International Journal of Creative and Open Research in Engineering and Management
ISSN: 3108-1754 (Online)
Volume 02 Issue 11 November-2026 | Impact Factor: 3.5



Evaluating a Simulated BB84 Key-Establishment Pipeline with Adaptive Key Management and AES-256-GCM File Transfer: An Implementation-Based Study

Dr. Anil Kumar B
Assistant Professor
Dept. of CSE, School of Engineering,
RV Institute of Technology and Management, Bengaluru, India

Nandini G V
Dept. of CSE, School of Engineering,
RV Institute of Technology and Management, Bengaluru, India

Narasani Chiruhaas Reddy
Dept. of CSE, School of Engineering,
RV Institute of Technology and Management, Bengaluru, India

Omkar Bhowmick
Dept. of CSE, School of Engineering,
RV Institute of Technology and Management, Bengaluru, India

Bhuvana M L
Dept. of CSE, School of Engineering,
RV Institute of Technology and Management, Bengaluru, India

How to Cite this Article:
V, N. G., Reddy, N. C., Bhowmick, O. & L, B. M. (2026). Evaluating a Simulated BB84 Key-Establishment Pipeline with Adaptive Key Management and AES-256-GCM File Transfer: An Implementation-Based Study. International Journal of Creative and Open Research in Engineering and Management, 02(11), 1-9. https://doi.org/10.55041/ijcope.v2i11.139

License:
This article is published under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.
© The Author(s). Published by International Journal of Creative and Open Research in Engineering and Management.







https://doi.org/10.55041/ijcope.v2i11.139

Abstract*—Quantum key distribution (QKD) is one proposed way to protect key exchange against future quantum computers, but real QKD needs specialised hardware, and many software studies stop at protocol simulation. This paper evaluates a Python platform that links a simulated BB84 key-establishment protocol to adaptive key management and AES-256-GCM encrypted file transfer. The quantum exchange is simulated with Qiskit Aer, with optional bit-flip channel noise and an intercept-resend eavesdropper. A round is accepted only if the measured quantum bit error rate (QBER) is at most 11%, at least eight secret bits remain, and the retained bits of both parties agree; accepted rounds give a 256-bit key through SHA-256. A key manager applies a 60-second time to live, a two-use limit and manual revocation, and files are handled by an authenticated FastAPI service. In 32-qubit experiments with 30 trials per setting, the mean QBER rose from 0.00% to 19.28% as configured noise increased from 0% to 25%, while the acceptance rate fell from 100% to 36.7%. Under intercept-resend the mean QBER was 28.78%, yet 4 of 30 attacked rounds were still accepted (29 of 30 clean rounds), so detection is not guaranteed. AES-256-GCM had lower mean timings than Fernet at every tested payload size. Fifteen automated tests passed. The acceptance rates were recorded before the final retained-bit agreement check was added to the code. Error correction and privacy amplification are not implemented, so the platform is a simulation-based teaching and evaluation tool, not a quantum-secure production system.

***Index Terms*—Quantum Cryptography, BB84, Quantum Key Distribution, QBER, AES-256-GCM, Adaptive Key Management, Secure File Transfer.**

I. Introduction

Many key-exchange and digital-signature methods used on the internet depend on mathematical problems that a large quantum computer could solve efficiently. For this reason, standards bodies are planning a transition to post-quantum cryptography [1]. A recent study commissioned by the UK government reports that post-quantum cryptography is currently the preferred response, while quantum key distribution (QKD) is mainly used in research, testbeds and pilot deployments [2].

QKD lets two parties build a shared random key whose secrecy is checked by looking for the disturbance that an eavesdropper would cause to quantum signals. The first QKD protocol, Bennett–Brassard 1984 (BB84) [3], is still the most widely deployed one [2]. QKD only distributes keys: the key must still be used with a classical cipher such as the Advanced Encryption Standard (AES), and the communicating parties must be authenticated by separate means [2]. A recent review also lists scalability, low key-generation rates, cost and standardisation among the main obstacles to deployment [4].

