Software Defined Network Based Error Correction Mechanism in Quantum Key Distribution

Rasha Shajahan
Center for Cybersecurity Systems and Networks
Amrita Vishwa Vidyapeetham
Amritapuri-campus, India
rashajahan2684@gmail.com

Dr. Kurunandan Jain
Center for Cybersecurity Systems and Networks
Amrita Vishwa Vidyapeetham
Amritapuri-campus, India
kurunandanj@am.amrita.edu,

Aneesh Kumar K B
Cyber Security Group
Center for Development of Advanced Computing
Thiruvanathapuram, Kerala, India
aneesh_kb@cdac.in,

Dr. Prabhakar Krishnan
Center for Cybersecurity Systems and Networks
Amrita Vishwa Vidyapeetham
Amritapuri-campus, India
kprabhakar@am.amrita.edu,

***Abstract*—Quantum Key Distribution (QKD) is a secure communication method that uses quantum mechanics to exchange secret keys. Currently, the QKD process uses a static error correction method, which can lead to inaccuracies and inefficient resource use. The project aims to develop a new QKD error correction method that adapts dynamically based on real-time Quantum Bit Error Rate (QBER) measurements and Software Defined Networking (SDN) principles. Two specific error correction mechanisms, the 2D Parity check and the Cascade Protocol, will be used to match the QBER value. The investigation will be conducted using the DARPAN application, providing a quantum-enabled environment.**

***Keywords*—Quantum Computing, Quantum Cryptography, Quantum Key Distribution (QKD), Quantum Bit Error Rate (QBER), Software Defined Networking (SDN), 2D Parity Check, Cascade Protocol**

I. Introduction

Quantum computing is revolutionizing traditional cryptography, posing significant threats to classical encryption methods with its immense computational power. Unlike classical computers, quantum computers use qubits, which can exist in multiple states simultaneously due to superposition and entanglement. This allows quantum computers to perform complex calculations, such as factoring large numbers or simulating quantum systems, much faster than classical computers. As a result, classical cryptographic algorithms are vulnerable to rapid decryption by quantum computers, necessitating the development of quantum-resistant encryption techniques to protect sensitive data.

To address this challenge, quantum cryptography offers a robust solution by using the principles of quantum mechanics to secure communication protocols. Central to this approach is Quantum Key Distribution (QKD), which ensures the confidentiality and integrity of secret keys transmitted between parties. QKD leverages principles like the Heisenberg Uncertainty Principle and the no-cloning theorem to guarantee security. The Heisenberg Uncertainty Principle states that any attempt to measure a quantum system disrupts it, enabling the detection of eavesdropping [12]. The no-cloning theorem further enhances security by preventing the replication of an unknown quantum state, thus safeguarding the key exchange process [12].

When disturbances occur in the quantum channel due to noise, hardware faults, or eavesdropping, the polarization states received by the receiver may be incorrect, leading to errors. During the key reconciliation phase in QKD, these errors can be detected and partially corrected. However, the current QKD systems use a static error correction mechanism, which can be inefficient. Low-efficiency methods may fail with many errors, while high-efficiency methods may waste resources when errors are few. This rigid approach does not adapt to the actual error rate, resulting in inaccuracies and inefficient resource use.

To optimize error correction, a dynamic method should be adopted, using the Quantum Bit Error Rate (QBER) to tailor the error correction technique to the current error rate. This adaptive approach ensures that the system uses resources efficiently and maintains high accuracy, significantly enhancing the overall security and efficiency of the QKD process.

