Electronics and
Electrical Engineering

https://www.aimspress.com/journal/ElectrEng

AIMS Electronics and Electrical Engineering, 8(4): 410–430.
DOI: 10.3934/electreng.2024020
Received: 17 May 2024
Revised: 25 August 2024
Accepted: 09 September 2024
Published: 13 September 2024

Research article

Software defined network implementation of multi-node adaptive novel quantum key distribution protocol

Hardeer Kaur[1,*] **and Jai Sukh Paul Singh**[2]

[1] School of Electronics and Electrical Engineering, Lovely Professional University, Jalandhar-Delhi GT Road, Phagwara, 144001, Punjab, India

[2] Department of Research Collaboration, Lovely Professional University, Jalandhar-Delhi GT Road, Phagwara, 144001, Punjab, India

* **Correspondence:** Email: hardeerkaurphd@gmail.com.

Abstract: Access to information can destroy nations and change the course of history altogether. Communication is very important, and in today's internet age, nothing moves without real-time information support. For securing communication, a commonly know technique is to use cryptography and public channels. Engineers have been working to create a better and more secure cryptographic system. Quantum key distribution stands at the top of this security system. Although QKD, based on principles of physics, provides a near-perfect security solution. It has a few drawbacks of its own, like low key generation rates and vulnerability to cyberattacks. Owning to these limitations, authors propose an adaptive quantum key distribution system based on software-defined networks. The authors propose to introduce redundancy in the key generation, thereby increasing the key generation rate and improving the resilience to cyberattacks. A performance comparison of novel quantum key distribution was done with BB84 and B92 quantum key distribution protocols.

Keywords: quantum mechanics; entanglement; quantum protocol; quantum simulator; quantum communication

1. Introduction

Quantum key distribution (QKD) proves to be the most secure way to generate keys and transfer data. Stephan Wiesner, in 1970, introduced the concept of quantum key distribution in his conjugate code for currency notes that cannot be forged [1]. Further, in the year 1984, Bennett and Brassard introduced the first key distribution based on quantum physics, commonly known as BB84 [2]. BB84 is the most researched QKD algorithm and is well known for its undoubted security. During the course of time, many QKD algorithms were developed to improve upon the limitations of their predecessors. Some of the prominent ones are (refer to Figure 1 for key highlights) E91 [3], B92 [4], EPR (BBM92) [32], CQPSK [31], SSP [5], COW [6], SARG04 [8], MDI [30], Twin-field QKD [29], Discrete-Modulation Continuous-Variable QKD [36], Asynchronous Measurement Device Independent Quantum Key Distribution [33].

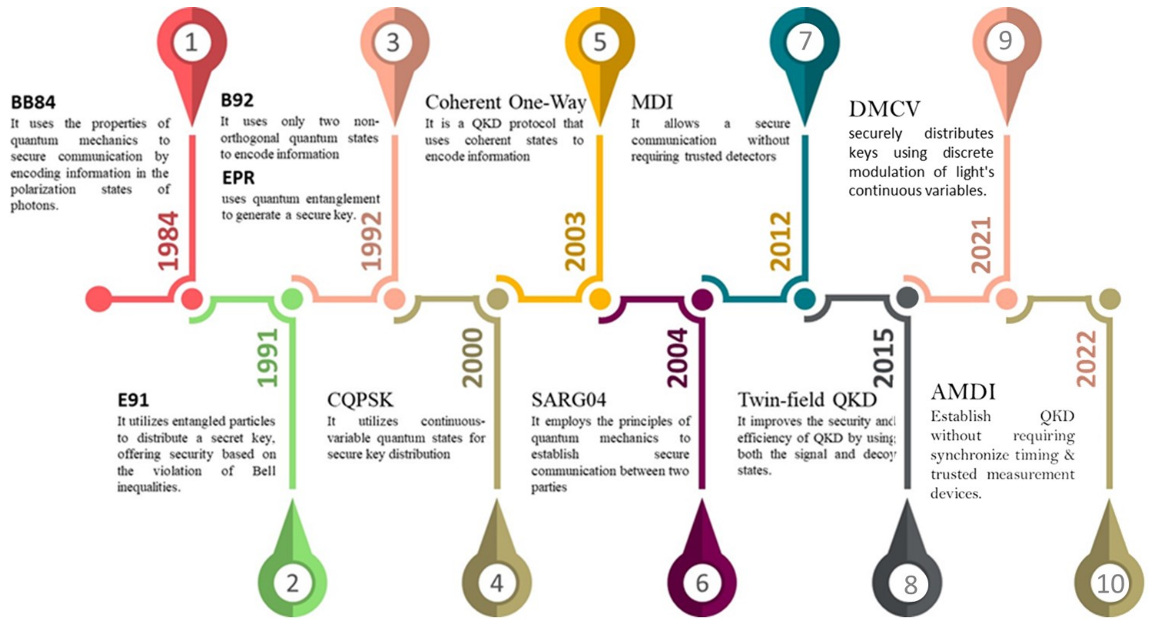
1
BB84
It uses the properties of quantum mechanics to secure communication by encoding information in the polarization states of photons.
1984
B92
It uses only two non-orthogonal quantum states to encode information
EPR
uses quantum entanglement to generate a secure key.
3
1992
5
Coherent One-Way
It is a QKD protocol that uses coherent states to encode information
2003
MDI
It allows a secure communication without requiring trusted detectors
7
2012
DMCV
securely distributes keys using discrete modulation of light's continuous variables.
9
2021
1991
E91
It utilizes entangled particles to distribute a secret key, offering security based on the violation of Bell inequalities.
2
2000
CQPSK
It utilizes continuous-variable quantum states for secure key distribution
4
2004
SARG04
It employs the principles of quantum mechanics to establish secure communication between two parties
6
2015
Twin-field QKD
It improves the security and efficiency of QKD by using both the signal and decoy states.
8
2022
AMDI
Establish QKD without requiring synchronize timing & trusted measurement devices.
10

Figure 1. Timeline for development of prominent QKD algorithms.