Because real quantum links need costly, specialised hardware, software simulation is widely used to study protocol behaviour under noise and attack [5]. However, many simulation studies stop at the error-rate estimate and do not show how a simulated key reaches an encrypted application. This paper reports an implementation-based study of a Python platform that addresses this gap in a small, simple setting. The platform simulates BB84 with Qiskit, derives a 256-bit key from accepted rounds, manages the lifetime of that key, and uses it to encrypt files with AES-256 in Galois/Counter Mode (GCM) through an authenticated web service. We also state clearly what this simulation does and does not guarantee.

The main contributions are: (1) a Qiskit Aer simulation of BB84 with optional channel noise and intercept-resend eavesdropping; (2) a round acceptance rule and key derivation step; (3) an adaptive key manager with expiry, use limits, revocation and an audit log; (4) AES-256-GCM file transfer with SHA-256 integrity checking behind an authenticated FastAPI service and browser interface; and (5) repeated experiments on noise, eavesdropping and cipher speed, together with a list of limitations found by reading the source code. The work is not a new protocol and is not presented as quantum-secure for production use; it combines existing techniques for teaching and evaluation.

Section II reviews related work. Section III describes the architecture and methods, Section IV the experimental setup, and Section V the results. Section VI discusses limitations, and Section VII concludes.



II. Related Work

A. QKD Protocols and Their Security

BB84 encodes random bits in one of two bases, so a measurement in the wrong basis gives a random result. An eavesdropper who measures and resends the signals disturbs them, and the legitimate parties detect this by comparing a sample of their results [3], [6]. A complete key-establishment chain also needs error correction and privacy amplification [6], [7], and what can be claimed for a system depends on finite key lengths and on the behaviour of real devices [6]. The UK study adds that QKD does not authenticate the communicating parties and that theoretical guarantees do not automatically apply to real systems [2].

B. Simulation Studies

Rasay et al. simulate BB84, B92 and E91 with Qiskit and study how the error rate changes with noise and attack strength; they also note that simplified noise models may give optimistic results [5]. Sharma et al. simulate an enhanced BB84 with a reconciliation method that avoids revealing key bits [8]. Both works stay at the protocol level and do not include an encrypted application, which is the part our project adds.

C. Secure File Transfer

Mozo combines a simulated BB84 exchange with AES-256 in cipher-block-chaining mode, a message authentication code, optional post-quantum signatures and a graphical interface, and tests it with 16 fault-injection cases [9]. Popa et al. transfer files with keys from a physical QKD link, using a one-time pad and a message authentication code [10]. Sola-Thomas and Imtiaz secure file transfer with post-quantum cryptography and a blockchain audit trail, without QKD [11]. Our platform is closest to [9] in aim, but it uses AES-256-GCM with bound metadata, a key lifecycle policy and an authenticated web service.

D. Standards

NIST SP 800-38D specifies GCM as an authenticated encryption mode and sets rules for nonces and for how often one key may be used [12]. NIST SP 800-57 describes the key lifecycle, including generation, use periods, expiry and revocation [13]. We use these documents as references for our encryption and key-management design but do not claim conformance to either.

III. System Architecture and Methodology

A. Overview

Fig. 1 shows the layered design. The browser interface calls a FastAPI service, which checks the user's login token and passes each request to the BB84 simulator, the adaptive key manager or the secure transfer service. The transfer service uses the AES-256-GCM encryption engine. Keys, round history, transfer records and the audit log are held in memory.

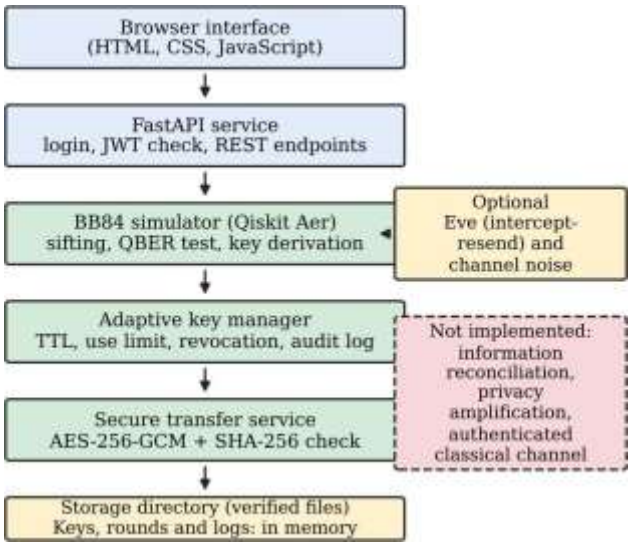

Fig. 1. Architecture of the platform. Green boxes are the security modules; the dashed box lists stages that are not implemented.