II. Background

A. Quantum Key Distribution

Quantum Key Distribution (QKD) has emerged as a crucial solution in addressing the threat posed by Quantum Computing to public key cryptography. Its main goal is to establish and distribute symmetric cryptographic keys among users who are physically separated, ensuring security in a quantum-safe manner, based on the fundamental laws of physics rather than computational complexity. Initially, introduced by Bennett and Brassard in 1984, with successful implementation achieved in 1989 [4], QKD functions through two channels: the quantum

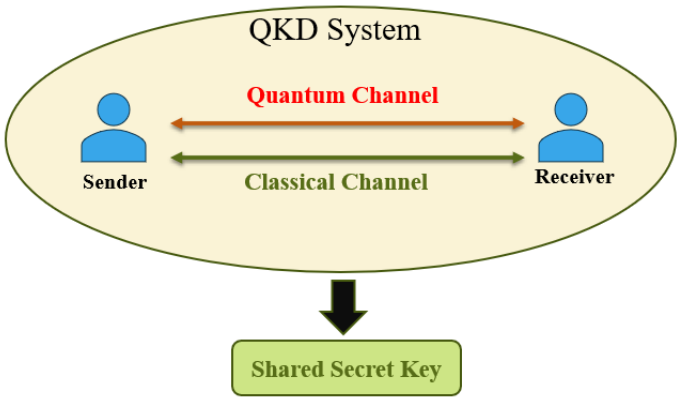

Fig. 1. QKD System

channel, which transmits secret key information in the form of polarized photons or qubits, and the public channel, facilitating discussions on qubit transmission and agreement on the shared secret key. Quantum channels are typically implemented using optical fiber or free space, but the distance and rate limitations of QKD links are influenced by both quantum/optical and public/classical channel usage. QKD protocols are categorized into discrete-variable, continuous-variable, and distributed-phase-reference coding, each differing in photon preparation and generation methods over the quantum channel. There are two primary QKD protocol schemes [4]: prepare-and-measure-based and entanglement-based. In the former, the sender prepares the information, and the receiver measures it, utilizing Heisenberg's uncertainty principle and the no-cloning theorem [12] to detect eavesdropping attempts through error parameter measurement. Conversely, entanglement-based protocols use entanglement photon principles for key distribution.

B. QKD Process Flow

Quantum Key Distribution encompasses two primary stages: the quantum channel and the classical channel. The quantum channel initiates the key sharing process as the sender transmits encoded quantum state raw bits to the receiver using polarization states. The subsequent phase, called post-processing, occurs after this initial key exchange and comprises the following stages:

1) Key Sifting: During the key sifting phase, both the sender and receiver select bit values from predetermined positions, creating what is known as the sifted key. This sifted key is then used to calculate the Quantum Bit Error Rate (QBER) and is discarded afterward.

2) Key Reconciliation: The main purpose of the key reconciliation process is to detect and remove errors from the key. In practical situations, errors can occur due to eavesdropping, device imperfection, or the inference from environment conditions [10]. In this phase, we have two subphases:

a) Parameter Estimation: At this juncture, the error rate of the sifted data is assessed through Quantum Bit Error Rate (QBER), which measures the ratio of errors within the key. The QBER provides insights into potential eavesdropping and the extent of information that may have been compromised.

b) Error Correction: Concentrate on identifying and correcting errors in the sifted data using various error correction methods, such as LDPC (Low-Density Parity-Check), 2D Parity Check, and Cascade protocols. The result of this process is an encoded codeword, which crucially does not involve sending the actual error-corrected sifted key to the receiver, nor does it add extra bits to the sifted key, providing a significant benefit.

3) Privacy Amplification: The privacy amplification phase seeks to remove any residual information that may have been exposed to an adversary during the post-processing phase.

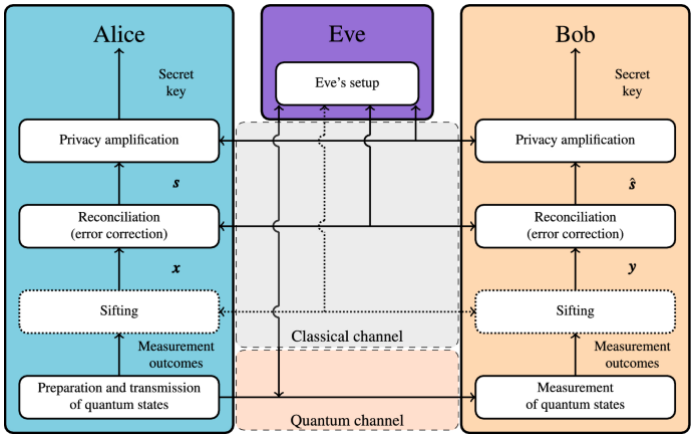

Fig. 2. QKD process with Alice, Bob and Eve (intruder).

