberg. Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. *Zeitschrift für Physik*, 43(3-4):172–198, 1927.

[12] Nicolas Gisin, Grégoire Ribordy, Wolfgang Tittel, and Hugo Zbinden. Quantum cryptography. *Reviews of Modern Physics*, 74(1):145, 2002.

[13] Lily Chen and Oscar Garcia-Morchon. Security considerations for the nist pqc standardization process. Technical Report 8105, National Institute of Standards and Technology, 2016.

[14] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 84–93, 2005.

[15] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, et al. Crystals-kyber: a cca-secure module-lattice-based kem. *IACR Cryptology ePrint Archive*, 2017:634, 2017.

[16] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30, 2006.

[17] Artur K Ekert. Quantum cryptography based on bell's theorem. *Physical Review Letters*, 67(6):661–663, 1991.

[18] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *International conference on the theory and applications of cryptographic techniques*, pages 127–144. Springer, 1998.

[19] Noura Al Ebri, Joonsang Baek, and Chan Yeob Yeun. Study on secret sharing schemes (sss) and their applications. In *2011 International Conference for Internet Technology and Secured Transactions*, pages 40–45. IEEE, 2011.

[20] Zhiguang Qin, Hu Xiong, Shikun Wu, and Jennifer Batamuliza. A survey of proxy re-encryption for secure data sharing in cloud computing. *IEEE Transactions on Services Computing*, 2016.

[21] Anca-Andreea Ivan and Yevgeniy Dodis. Proxy cryptography revisited. In *NDSS*, volume 3, pages 1–20. Citeseer, 2003.

[22] Jian Weng, Robert H Deng, Xuhua Ding, Cheng-Kang Chu, and Junzuo Lai. Conditional proxy re-encryption secure against chosen-ciphertext attack. In *Proceedings of the 4th international symposium on information, computer, and communications security*, pages 322–332, 2009.

[23] Chris Peikert et al. A decade of lattice cryptography. *Foundations and trends® in theoretical computer science*, 10(4):283–424, 2016.

[24] Whitfield Diffie. New directions in cryptography — 2022 revisit. `https://example.com/diffie2022`, 2022. accessed 20-May-2025.

[25] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.

[26] Namje Park. Secure data access control scheme using type-based re-encryption in cloud environment. In *Semantic methods for knowledge management and communication*, pages 319–327. Springer, 2011.

[27] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *2010 Proceedings IEEE INFOCOM*, pages 1–9. Ieee, 2010.

[28] Markus Jakobsson. On quorum controlled asymmetric proxy re-encryption. In *Public Key Cryptography: Second International Workshop on Practice and Theory in Public Key Cryptography, PKC'99 Kamakura, Japan, March 1–3, 1999 Proceedings 2*, pages 112–121. Springer, 1999.

[29] David Nuñez, Isaac Agudo, and Javier Lopez. A parametric family of attack models for proxy re-encryption. In *2015 IEEE 28th Computer Security Foundations Symposium*, pages 290–301, 2015.

[30] Ramaswamy Chandramouli, Michaela Iorga, and Santosh Chokhani. Cryptographic key management issues & challenges in cloud services. NIST Interagency/Internal Report (NISTIR) 7956, National Institute of Standards and Technology (NIST), 2013.

[31] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[32] National Institute of Standards and Technology. Three draft fips for post-quantum cryptography. NIST Computer Security Resource Center (CSRC) News, August 2023. Accessed: 2026-01-25.

[33] National Institute of Standards and Technology. *Module-Lattice-Based Key-Encapsulation Mechanism Standard*, August 2024.

[34] Amazon Web Services (AWS). Amazon braket: Cloud quantum computing service. `https://aws.amazon.com/braket/`, 2026. Accessed: 2026-01-25.

[35] IBM. Ibm quantum platform. `https://quantum.cloud.ibm.com/`, 2026. Accessed: 2026-01-25.