2. Software defined network

Software Defined Networking (SDN) is a revolutionary paradigm in computer networking that decouples the control plane from the data plane, enabling centralized management and orchestration of network resources [24]. SDN emerged in the early 2010s as a response to the limitations of traditional networks in terms of scalability, flexibility, and manageability [25]. By separating the control logic from the forwarding devices, SDN enables network administrators to define and implement network policies and services in a more agile and automated manner [26]. A typical pictorial representation of SDN implementation can be shown in the Figure 2.

SDN offers several advantages over traditional networks, including improved network programmability, increased network visibility, and enhanced security. Popular algorithms in SDN include OpenFlow and BGP-LS. From a practical perspective, SDN has been successfully deployed in various scenarios, including data center networks, wide area networks and Internet of Things (IoT) networks [25]. SDN is gaining importance as it allows better utilization of network resources. As discussed in multiple studies, SDN has effectively managed QKD-enabled networks. With SDN orchestrator, it becomes possible to integrate various QKD networks to develop complex Key Management Systems (KMS) wherein multiple networks use different configurations from various hardware vendors [37]. Although most of the QKD communication is tested for point-to-point communication. However, if QKD is established over an interconnected network, each node can work as a Classical Trusted Relay (CTR). This will help increase the distance of communication. When

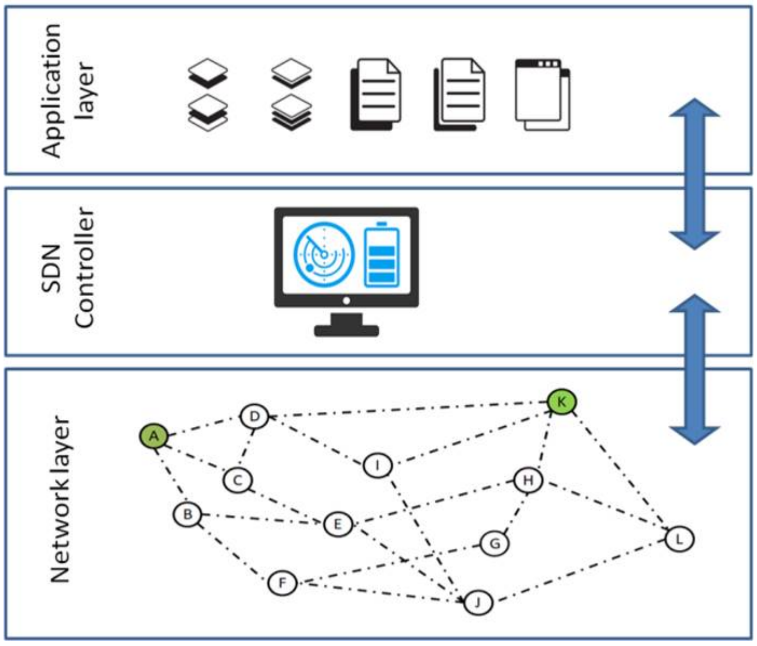

Figure 2. Graphical representation of SDN implementation.

using multiple CTRs, the error is produced at each node and gets accumulated causing QKD failure. An innovative solution to this is using SDQTRF [38]. Which allows SDN controller to assess the reason and overcome the CTR failure.

3. Quantum key distribution protocols

3.1. BB84 QKD protocol

The BB84 protocol is a discrete-variable (DV-QKD) "prepare and send" protocol, which has been extensively researched. It exploits the Heisenberg uncertainty principle to encode photons in four non-orthogonal quantum states. The working principle of BB84 (demonstrated in Figure 3) and its functionality can be classified into four stages:

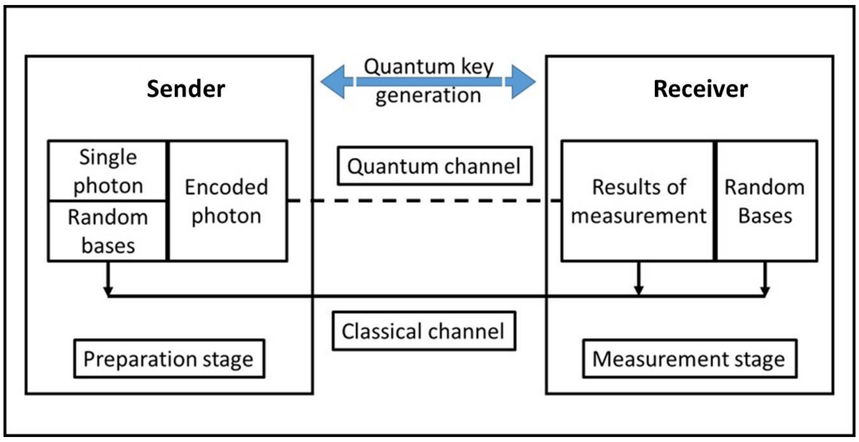

Figure 3. Block diagram for working of BB84.