III. Methodology

A. System Architecture

In our experiment, we've developed a simulated quantum environment to delve into the Quantum Key Distribution (QKD) process. The setup, once configured, is initiated through the MAQN controller, which ensures the smooth operation of the simulation platform. The simulation runs within a VMware environment, utilizing C++ for simulating the quantum channel and implementing error correction mechanisms, essential for the integrity of QKD communications. The experiment further involves the DARPAN application, which simulates the QKD process with nodes named Alice and Bob, facilitating secure communication over an IP network. A significant feature of our setup is the integration of classical and quantum channels through a tap bridge, enabling seamless communication and key exchange.

B. System Configuration

To run the Quantum Key Distribution (QKD) process effectively, several foundational and strategic steps have been configured, ensuring a robust and secure environment for the QKD operations. Each component plays a critical role in the setup, enabling seamless execution and management of QKD tasks.

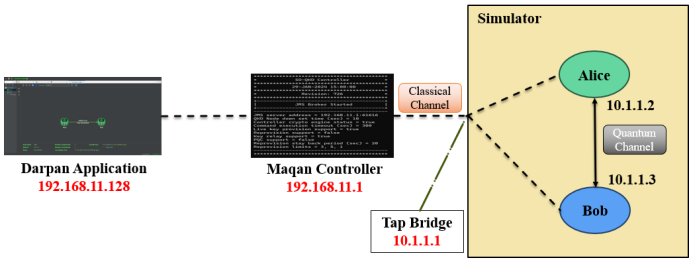

Fig. 3. System Architecture

1) Apache Tomcat v10 Installation: Apache Tomcat serves as a web server and servlet container that allows us to deploy and run web applications. For the QKD process, Tomcat is essential for hosting web-based interfaces or applications that facilitate user interaction with the QKD system, providing a platform for management and monitoring tasks.

2) PostgreSQL v15 with PGAdmin Installation: PostgreSQL is a powerful, open-source database management system used for storing, retrieving, and managing the data generated during the QKD process. This could include key material, configuration settings, and operational logs. PGAdmin is a graphical database management tool for PostgreSQL, simplifying database administration tasks such as monitoring, troubleshooting, and performance tuning.

3) Amazon Corretto (Java Environment) Installation: Amazon Corretto is a no-cost, multiplatform, production-ready distribution of the Open Java Development Kit (OpenJDK). It ensures that the QKD system has a reliable and secure Java environment for running Java-based applications and services, which are often integral parts of the QKD process, including cryptographic functions and network communication.

4) Configuration Setup for MAQAN Controller: The MAQAN Controller is likely a specialized component designed for managing and orchestrating the QKD process, possibly involving key management, policy enforcement, and interaction with quantum and classical network components. Configuring the MAQAN Controller is crucial for customizing the QKD operations to meet specific security requirements and operational parameters.

5) VMware Player 17.5 Setup: VMware Player offers a virtualization platform that enables the execution of multiple operating systems on a single physical machine. Specifically, for Quantum Key Distribution (QKD) applications, utilizing VMware Player permits the simulation and evaluation of QKD protocols across different settings without necessitating extra hardware resources. For this task, the virtual machines are configured with 4GB of memory, 3 processors, and 32GB of hard disk space, providing a balanced environment for efficiently handling the computational and storage demands of QKD simulations.

6) Simulation of NS3 and NetSquid: NS3 is a discrete-event network simulator, and NetSquid is a network simulator for quantum information. Together, they enable the simulation of both classical and quantum networks. This is vital for designing, analyzing, and optimizing the QKD process, allowing researchers and developers to model quantum key distribution scenarios, evaluate performance, and identify potential issues in a controlled environment.