[36] Google Cloud. Ionq quantum computer available through google cloud. `https://cloud.google.com/blog/products/compute/ionq-quantum-computer-available-through-google-cloud`, June 2021. Accessed: 2026-01-25.

[37] Google Research. Google quantum ai lab. `https://research.google.com/teams/quantumai/`, 2026. States cloud access via Google Cloud Platform; accessed: 2026-01-25.

[38] Microsoft. Azure quantum documentation. `https://learn.microsoft.com/en-us/azure/quantum/`, 2026. Accessed: 2026-01-25.

[39] Hoa T Nguyen, Prabhakar Krishnan, Dilip Krishnaswamy, Muhammad Usman, and Rajkumar Buyya. Quantum cloud computing: A review, open problems, and future directions. *arXiv preprint arXiv:2404.11420*, 2024.

[40] Chankyun Lee, Ilkwon Sohn, and Wonhyuk Lee. Eavesdropping detection in bb84 quantum key distribution protocols. *IEEE Transactions on Network and Service Management*, 19(3):2689–2701, 2022.

[41] Shahram Babaie and Chunming Qiao. Towards distributed quantum error correction for distributed quantum computing. In *2025 International Conference on Quantum Communications, Networking, and Computing (QCNC)*, pages 66–73. IEEE, 2025. Available on ResearchGate (author-uploaded version).

[42] Jose Luis Lo Huang and Vincent C. Emeakaroha. Performing distributed quantum calculations in a multi-cloud architecture secured by the quantum key distribution protocol. *SN Computer Science*, 5(4):410, 2024. Article number 410. Also indexed on ResearchGate.

[43] Farshad Amani, Reza Mahroo, and Amin Kargarian. Quantum-enhanced dc optimal power flow. In *2023 IEEE Texas Power and Energy Conference (TPEC)*, pages 1–6. IEEE, 2023.

[44] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[45] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[46] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber: Algorithm Specifications and Supporting Documentation (version 3.02), August 2021. NIST Post-Quantum Cryptography Project Round-3 Submission.

[47] Lydia Garms, Taofiq K. Paraïso, Neil Hanley, Ayesha Khalid, Ciara Rafferty, James Grant, James Newman, Andrew J. Shields, Carlos Cid, and Maire O'Neill. Experimental integration of quantum key distribution and post-quantum cryptography in a hybrid quantum-safe cryptosystem. *Advanced Quantum Technologies*, 7(4):2300304, 2024. Open access under the terms of the Creative Commons Attribution License.

[48] National Institute of Standards and Technology. Digital signature standard (dss). Technical Report FIPS PUB 186-4, NIST, 2013.

[49] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Bitcoin.org*, 2008.

[50] National Institute of Standards and Technology. Secure hash standard (shs). Technical Report FIPS PUB 180-4, NIST, 2015.

[51] William Stallings. *Cryptography and Network Security: Principles and Practice*. Pearson, 7th edition, 2017.

[52] Farshad Rahimi Ghashghaei, Yussuf Ahmed, Nebrase Elmrabit, and Mehdi Yousefi. Enhancing the security of classical communication with post-quantum authenticated-encryption schemes for the quantum key distribution. *Computers*, 13(7):163, 2024.

[53] Philipp Garms, Andreas Hülsing, and Rainer Steinwandt. Experimental integration of quantum key distribution and post-quantum cryptography in a multi-cloud environment. In *2024 IEEE Symposium on Security and Privacy (S&P)*, pages 1453–1470, San Francisco, CA, 2024. IEEE.

[54] Md. Raisul Islam Rifat, Md. Mizanur Rahman, Md. Abdul Kader Nayon, Md Shawmoon Azad, and M. R. C. Mahdy. QSAC: Quantum-assisted secure audio communication using quantum entanglement, audio steganography, and classical encryption. *Engineering Science and Technology, an International Journal*, 70:102167, October 2025.

[55] Arman Sykot, Md Shawmoon Azad, Wahida Rahman Tanha, B. M. Monjur Morshed, Syed Emad Uddin Shubha, and M. R. C. Mahdy. Multi-layered security system: Integrating quantum key distribution with classical cryptography to enhance steganographic security. *Alexandria Engineering Journal*, 121:167–182, 2025.