1. **Prepare stage:** In the preparation stage, Alice generates an array of individual photons from a single photon source. Having a single photon source is an important pre-requisite, as it helps in proving the security of the QKD system. Utilizing a quantum random generator, Alice generates a random sequence of zeros and ones. This sequence is called random bases for encoding photons. These photons are encoded using generated random bases, i.e., horizontal/vertical or diagonal/anti-diagonal polarization (polarization shown in Figure 4). Afterwards, these encoded photons are sent to Bob using a quantum channel.

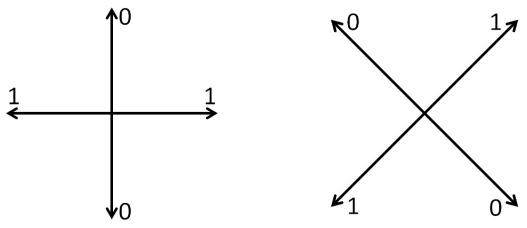

Figure 4. Non orthogonal bases used for BB84.

2. **Measurement stage:** In the measurement stage, Bob generates random bases to perform measurements on the received photons. Without being aware of the bases used by Alice for encoding, Bob uses his random bases for measurement. He only gets results for those photons that are measured with the same bases as Alice. It's worth mentioning that any set of non-orthogonal bases can be used to perform these measurements.
3. **Key generation stage:** Based on the results from measurement, Bob publicly announces the bases for which he had results, also indicating the positions where the results were obtained. Alice and Bob discard the bits for which Bob doesn't have results, keeping the remaining bits as the shared key.
4. **Verification stage:** A small portion of the generated key is compared to check for errors. The error observed can also be used to check for the presence of Eve. Typically, Eve tries to intersect the encoded photons sent from Alice, thereby disturbing the quantum state of the photons. This invariably produces a large amount of error in the generated key.

In quantum logic circuits, quantum bits are denoted as Qbits. A classical bit can attain only one out of two values (i.e., 0 or 1), but in comparison, Qbit can have all possible values between 0 and 1 (i.e., 0 or 1 or between 0 and 1). Because of this inherent property, the exact prediction of the state of Qbit is impossible. Hence, everything is calculated in the probability of attaining a state. Because of this, the results or measurements are never exact, thereby we will have some small errors even if everything is done perfectly. Essentially, it is difficult to interpret this cumulative error. Hence, the term QBER (Quantum Bit Error Rate) is used to denote the cumulative sum of all errors [9, 16, 17]. According to Bechmann-Pasquinucci and Bocquet, the threshold or reasonable limit of 11% and 15% is calculated for BB84 and BB84 with memory-less attack [15], respectively. The presence of Eve can also be determined by the magnitude of QBER in the produced key. The only way Eve can gain access to the data is by making a measurement of the encoded photons that are sent by Alice. But any measurement done on the photon invariably disturbs its quantum state. This change will generate a large amount of error in the system and can be detected easily.

3.2. B92 QKD protocol

Charles H. Bennett, in 1992 [4], proposed the B92 QKD protocol. This was quite similar to BB84 in operation and implementation, with only a major difference in the selection of random bases used for encoding and measurement of photons. BB84 requires four non-orthogonal bases (commonly horizontal, vertical, diagonal 45°, and diagonal 135°). In contrast, B92 uses only two bases (as shown in Figure 5). The methodology can be elaborated in the following steps:

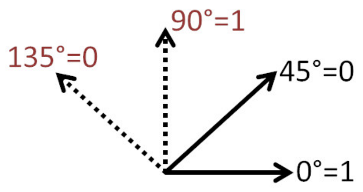

Figure 5. Non orthogonal bases used for B92.

1. **Prepare stage:** In the preparation stage, Alice generates an array of individual photons from a single photon source. These photons are encoded using random bases, i.e., horizontal/vertical or diagonal/anti-diagonal polarization (as shown in Figure 5). Afterward, these encoded photons are sent to Bob using a quantum channel.
2. **Measurement stage:** In the measurement stage, Bob also generates a set of random bases to perform measurements on the received photons. Like BB84, Bob only gets results for those photons that are measured with the same bases as Alice.
3. **Key generation stage:** Based on the results from measurement, Bob publicly announces the bases for which he had results, also indicating the positions where the results were obtained. Alice and Bob discard the bits for which Bob doesn't have results, keeping the remaining bits as the shared key.
4. **Verification stage:** A small portion of the generated key is compared to check for errors. The error observed can also be used to check for the presence of Eve. Typically, Eve tries to intersect the encoded photons sent from Alice, thereby disturbing the quantum state of the photons. This invariably produces a large amount of error in the generated key.

By using these steps, the B92 protocol enables secure key distribution between Alice and Bob, providing resistance against eavesdropping.