A file transfer proceeds as follows. The user logs in and uploads a file of up to 15 MB. If no usable key exists, the service runs one BB84 round with 48 qubits, no eavesdropper and no noise; otherwise it reuses an active key. The key manager checks the key's expiry and use count. The file is encrypted with AES-256-GCM and then decrypted again, and the SHA-256 hashes of the original and decrypted data are compared. If they match, the file is stored, marked VERIFIED and made available for download. The sender-to-receiver step is simulated inside one process; no transfer between separate machines was evaluated.

B. Simulated BB84 Round

BB84 uses two measurement bases, called computational and diagonal. If a qubit is measured in the wrong basis, the result is random, which is why an eavesdropper who measures the qubits disturbs them. One simulated round (Fig. 2) works as follows. Alice generates n random bits and bases and prepares one qubit per bit, using an X gate for bit 1 and a Hadamard gate for the diagonal basis. If enabled, Eve measures each qubit in a random basis and resends it (intercept-resend), and channel noise applies an X gate to each qubit with probability p. Bob then measures each qubit in a random basis in a single-shot run on the Aer simulator.

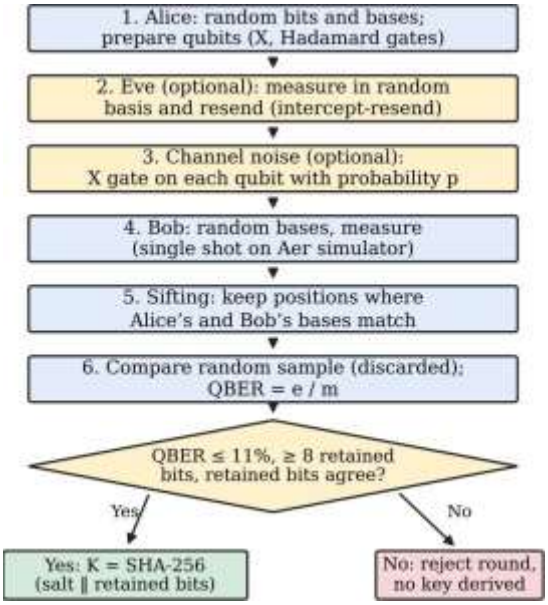

Fig. 2. Flow of one simulated BB84 round and the accept/reject decision.

In sifting, Alice and Bob keep the positions where their bases match, which is about half of the positions. A random sample of the sifted bits (33% by default) is compared in public and then discarded. If e of the m compared bits differ, the QBER is

$$QBER = e / m \quad (1)$$

A round is accepted only if the QBER is at most the threshold (11% by default), at least eight secret bits remain, and Alice's and Bob's retained bits are identical. For an accepted round, the 256-bit key is

$$K = \text{SHA-256}(\text{salt} \,\|\, \text{retained bits}) \quad (2)$$

where the salt is a public value. Two theoretical expectations help in reading the results; they are not experimental findings. In an intercept-resend attack, Eve uses the wrong basis half of the time and then causes an error in half of those results, so the expected QBER is about 25% [6]. In our noise model the X gate changes the result only in the computational basis, so the expected QBER is lower than the configured noise probability (roughly half of it).

C. Adaptive Key Management

Each accepted key is registered with an identifier, a fingerprint (the first 16 hexadecimal characters of its SHA-256 digest), an expiry time and a use limit. Three controls decide whether a key may be used: a time to live (default 60 s, allowed range 5–3600 s), a maximum number of uses (default 2, allowed range 1–50) and manual revocation, which also clears the stored key. A key is ACTIVE until it expires, reaches its use limit (DEPLETED) or is revoked; keys in any other state are refused. An audit log records key creation, use, refusal and revocation. These controls follow the key lifecycle idea in [13], and the use limit also bounds how often one key is used with randomly chosen nonces [12].