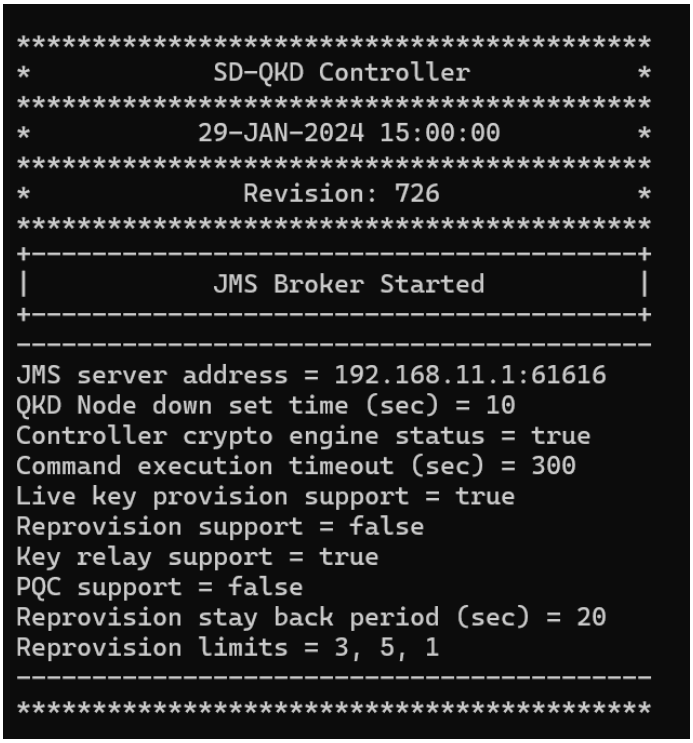

Fig. 4. MAQAN Controller simulation

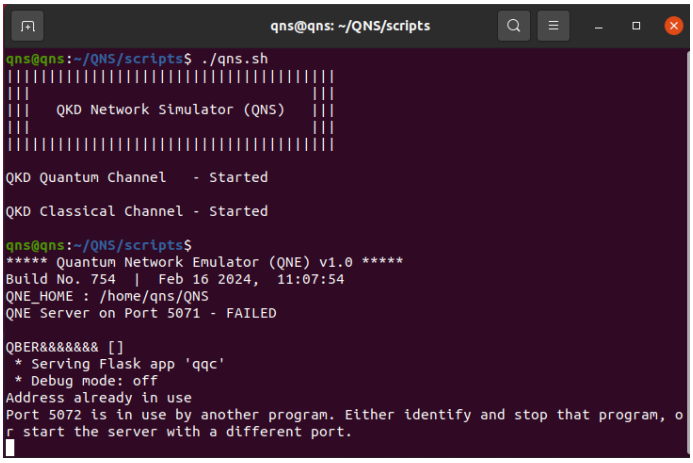

Fig. 5. QKD Network Simulator

7) Darpan Application Installation: An essential software element for the QKD process, this application aids in monitoring, analytics, and user interface, significantly improving the QKD system's usability and performance.

C. Proposed Solution

In this experiment, the focus is on the key reconciliation phase during the post-processing stage of Quantum Key Dis-

tribution (QKD). The proposed solution begins with the sender exchanging raw bits. During the sifting stage, the receiver provides a subset of his sifted key along with a timestamp to the sender. The sender then calculates the Quantum Bit Error Rate (QBER) as part of the key reconciliation phase. This QBER value is forwarded to the controller, which serves as the central intelligence hub. By integrating the Software-Defined Networking (SDN) concept, the controller determines the appropriate error correction mechanism based on predefined QBER criteria. This decision is communicated simultaneously to both the sender and the receiver, allowing the receiver to initiate the error correction process. The sender then transmits a codeword along with the HMAC of the codeword to the receiver. If the QBER value exceeds the threshold, both the sender and receiver discard the entire data set and restart the QKD process as instructed by the controller. For this experiment, two error correction mechanisms are implemented: 2D-Parity Check and the Cascade protocol.