3.3. Limitations of BB84 and B92

Both QKD protocols (BB84 and B92) are based on similar principles; hence, they have some common limitations, as pointed out by [11, 12, 18, 19]. Although both protocols are infallibly secure. The threats in terms of security are caused by hardware limitations. Significant limitations can be listed as follows:

1. **Vulnerable to attacks:** QKD is an established secure method of communication that uses quantum physics to encrypt and decrypt data. On the contrary, QKD is not immune to cyber-attacks, and three standard attacks are Denial of Service (DOS), Photon Number Splitting (PNS) attacks, and Man-in-the-Middle (MITM) attacks. These attacks are detailed below:

- **Man-in-the-Middle (MITM):** In MITM [20] attack, the attacker impersonates as both the sender and the receiver. The attacker intercepts the communication between both and establishes two keys, one with the sender and the other with the receiver. Once both keys are established, complete access to the encrypted data can be gained without the knowledge of both parties.
- **Photon-number-split (PNS):** As discussed in chapter above, a single photon source is required to generate the initial photons that will be encoded. As a single photon source is not available, for all practical purposes, an attenuated laser source is used, which ideally should produce one or less than one photon per pulse. However, in some conditions, more than one photon is produced in each pulse. The attacker is equipped with a photon counter. Attacker [7] intercepts the message with more than one photon. It keeps one set of photons with itself and lets the remaining photons get to the receiver. These photons can be used later to replicate the generated key. If the attacker gets hold of enough photon pulses, it can generate a complete key.
- **Denial-of-Service (DOS):** The main objective of a DOS attack is not to gain access to the information but to stop communication. In most cases, the attacker tends to add additional information to the communication to overburden the network. DOS can be damaging, rendering the QKD system useless [21]. This might imply physical assaults on the communication infrastructure or employing other methods to overwhelm and interrupt the transmission of quantum signals.

2. **Length of the generated key :** Both BB84 and B92 are acknowledged to have a small length of generated keys. As the efficiency of BB84 and B92 are about 50% and 25% [23], respectively. For every 4 bits used to initiate QKD, the BB84 and B92 generates only 2 and 1 bits, respectively. Therefore, to obtain a large key, it is mandatory to start with an extremely large number of single-photon pulses, which is not practical. Hence, restricting the key length.
3. **Key generation rate :** For superior security, it is recommended [10] that a one-time keypad shall be used to encrypt and decrypt the data. If the same key is reused, then it increases the chances of the eavesdropper finding a pattern in the encrypted data and lead to the regeneration of the key. The limitation of using OTP is that the length of the key should be at least the length of data to be sent. For a true quantum network to match the current data transfer rates (gigabytes per second), one must generate a key at a similar pace. This is practically impossible with BB84 and B92 with 50% and 25% efficiency.

3.4. Modifications of existing protocols and latest protocols

Scientists have developed many algorithms to improve or reduce the above-mentioned limitations. A few worth mentioning are:

3.4.1. BB84 with decoy states

BB84 with decoy states enhances the security of the original BB84 quantum key distribution protocol. Despite the inherent security of the BB84 protocol, practical implementations are vulnerable to specific types of attacks, particularly the PNS attack. In an ideal scenario, Alice would send single photons to Bob, but real-world sources often emit weak coherent pulses (WCPs), which can contain multiple photons. This opens the door for an eavesdropper to exploit the multi-photon pulses by splitting off one or more photons while allowing the rest to reach Bob undisturbed. The decoy-state method was introduced to counter such vulnerabilities [22]. Working of BB84 with decoy states (Figure 6) can be summarized in the following steps:

1. **Generation of signal and decoy states:** Alice generates quantum states that include both signal states and decoy states, where the signal states are used for key generation, and decoy states are for security checks. The decoy states are identical in form to the signal states but have different photon intensities (e.g., weaker or stronger signals).
2. **Transmission:** Alice sends the prepared quantum states to Bob through a quantum channel, randomly choosing between signal and decoy states. Neither Alice nor Bob knows in advance whether any particular pulse is a signal or a decoy-state during transmission.
3. **Measurement by Bob:** Bob measures the incoming photons using a randomly chosen basis, just as in the original BB84 protocol. He recorded the results without initially knowing whether the photon he measured was from a signal state or a decoy-state.
4. **Classical communication and sifting:** After all photons are transmitted, Alice and Bob engage in classical communication using a strong light pulse. Additionally, Alice reveals which states were signal and which were decoy states. Bob compares this information with his measurements and sifts out the mismatched bases and irrelevant decoy states.
5. **Statistical analysis:** Alice and Bob perform a statistical analysis to compare the detection rates of signal and decoy states. This analysis allows them to estimate the level of potential eavesdropping and detect inconsistencies that would indicate an attack like the PNS attack.
6. **Key generation:** If the statistical analysis confirms no significant discrepancies, Alice and Bob proceed to generate a secure cryptographic key from the data of the signal states. The key is only generated if the results indicate the quantum channel is secure; otherwise, the session is aborted.

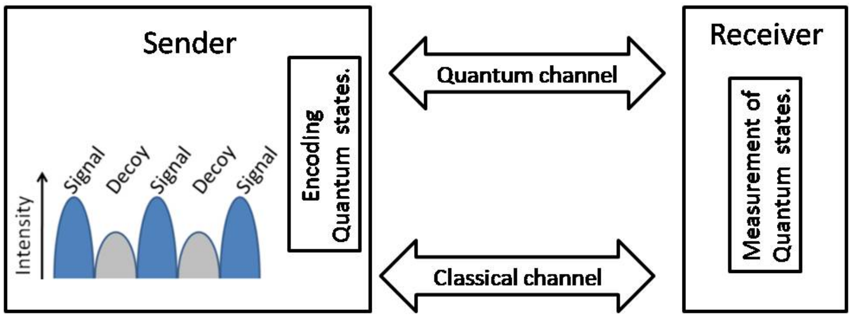

Figure 6. Working of BB84 with decoy states.