D. Authenticated File Encryption

Files are encrypted with AES-256-GCM, which provides confidentiality and detects any modification. For a key K, nonce N, plaintext P and associated data A, encryption produces a ciphertext C and an authentication tag T:

$$(C, T) = \text{AES-GCM-Encrypt}(K, N, P, A) \quad (3)$$

A fresh random 12-byte nonce is used for every encryption and is stored with the ciphertext. The associated data is built from the transfer identifier, sender and receiver, so a ciphertext cannot be validated in another context. Decryption fails if the tag is invalid. Files are limited to 15 MB and file names are sanitised. Fernet, which combines AES in cipher-block-chaining mode with a hash-based authentication code, is used only as a comparison in the timing experiment.

E. User Authentication

Users log in with a password that is stored as a bcrypt hash. A successful login returns a JSON Web Token (JWT) signed with HS256 that expires after 24 hours, and every endpoint except the health check requires it. This protects the web service and is separate from authenticating the classical channel of the BB84 protocol, which is not implemented. The platform has a single demonstration account.

IV. Experimental Setup

A. Environment and Procedure

The platform was written in Python 3.13.0 using Qiskit 2.4.1, Qiskit Aer 0.17.2, the cryptography package 46.0.6 and FastAPI 0.143.0. The experiments were run with a script that calls the unmodified simulator and cipher functions and saves every trial to a CSV file; the graphs in Section V were drawn from these files. The processor and memory of the computer were not recorded, so timing results are given without a hardware specification. Three experiments were performed (Table I).

TABLE I
Experiments Performed

Id	Objective	Settings
A	Noise versus QBER and acceptance	32 qubits, no eavesdropper; configured noise 0, 5, 10, 15, 25%; 30 trials per level (150 records)
B	Clean versus intercept-resend	32 qubits, no noise; eavesdropper off and on; 30 trials per condition (60 records)
C	Cipher timing	AES-256-GCM and Fernet; encrypt and decrypt; 1, 10, 100, 1000, 5000 KB; 5 warm-up then 30 timed repetitions (600 records)

B. Metrics and Analysis

For Experiments A and B we recorded the measured QBER and the accept/reject decision of every trial; for Experiment C we recorded the time of every single encryption or decryption call in milliseconds. All reported values are arithmetic means or counts of these records. No statistical tests were performed. The simulator's random numbers are not seeded, so repeated runs reproduce the overall behaviour but not individual values.



C. Automated Testing

The implemented system includes 15 automated tests: 12 for the web platform (simulator, cipher, key manager, transfer service and API) and 3 for an earlier prototype. All 15 passed in the recorded run. The tests check functions such as tamper detection, unique nonces, use-limit depletion, revocation and an end-to-end transfer with matching hashes. They do not establish cryptographic security and do not cover key expiry by elapsed time.

V. Results and Discussion

A. Effect of Channel Noise

Fig. 3 shows the mean measured QBER at each configured noise level, and Table II lists the values. The QBER was 0.00% with no noise and 19.28% at 25% noise. The curve does not rise steadily: the mean at 10% noise (3.28%) is slightly above that at 15% (3.03%). This is expected because the QBER is estimated from only a few sampled bits per round and each level has only 30 trials. At every nonzero setting the measured QBER stayed below the configured noise level, which is consistent with a noise model that affects only one basis, although the means are not expected to match the theoretical value closely.

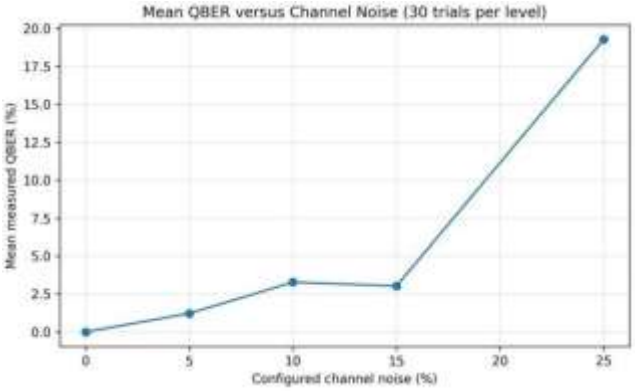

Fig. 3. Mean measured QBER versus configured channel-noise level (30 trials per level).