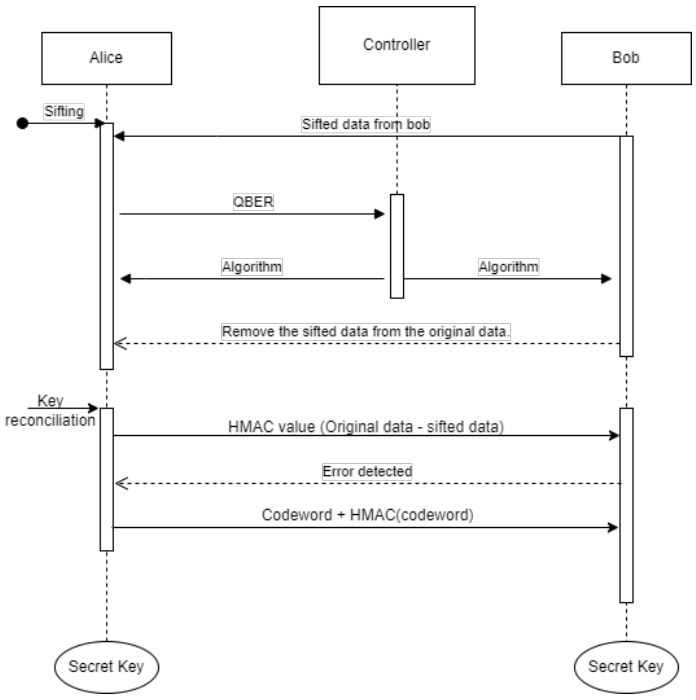

Fig. 6. Proposed solution's sequence diagram

This approach ensures a robust and adaptive QKD process, leveraging SDN to dynamically select and apply the most effective error correction methods, thereby enhancing the overall security and reliability of quantum communications. In the initial stage of the QKD process, a quantum bit is generated on Alice's side (assuming Alice is the sender). Bob (the receiver) then uses a random basis to generate his data based on what he receives. During the sifting phase, Bob sends a small portion of his data to Alice to check the Quantum Bit Error Rate (QBER). Alice calculates the QBER because she has the original data. The calculated QBER value is sent to the MAQAN controller to determine the appropriate error correction mechanism. The controller then sends the chosen algorithm back to both Alice and Bob. After receiving the error correction algorithm, both Alice and Bob discard the sifted data from their original datasets, ensuring the security and integrity of the key reconciliation process.

Bob then performs a Hash-based Message Authentication Code (HMAC) on his remaining data and sends it to Alice to confirm whether there are errors, since the QBER was only calculated on the sifted data. If the HMAC value of Bob's data does not match Alice's data, Alice runs the error correction algorithm on her side, creating a codeword. This codeword, along with its HMAC, is sent to Bob. Using the codeword, Bob corrects the errors in his data and performs an HMAC operation on the corrected data to verify it. This process ensures that a shared secret key is obtained by both parties.

For the experiment, we are implementing two error correction mechanisms: 2D Parity Check, and Cascade protocol.

1) 2D Parity Check: The 2D parity check is an effective method for error detection and correction in data transmission, utilizing both longitudinal and vertical parity checks to ensure data integrity. This technique organizes data into a two-dimensional grid, adding an extra parity bit to each row and column to ensure an even number of 1s. During transmission, both data and parity bits are sent to the receiver, who recalculates and compares the parity bits with the transmitted ones to detect errors. For error correction, a single bit error can be precisely located and corrected at the intersection of the row and column with incorrect parity. In cases of multiple errors, the 2D parity check provides valuable information for more advanced correction algorithms. This robust dual-level checking mechanism enhances data transmission reliability, particularly in Quantum Key Distribution (QKD), where data integrity and security are paramount.