The primary advantage of the decoy state protocol is its ability to thwart PNS attacks. In a PNS attack, Eve exploits the multi-photon nature of weak coherent pulses by splitting off one photon from a multi-photon pulse while allowing the rest to continue to Bob. Since the decoy states have different intensities, Eve cannot selectively attack only the multi-photon pulses without being detected. The statistical analysis of the detection rates of decoy and signal states allows Alice and Bob to detect any discrepancies that would suggest a PNS attack. By monitoring the rate of photon detection in decoy states, Alice and Bob can estimate the channel's loss and detect any unusual behaviour indicative of an eavesdropper.

3.4.2. Asynchronous measurement-device-independent quantum key distribution protocol (AMDI-QKD)

The AMDI-QKD protocol offers enhanced security and flexibility compared to traditional QKD protocols like BB84 and B92. However, this comes at the cost of increased complexity and error correction challenges. As research advances, the benefits of AMDI-QKD may outweigh its limitations, making it a promising candidate for secure key distribution in various applications. The working AMDI-QKD protocol is similar to twin field quantum key distribution, but it operates without requiring a shared clock or synchronized measurements between Alice and Bob. The working principles of AMDI-QKD (refer to Figure 7) can expressed as:

1. **Preparation:** Alice and Bob each prepare a set of quantum states independently. These states can be in the form of photons encoded with Qbits. These encoded Qbits are sent to Charlie using multiple time bins. Each bin must contain only one single photon pair.
2. **Transmission:** Alice and Bob send their quantum states to Charlie, a central, untrusted measurement device. The key feature is that their transmissions do not need to be synchronized or simultaneous.
3. **Measurement:** Charlie performs a Bell state measurement on the received quantum states and announces the results to Alice and Bob. Importantly, Charlie's measurement device does not need to be trusted, as the protocol's security does not rely on it.
4. **Sifting:** Alice and Bob use the information from Charlie's announcements to discard non-useful measurement results and select sets of potentially correlated data.
5. **Error correction and privacy amplification:** Alice and Bob perform classical post-processing steps, including error correction to reconcile their data and privacy amplification to distil a secure key from the correlated data. These steps ensure the security of the final key, even if an eavesdropper has partial information about their quantum states.

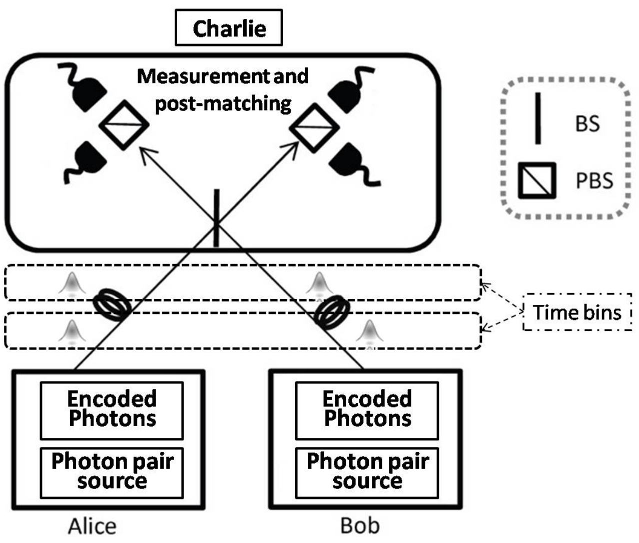

Figure 7. Working of asynchronous measurement device independent quantum key distribution (AMDI-QKD).

AMDI-QKD revolutionized secure communication by decoupling the measurement device from quantum key generation. This innovative protocol offers enhanced security, flexibility, and simplicity, making it a game-changer for secure key exchange in various applications, including finance, government, and high-stakes data transfer.

Table 1. Comparative review of BB84, B92, BB84 with decoy states and AMDI-QKD protocols: comparing the maximum key generation rate, performance metrics, and countermeasures.

Protocol	**Key Generation Rate**	**Distance**	**Pre-Authentication Required**	**Vulnerable to PNS**	**Vulnerable to MITM**	**PNS Countermeasures**	**MITM Countermeasures**
BB84 [2]	1.26 Mbps	20 km	No	Difficult to detect	Detectable	Terminate the current key exchange	Terminate the current key exchange
B92 [4]	12.7 kbps	10 km	No	Detectable	Easily detectable	Terminate the current key exchange	Terminate the current key exchange
BB84 with Decoy States [34]	144.8 Mbps	20 km	No	Difficult to detect	Cannot be detected	Adjust decoy state intensity	Terminate the current key exchange
AMDI-QKD [33]	24.6 Mbps	10 km	Yes	Easily detectable	Easily detectable	Error correction below threshold limit. Above the threshold limit, the key generation is restarted.	Authentication eliminates most of the attacks

BB84, B92 and BB84 with decoy states don't use any pre-authentication, which makes them vulnerable to MITM attack; in contrast, AMDI-QKD uses authentication, making it resilient to MITM attack. This pre-authentication is done using classical communication. This can be considered as a security risk, if the eavesdropper has access to this pre-authentication, then the entire QKD can be at risk. Similarly, in the case of a PNS attack, BB84 and B92 have nearly no immunity. Communication remains secure as a PNS attack can be detected, but the only countermeasures are to restart the communication and check for attacks again. BB84 with decoy states can adjust the intensity of decoy states to check and avoid PNS attacks. AMDI-QKD uses pre-authentication and error correction algorithms to detect and circumvent PNS attacks. It monitors the errors in the generated key and uses error correction if the error is below the threshold limits. If the error is above the threshold limits, the key generation process is restarted. A summary of the comparison is presented in the Table 1.