Fig. 4 shows the percentage of accepted rounds. Acceptance fell from 100.0% at 0% noise to 36.7% at 25% noise. A round can also be rejected when the measured QBER is 0%, because at least eight retained secret bits are required. The acceptance rate therefore reflects both the QBER test and the minimum retained-bit condition.

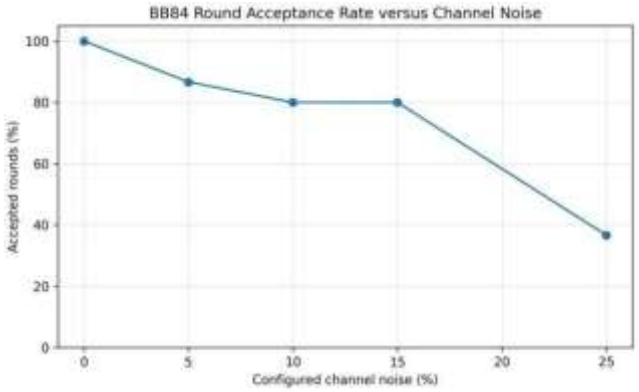

Fig. 4. BB84 round acceptance rate versus configured channel-noise level (30 trials per level).

TABLE II
Noise Experiment Results (30 Trials per Level)

Configured noise (%)	Mean QBER (%)	Accepted rounds (%)
0	0.00	100.0
5	1.22	86.7
10	3.28	80.0
15	3.03	80.0
25	19.28	36.7

B. Intercept-Resend Eavesdropping

Fig. 5 compares a clean channel with the intercept-resend condition. On the clean channel the mean QBER was 0.00% and 29 of 30 rounds were accepted. Under intercept-resend the mean QBER was 28.78%, close to the theoretical expectation of about 25%, and 4 of 30 rounds were still accepted. The simulated attack therefore raised the error rate in this run, but detection was not guaranteed: because only a small random sample is compared, some attacked rounds showed no errors. One clean round and one attacked round were rejected despite a QBER of zero because fewer than eight secret bits were retained, so rejection should not be read as detection of an attack by QBER alone.

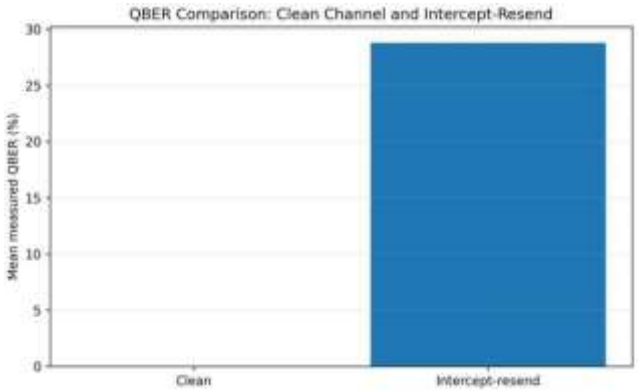

Fig. 5. Mean measured QBER under clean-channel and intercept-resend conditions (30 trials each).

C. Encryption and Decryption Time

Fig. 6 and Table III compare AES-256-GCM with Fernet. Each value is the mean of 30 timed repetitions. AES-256-GCM

had a lower mean time than Fernet for both operations at all five payload sizes. For the 5000 KB payload, mean encryption took 5.94 ms with AES-256-GCM and 38.97 ms with Fernet. These values depend on the computer, the software libraries and the run conditions, so they are not general performance guarantees.