2) Cascade Protocol: The cascade protocol serves as an error correction mechanism within Quantum Key Distribution (QKD) systems, operating during the key reconciliation phase. In this protocol, the sender iteratively transmits quantum states until the receiver confirms successful reception, ensuring the accuracy of the exchanged cryptographic key [3]. Upon receiving the quantum states, the receiver compares them with the expected states and identifies any discrepancies. If errors are detected, the receiver requests re-transmission of the erroneous quantum states from the sender. This iterative process continues until the receiver successfully receives all quantum states without errors. The primary objective of the cascade protocol is to achieve consensus between the sender and receiver regarding the correct quantum states, thereby upholding the integrity of the exchanged cryptographic key. Despite its effectiveness for certain error types, cascade protocols may require multiple iterations to efficiently correct errors, potentially leading to latency and affecting system performance. Additionally, cascade protocols may not be as suitable for dynamic environments where error patterns change frequently, as they depend on predetermined iteration strategies. In summary, while each error correction mechanism, including cascade protocols, has unique advantages and dis-

advantages, the selection of the mechanism depends on the specific requirements and constraints of the QKD system.

IV. Result and Analysis

The DARPAN application provides users with a graphical user interface (GUI) showcasing the entire QKD process. As illustrated in Figure 7, the communication occurs between two nodes, Alice and Bob. And the data transmission occour without any error (the QBER value is 0.0%).

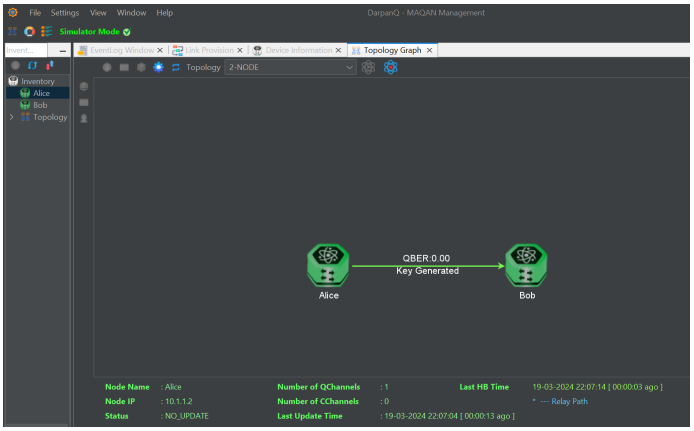

Fig. 7. DARPAN Application with 2-Node topology.

Upon initiating the simulation between these nodes, the subsequent phase involves the distribution of the secret key via the QKD process implementation. Initiating link provisioning triggers the execution of all QKD steps within the DARPAN application, which are displayed in Figure 8.

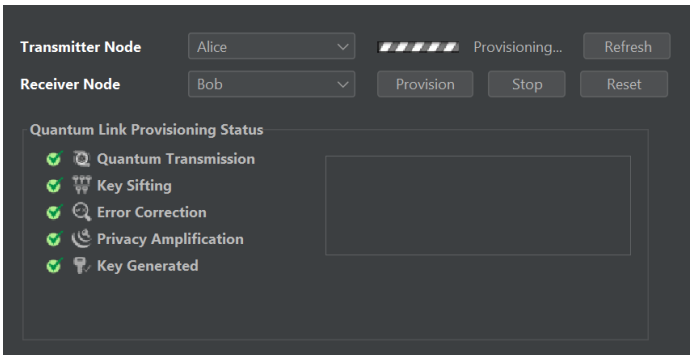

Fig. 8. Key provisioning in Darpan application.

Once link provisioning is complete and the shared secret key is generated, the DARPAN application features a configuration tab depicted in Figure 9. Here, users can access full information about the key, such as its size, length, initiator, key manager port, and other relevant details.

Within the DARPAN application, users can access a graphical analysis of the simulation's performance. Figure 10 demonstrates that the key generated between Alice and Bob exhibits a 0% Quantum Bit Error Rate (QBER), assuming there are no interruptions from intruders or external noise. Regarding the Key rate, it is observed to be 45 Kbps.