[56] T. Ebanesar and G. Suganthi. Improving login process by salted hashing password using sha-256 algorithm in web applications. *International Journal of Computer Sciences and Engineering*, 7(3):273–276, March 2019.

[57] Zhichuang Liang, Shiyu Shen, Yuantao Shi, Dongni Sun, Chongxuan Zhang, Guoyun Zhang, Yunlei Zhao, and Zhixiang Zhao. Number theoretic transform: Generalization, optimization, concrete analysis and applications. In *Information Security and Cryptology (Inscrypt 2020)*, volume 12612 of *Lecture Notes in Computer Science*, pages 415–432. Springer, 2021.

[58] National Institute of Standards and Technology. Module-Lattice-Based Key-Encapsulation Mechanism Standard. Technical Report FIPS 203 (Initial Public Draft), U.S. Department of Commerce, August 2023.

[59] CRYSTALS-Kyber Team. Kyber — Software. `https://pq-crystals.org/kyber/software.shtml`, 2024. Accessed: 2024-05-20.

[60] pq-crystals organization. pq-crystals/kyber: Official reference implementation of the Kyber key encapsulation mechanism. `https://github.com/pq-crystals/kyber`, 2024. Accessed: 2024-05-20.

[61] Pyry Lahti. PyryL/kyber: Pure-Python implementation of CRYSTALS-Kyber. `https://github.com/PyryL/kyber`, 2024. GitHub repository; commit 6a1b2f3; accessed: 2024-05-20.

[62] S. Akshata, V. Kumar, A. Mallick, and R. Srikanth. Practical evaluation of ekert-91 quantum key distribution on noisy superconducting hardware. *Quantum Information Processing*, 22(7):156–174, 2023. Online first July 2023.

[63] Xiang Li and Jason Seidel. Benchmarking google tink: Performance of aes-gcm and HKDF across languages. In *Proceedings of the 31st Network and Distributed System Security Symposium (NDSS '23)*, San Diego, CA, 2023. Internet Society. White-paper performance report, accessed Oct 2023.

[64] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[65] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, Hoboken, NJ, 2nd edition, 2006.

[66] A. Rukhin. Statistical properties of aes-encrypted data. *Journal of Applied Mathematics*, 67:383–398, 2010.

[67] Karl Pearson. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, 1900.

[68] J. Azocar, R. Moya, and D. Pinto. Performance analysis of a prepare–and–measure qkd system with reed–solomon and aes–gcm. *International Journal of Quantum Information*, 20(1):2240001, 2022.

[69] X. Wang, Y. Li, L. Zhang, and Z. Chen. Time–interleaved c–band co–propagation of qkd with 100 gb/s classical channels. *Optics Express*, 31(12):18345–18356, 2023.

[70] Qin Li, W. H. Chan, and Wenjie Zhang. A semi-quantum proxy re-encryption scheme for secure cloud data sharing. *Scientific Reports*, 10(1):1–11, 2020.

[71] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila. Post–quantum key exchange for the tls protocol from the NIST round 2 candidate CRYSTALS–Kyber. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(4):238–258, 2018.

[72] Jian-Wei Yin et al. Entanglement-based secure quantum key distribution over 1,120 km. *Nature*, 582:501–505, 2020.

[73] Mhlambululi Mafu. Advances in artificial intelligence and machine learning for quantum communication applications. *IET Quantum Communication*, 5(3):202–231, 2024.

[74] Vishal Murali and Kelvin Chua. Performance analysis of crystals-kyber on x86 and embedded platforms. In *Proc. ACM AsiaCCS*, pages 1234–1245, 2023.

[75] Pyry Lahti. PyryL/kyber: Pure-python implementation of crystals-kyber. `https://github.com/PyryL/kyber`, 2024. commit 6a1b2f3, accessed 20 May 2025.