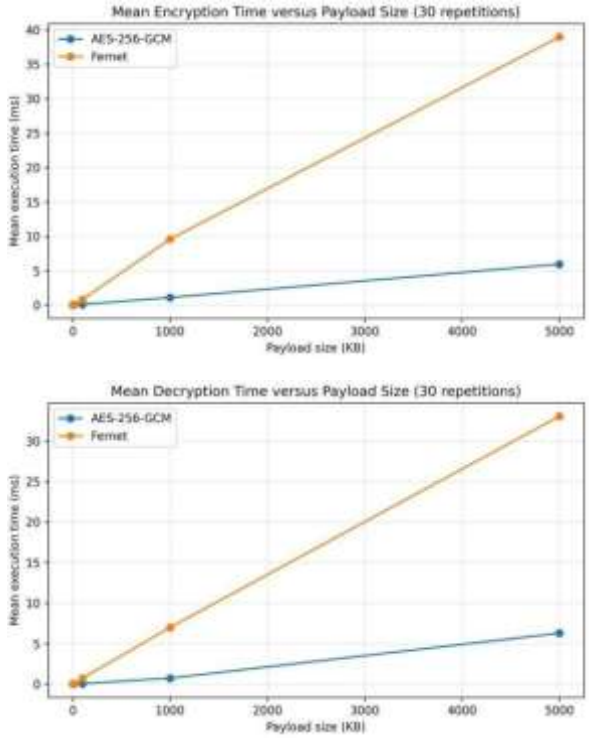

Fig. 6. Mean (a) encryption time and (b) decryption time of AES-256-GCM and Fernet versus payload size (30 repetitions per point).

TABLE III
MEAN ENCRYPTION AND DECRYPTION TIME (MS)

Payload (KB)	AES-GCM enc.	Fernet enc.	AES-GCM dec.	Fernet dec.
1	0.0073	0.0331	0.0048	0.0275
10	0.0055	0.0644	0.0052	0.0452
100	0.0914	0.7947	0.0445	0.7138
1000	1.0843	9.5758	0.7303	6.9924
5000	5.9396	38.9728	6.2805	33.0672

D. Discussion

The experiments support three limited findings. First, in the simulator the measured QBER was higher at higher noise levels overall, and acceptance depended on both the QBER and the number of retained bits. Second, intercept-resend raised the mean QBER compared with the clean channel, but a short round of 32 qubits does not detect every attack. Third, AES-256-GCM was faster than Fernet in this environment. Together with the 15 passing tests, these results show that the simulated BB84 stage, the key manager and the encrypted transfer service work together as intended. They do not show that the system is secure against real attackers.

VI. LIMITATIONS

The limitations below were identified from the source code and the experiments. Table IV summarises what each implemented protection does and where it falls short.

1) *Simulation only.* No photons, detectors or optical channels are used. Only intercept-resend attacks and an X-gate noise model are implemented, and the noise model affects only one basis.
2) *Missing post-processing.* Information reconciliation, privacy amplification and authentication of the classical channel are not implemented. The key is a single SHA-256 hash of a public salt and the retained bits, and a hash cannot add secret information. A derived 256-bit key therefore holds no more secret information than the few retained bits supplied to it [6], [7]. The receiver also does not derive its own key; the transfer service reuses the key held by the key manager.
3) *Fixed threshold.* The 11% threshold is an asymptotic value cited in the literature [6]. It has no finite-size security justification for rounds of only tens of qubits.
4) *System design.* Keys, records and logs are kept in memory and lost on restart, the sender-to-receiver step runs in one process, and a single demonstration account with default credentials is provided. The round endpoint also returns Alice's bits and bases and Bob's bases and measured bits to the logged-in user. This exposure was found in the source code and was not tested.
5) *Experiments.* The experiments used 32-qubit rounds, 30 trials per setting and one computer whose specification was not recorded. The data for Experiments A and B were recorded before the retained-bit agreement rule was added to the final code. Of the 150 records of Experiment A, 37 are accepted rounds with at least one retained-bit mismatch, which the final rule would reject. The acceptance rates in Section V therefore describe the earlier rule (QBER threshold and at least eight retained bits), and a re-run with the final code is expected to give lower acceptance at nonzero noise.