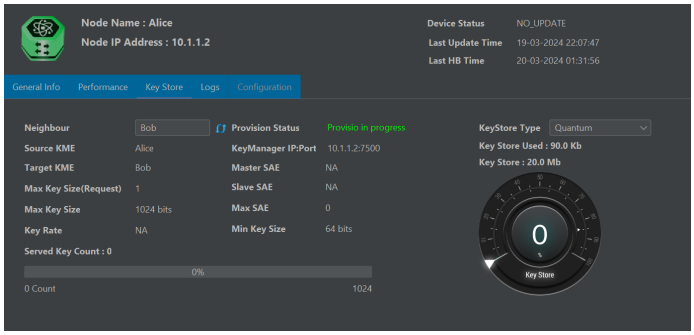

Fig. 9. Configuration setup at Alice side.

The program's execution results, shown in Figures 11 to 13, highlight its main functions. Figure 11 demonstrates that with a QBER value of 0, no error correction mechanism is triggered. If the QBER is between 1 and 5, the 2D parity check operation is executed (Figure 12), and if the QBER is between 5 and 15, the cascade error correction is applied. If the QBER exceeds 15, the entire key is discarded (Figure 13), and the QKD process restarts from the beginning. Integration of this

Fig. 10. Performance analysis based on QBER and Key Rate.

```
qns@qns:~/Documents$ ./MAIN11
Calculated QBER value: 0%
Matched: HMAC values are the same. Exiting.
qns@qns:~/Documents$
```

Fig. 11. QBER value equal to 0; no error correction.

```
qns@qns:~/Documents$ ./MAIN11
Calculated QBER value: 3%
Mismatched: HMAC values are different. Proceeding with error correction.
Selected Error Correction Mechanism: 2D-Parity
Running 2D Parity Error Correction Mechanism...
Bob's Key: 0000010010111100011010001101111111001101010010001011001100011011111110100110010011111100111110
Alice's Key: 0000010010111100011010001101111111001101010010001011001100011011111110100110010011111100111110
qns@qns:~/Documents$
```

Fig. 12. Output of 2D parity check.

```
qns@qns:~/Documents$ ./outt
NodeA QBER value: 32%
Mismatched: HMAC values are different. Proceeding with error correction.
QBER value beyond the treshold value. Discard
NodeB QBER value: 32%
Mismatched: HMAC values are different. Proceeding with error correction.
QBER value beyond the treshold value. Discard
```

Fig. 13. QBER value greater than 15; discard the entire key been exchanged.

function within the QKD environment results in an error-free secret key at both ends. Users of the Darpan application will not notice any changes and will only receive the final secret key.

V. Conclusion

Quantum Key Distribution (QKD) involves establishing a shared key between two parties, with the key reconciliation step being crucial for detecting and correcting errors caused by external factors or intruders during the transmission of raw bits. This post-processing phase ensures a secure key exchange by employing different error correction mechanisms, each with its own pros and cons.

In this work, we have implemented two error correction techniques, 2D Parity Check and Cascade Protocol, which are selected based on predefined Quantum Bit Error Rate (QBER) criteria. The controller within our setup, utilizing the Software-Defined Networking (SDN) concept, handles this decision-making process. Integrating SDN into QKD's key reconciliation phase allows for adaptive error correction based on live network conditions, improving resource efficiency and enhancing security through centralized control and automation. As the MAQAN controller, which interfaces with both classical and quantum channels in our QKD environment, continues to evolve, it increasingly assumes a master role while the sender and receiver function as slaves, following its directives. This implementation enhances system functionality and control. Future work includes integrating additional error correction mechanisms and evaluating the efficiency and performance of various key reconciliation methods within an SDN framework for ongoing improvement.

VI. Acknowledgement

This research received corporate support from the Center for Development of Advanced Computing (C-DAC) in Thiruvananthapuram.

References

[1] Aji, Aravind, Kurunandan Jain, and Prabhakar Krishnan. "A Survey of Quantum Key Distribution (QKD) network simulation platforms." 2021 2nd Global Conference for Advancement in Technology (GCAT). IEEE, 2021.

[2] Prakasan, Avani, Kurunandan Jain, and Prabhakar Krishnan. "Authenticated-encryption in the quantum key distribution classical channel using post-quantum cryptography." 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE, 2022.