4. Development of novel adaptive QKD algorithm

As discussed in the above sections, classical QKD algorithms, including BB84 and B92, have major limitations of key generation rate, distance of key generation, and vulnerability to network attacks. BB84 with decoy states tries to solve these limitations. It provides a strong defense against PNS attacks and detects the presence of any eavesdropper. As it stops the communication once the eavesdropper is identified, the communications remain secure, but the data is not transferred, which is undesirable. A novel adaptive QKD algorithm is proposed, which can be deployed on SDN to address common attacks and overcome the limitations (as shown in Figure 8). The algorithm is proposed to be implemented on a SDN because the network can be programmed to self-optimize the network settings. It is presumed that the implementation is done on an existing quantum network in which multiple nodes are interconnected in a web formation, providing multiple paths to reach from one terminal to another.

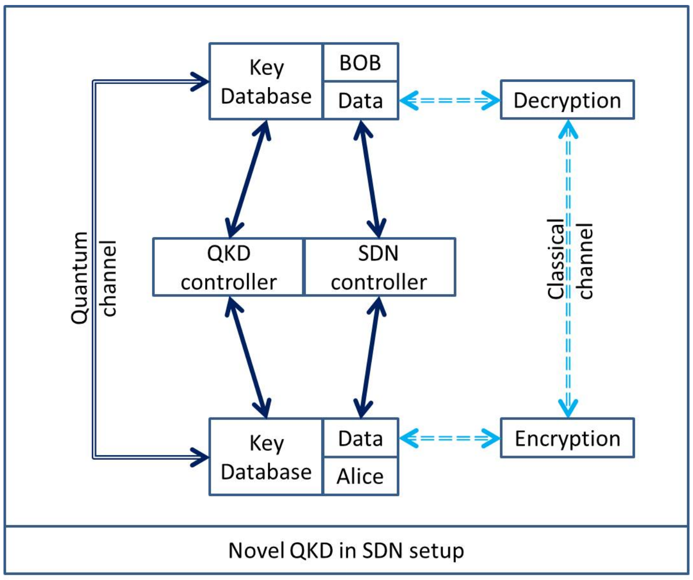

Figure 8. Proposed schematic of QKD over software-defined network.

4.1. Working of novel adaptive QKD algorithm

Working of the novel adaptive QKD can be explained as follows:

1. **Node recognition and channel identification:** Taking an example that a node **"A"** wants to establish QKD and transfer data to node **"K"**. The first step is to discover the appropriate paths. This can be achieved by:
   - Select two nodes among which the key needs to be generated (Node A and Node K).
   - The most suitable four channels are selected (as shown in Figure 9) based on stack value computation (distance, errors, noise, presence of Eve). Dijkstra's algorithm can be used to find the most suitable paths between Node A & Node K [13, 14]. Although there are numerous paths possible between A and K, one can choose any number of paths based on the availability of hardware. For ease of implementation and faster simulation, we have opted to select only the first four paths.
   - Utilize the following four channels for establishing parallel QKD connections:
     - A-D-K
     - A-D-I-K
     - A-B-E-H-K
     - A-C-E-H-K
2. **Establishment of quantum key distribution:**
   - Node "A" prepares a set of random photon pulses and encodes them using another set of randomly generated non-orthogonal bases (e.g., rectilinear and diagonal). Any two sets of non-orthogonal states can be used for encoding. This process is done separately for all the four paths.
   - Node "A" sends the encoded photons through the respective channels in parallel.
   - Node "K" separately measures the photons received from four paths and records the outcomes. The measurements are done using random (non-orthogonal) bases. Any two sets

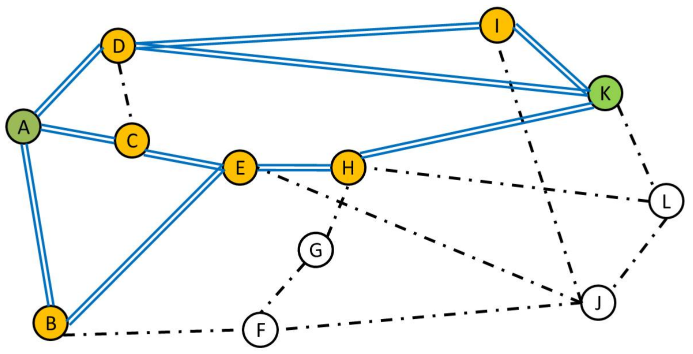

Figure 9. Interconnected quantum network with four paths (marked in blue).

of non-orthogonal bases can be used, the only constraint is that both the nodes shall use the same set for encoding and measurement.

- Node "A" and "K" repeat the process several times to generate a large number of outcomes.
- Standard QKD classical post-processing is used to correct errors and extract a shared secret key. These keys generated by separate paths are saved as redundant keys in separate databases.