TABLE IV
IMPLEMENTED PROTECTIONS AND REMAINING GAPS

Threat	Implemented measure	Remaining gap
Altered ciphertext	AES-GCM tag; SHA-256 comparison	Tested only in unit tests; no network setting
Nonce reuse	Fresh random 12-byte nonce per call	Nonce-count limits of GCM not analysed; uniqueness checked by a unit test only
Old or overused key	TTL, use limit, revocation	Expiry by time not covered by tests; keys kept in memory
Unauthorised access	bcrypt hashes; JWT on endpoints	Single demo account, default credentials
Eavesdropping	QBER threshold test	4 of 30 attacked rounds accepted; no privacy amplification



VII. Conclusion and Future Work

This paper presented a Python platform that connects a simulated BB84 exchange to adaptive key management and AES-256-GCM encrypted file transfer. In 32-qubit experiments, the measured QBER and the acceptance rate responded to channel noise, intercept-resend eavesdropping raised the mean QBER to 28.78% although 4 of 30 attacked rounds were still accepted, and AES-256-GCM was faster than Fernet at all tested payload sizes. Fifteen automated tests passed. Because BB84 is only simulated and no error correction or privacy amplification is implemented, the platform is a teaching and evaluation tool and not a quantum-secure production system.

Future work includes adding information reconciliation and privacy amplification with a key length justified by the amount of secret information; deriving the key independently at the receiver and removing key-related data from interface responses; authenticating the classical channel and testing transfers across a real network; using richer noise and attack models; storing key state persistently and protecting it; and repeating the experiments with the final acceptance rule, more trials, several qubit counts and statistical tests. Connecting the platform to a hardware or emulated key-delivery interface would allow simulated and physical keys to be compared.

References

[1] National Institute of Standards and Technology, "Transition to post-quantum cryptography standards (initial public draft)," NIST Interagency Rep. 8547, Nov. 2024. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2024/NIST.IR.8547.ipd.pdf

[2] Department for Science, Innovation and Technology, "Quantum key distribution research report," GOV.UK, Jun. 10, 2026. [Online]. Available: https://www.gov.uk/government/publications/quantum-key-distribution-research-report/quantum-key-distribution-research-report

[3] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theor. Comput. Sci.*, vol. 560, pp. 7–11, 2014, doi: 10.1016/j.tcs.2014.05.025.

[4] A. Saji and A. Pratap, "Trends in quantum key distribution technologies: A systematic review of protocols, network architectures, and deployment challenges," *Int. J. Comput. Inf. Syst. Ind. Manag. Appl.*, vol. 18, no. 23s, pp. 460–479, 2026, doi: 10.70917/ijcisim-2026-5587.

[5] M. Rasay, E. D. Sebastian, S. P. Sah, D. C. Akah, and A. K. Singh, "Secure quantum communication: Simulation and analysis of quantum key distribution protocols," arXiv:2603.16690, 2026.

[6] V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf, M. Dusek, N. Lutkenhaus, and M. Peev, "The security of practical quantum key distribution," *Rev. Mod. Phys.*, vol. 81, pp. 1301–1350, 2009, doi: 10.1103/RevModPhys.81.1301.

[7] C. H. Bennett, G. Brassard, and J.-M. Robert, "Privacy amplification by public discussion," *SIAM J. Comput.*, vol. 17, pp. 210–229, 1988, doi: 10.1137/0217014.

[8] N. Sharma, V. Saxena, V. Chamola, and V. Hassija, "Key reconciliation protocol for quantum key distribution," *EPJ Quantum Technol.*, vol. 12, Art. 21, 2025, doi: 10.1140/epjqt/s40507-025-00319-4.

[9] H. E. Mozo, "Quantum-classical hybrid encryption framework based on simulated BB84 and AES-256: Design and experimental evaluation," arXiv:2511.02836, 2025.

[10] A.-B. Popa, B.-C. Ciobanu, and P. G. Popescu, "Practical implementation of unconditionally secure file transfer application with QKD and OTP," *Quantum Rep.*, vol. 7, no. 1, Art. 12, 2025, doi: 10.3390/quantum7010012.

[11] E. Sola-Thomas and M. H. Imtiaz, "Development of a quantum-resistant file transfer system with blockchain audit trail," arXiv:2504.07938, 2025.

[12] M. Dworkin, "Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC," NIST Spec. Publ. 800-38D, Nov. 2007, doi: 10.6028/NIST.SP.800-38D.

[13] E. Barker, "Recommendation for key management: Part 1 – General," NIST Spec. Publ. 800-57 Part 1 Rev. 5, May 2020, doi: 10.6028/NIST.SP.800-57pt1r5.