[3] Mehic, Miralem, et al. "Error Reconciliation in Quantum Key Distribution Protocols." (2020): 222-236.

[4] Nurhadi, Ali Ibnun, and Nana Rachmana Syambas. "Quantum key distribution (QKD) protocols: A survey." 2018 4th International Conference on Wireless and Telematics (ICWT). IEEE, 2018.

[5] Bilash, Bohdan, et al. "Error-correction method based on LDPC for quantum key distribution systems." 2020 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, 2020.

[6] Gümüş, Kadir, et al. "A novel error correction protocol for continuous variable quantum key distribution." Scientific reports 11.1 (2021): 10465.

[7] Sajimon, P. C., Kurunandan Jain, and Prabhakar Krishnan. "Analysis of post-quantum cryptography for internet of things." 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE, 2022.

[8] Krishnan, Prabhakar, et al. "Sdn enabled qoe and security framework for multimedia applications in 5g networks." ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 17.2 (2021): 1-29.

[9] Raghunath, Karthik, and Prabhakar Krishnan. "Towards a secure SDN architecture." 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, 2018.

[10] Biswas, Chitra, Md Mokammel Haque, and Udayan Das Gupta. "A modified key sifting scheme with artificial neural network based key reconciliation analysis in quantum cryptography." IEEE Access 10 (2022): 72743-72757.

[11] Dhanush, Cheruku Sai, and Kurunandan Jain. "Comparison of Post-Quantum Cryptography Algorithms for Authentication in Quantum Key Distribution Classical Channel." 2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS). IEEE, 2023.

[12] W.K. Wootters, and W. H. Zurek, "A single quantum cannot be cloned," Nature 299, pp. 802-803, October 1982.

[13] Mummadi, Swathi, and Bhawana Rudra. "Practical demonstration of quantum key distribution protocol with error correction mechanism." International Journal of Theoretical Physics 62.4 (2023): 86.

[14] Wang, Hua, Yongli Zhao, and Avishek Nag. "Quantum-key-distribution (qkd) networks enabled by software-defined networks (sdn)." Applied Sciences 9.10 (2019): 2081.

[15] Wolf, Ramona. Quantum key distribution. Berlin/Heidelberg, Germany: Springer International Publishing, 2021.

[16] Reis, André. Quantum Key Distribution Post Processing-A Study on the Information Reconciliation Cascade Protocol. Diss. Universidade do Porto (Portugal), 2019.

[17] Al-Janabi, Sufyan T. Faraj, and Ruqayah Rabee Hashim. "Key Reconciliation Techniques in Quantum Key Distribution."

[18] Harmalkar, Manjiri, Kurunandan Jain, and Prabhakar Krishnan. "A Survey of Post Quantum Key Encapsulation Mechanism." 2024 5th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI). IEEE, 2024.

[19] Lorünser, Thomas, et al. "On the Security of Offloading Post-Processing for Quantum Key Distribution." Entropy 25.2 (2023): 226.

[20] Lin, Xiaxiang, et al. "An implementation of post-processing software in quantum key distribution." 2009 WRI World Congress on Computer Science and Information Engineering. Vol. 3. IEEE, 2009.

[21] Elkouss, David, Jesus Martinez-Mateo, and Vicente Martin. "Information reconciliation for quantum key distribution." arXiv preprint arXiv:1007.1616 (2010).

[22] Pirandola, Stefano, et al. "Advances in quantum cryptography." Advances in optics and photonics 12.4 (2020): 1012-1236.

[23] Broadbent, Anne, and Christian Schaffner. "Quantum cryptography beyond quantum key distribution." Designs, Codes and Cryptography 78 (2016): 351-382.

[24] Sharma, Purva, et al. "Quantum key distribution secured optical networks: A survey." IEEE Open Journal of the Communications Society 2 (2021): 2049-2083.

[25] Xu, Feihu, et al. "Secure quantum key distribution with realistic devices." Reviews of modern physics 92.2 (2020): 025002.