3. **Error estimation and key assortment:**

- Error percentage for each redundant key is calculated; in our example, four error percentages are calculated (E1, E2, E3, E4). The error generated in each path is calculated as [35]:

$$P(\text{error}) \leq \frac{1}{2}\left(1 - (1 - 2p_d)\, e^{-2\mu\eta}\right) \tag{4.1}$$

Where: P(error) is the probability of error

p_d is the dark count probability

μ is the mean photon number

η is the detection efficiency

for $p_d \ll 1$ (low dark count probability), the equation can be reduced to:

$$\text{QBER} \approx \frac{4e^{-2\mu\eta}(1 - e^{-2\mu\eta})}{1 - 2e^{-2\mu\eta}} \tag{4.2}$$

Going forward, we consider that we have a perfect single photon source as prescribed in the BB84 algorithm. If $\mu = 1$, then the equation 4.2 will reduce to $QBER \approx 0$. Hence, it can be said that there is no error or perfect correlation under perfect network conditions with a single photon source. However, in a physical testing scenario, the errors are introduced by multiple factors such as photon loss, noise, interference, detector efficiency, network fluctuation, etc. In a more global approach, the following equation can be used:

$$E_i = \frac{N_{error}}{N_{sifting}} \tag{4.3}$$

Where: N_{error} is the number of error bits.

$N_{sifting}$ is the number of bites used for sifting.

- The average error percentage is calculated using all four redundant keys (using equation 4.4).

$$E_{Avg} = \frac{1}{n}\sum_{i=1}^{n} E_{\mathrm{i}} \tag{4.4}$$

Where: E_i is the error in the i^{th} path.

- If the average error percentage in the concatenated key is above 11%, then the key with the highest individual error is discarded. The path from which this key is extracted is also ignored for future key generation.

This complete process of key generation is repeated until the required length of the key is generated with an error percentage within the limits.

4. **Key storage and data transfer:**
   - The concatenated key is stored in individual databases at both nodes (Node "A" and "K"), as shown in Figure 8.
   - Keys from these databases can be used for encrypting and decrypting data while sending it through the classical channel.
   - Parameters like error rate and key generation rate are continuously monitored for performance assessment.
   - With this algorithm, the major focus is on evaluating errors, which inevitably eliminates any hidden eavesdroppers in the system. Any measurement performed by Eve will result in a spike error.
   - Utilizing the adaptive SDN framework helps reprogram the network connections to avoid compromised channels and nodes. It also aids in continuous key generation, even in the presence of Eve in a few of the connections.

With SDN, the network can be programmed to respond to environmental changes, such as Eve's presence, and the QKD protocol can be adjusted accordingly. One scenario could be when Eve starts moving from one section to another. In this case, our adaptive QKD can also adaptively switch the communication between the available channels to avoid data loss to Eve. This enables a more dynamic and adaptive approach to securing the quantum key distribution process.

4.2. Utilizing SDN to make novel QKD algorithm resilient

Novel-QKD algorithm is designed to be agile and should be able to transform itself according to the network environment. Implementing the QKD over SDN allows the QKD to be self-governing and self-modulating. We can understand this by using a simplified flowchart (refer to Figure 10). Both the sender and receiver have their own separate databases to store keys. These databases are essentially identical to each other as the key used is asymmetric. At the beginning of the communication, the database is checked for the sufficiency of key availability. The key length shall equal the size of the data to be sent. If found sufficient, then communication is continued. If the keys are less than the required quantity, the QKD process begins.

For the QKD process, first, depending on available hardware, the number of most suitable paths connecting sender to receiver is estimated. Paths are ranked based on the availability and presence of eavesdroppers, as well as errors and losses. The paths with higher ranks are chosen first. Once the paths are identified, multiple QKD(s) are formed between sender and receiver. Errors for all the paths are checked, and if the error is found to be above the predefined threshold limit, QKD from that particular path is terminated. QKD from other paths continues until the required key length is achieved. Further, new paths are calculated while neglecting the terminated path. Once found, QKD is also stored from that path. This loop of error checking, terminating paths, and recalculating the path will continue until a sufficient number of keys are accumulated in the database.

In case of an attack, redundancy due to multiple paths and agility to switch between the paths helps the protocol to circumvent attacks. Any attack invariably will produce an error in the key generation; this error can be detected easily, and based on the error, the protocol will adjust itself by switching to different paths. At any given time, the algorithm will use the most suitable paths from the network. This will ensure the key generation even if some network section is compromised.

SDN controller ascertains flexibility in finding multiple paths, allowing switching between paths, checking for errors, and correcting the QKD process.

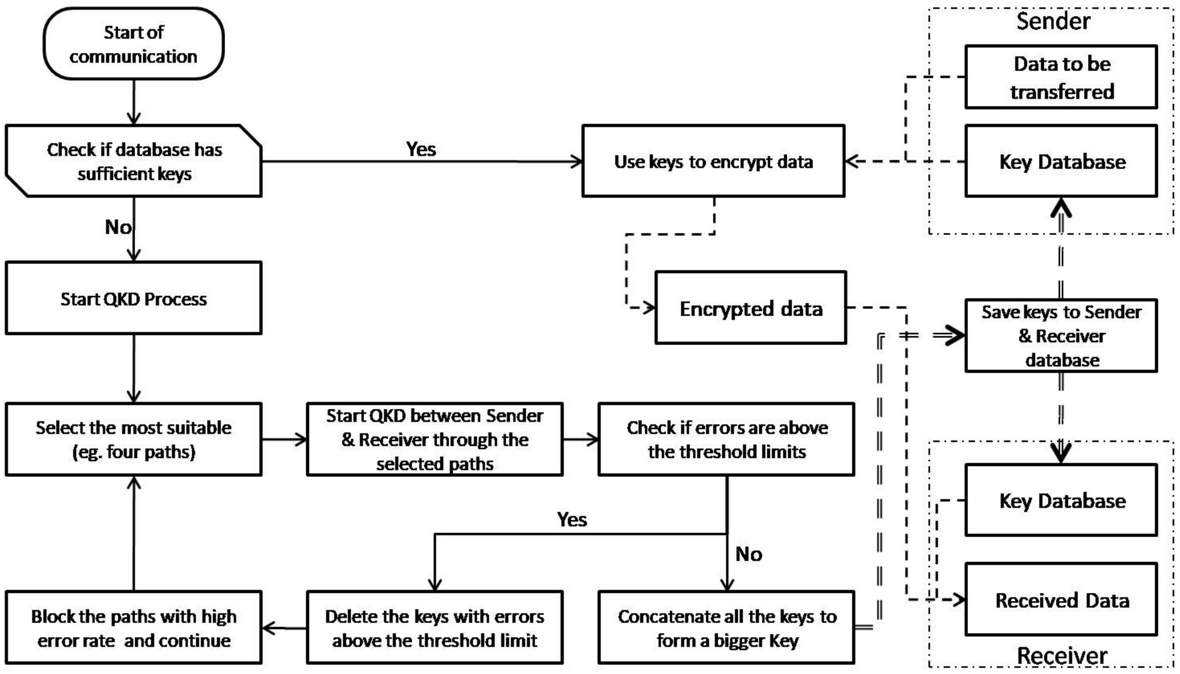
Start of communication
Check if database has sufficient keys
Yes
No
Use keys to encrypt data
Start QKD Process
Encrypted data
Select the most suitable (eg. four paths)
Start QKD between Sender & Receiver through the selected paths
Check if errors are above the threshold limits
Yes
No
Block the paths with high error rate and continue
Delete the keys with errors above the threshold limit
Concatenate all the keys to form a bigger Key
Sender
Data to be transferred
Key Database
Save keys to Sender & Receiver database
Key Database
Received Data
Receiver

Figure 10. Flow chart showing the implementation of novel QKD algorithm.

5. Relative evaluation of BB84, B92 and novel QKD protocols

A relative analysis of BB84, B92 and novel QKD protocols is done with special attention to the following parameters:

- Key generation efficiency.
- Generated key length.
- Rate of key generation.
- Error percentage in generated key.
- Resilience to cyberattacks.

5.1. Simulation setup for comparative analysis

A simulation environment was developed using Python3 programming (refer to Figure 11). For simulation, a Windows 10 system with an i7-10700F CPU and 32 GB RAM was used.

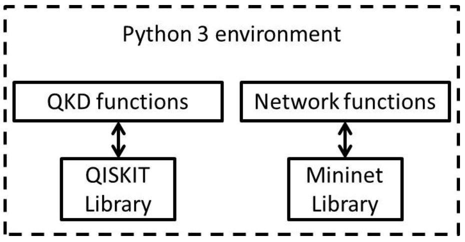

Figure 11. Simulation setup for Python code.

QISKIT library functions were used to perform all QKD related functions. QISKIT is an open-source Python-based library maintained by QBM research [27]. It aims to help researchers develop and test complex quantum circuits and various quantum functions. All the network-related functions required for the realization and testing of the QKD protocols were assembled using the open-source Python library "Mininet". Mininet acts as an emulator for creating standard and software-defined networks. It delivers an appropriate and dependable test bed for network testing at a meagre cost [28].

5.2. Key generation efficiency

Key generation efficiency is estimated by calculating the ratio between the generated key length and the bit length of the initial set of photons used to initiate the QKD ($\mathrm{N}_{\mathrm{Initial}}$) used.

For BB84 and B92 algorithms, the efficiency (η) is estimated around 50% and 25%, respectively; the channel doesn't have any noise or interference.

The length of the generated key ($\mathrm{Key}_{\mathrm{Length}}$) can be estimated as

$$Key_{\mathrm{Length}} = \eta N_{\mathrm{Initial}} \tag{5.1}$$

Theoretically, novel QKD essentially consists of multiple QKDs working in parallel. Hence, the length of the generated key also follows the same analogy. The only difference is that the total length is dependent upon the number of paths or channels used ($\mathbf{N}_{\mathbf{paths}}$) for establishing QKD. The standard equation (Equation 5.1) can be modified into the following equation.

The length of the generated key ($\mathrm{Key}_{\mathrm{Length}}$) can be derived as

$$Key_{\mathrm{Length}} = \eta N_{\mathrm{Initial}} N_{\mathrm{paths}} \tag{5.2}$$

It is worth noting that, as multiple QKD(s) are running in parallel, the number of initial bits (N_{Initial}) will also be the sum of initial bits used in each individual QKD path. Hence, effective efficiency will largely depend on the algorithm used in the individual paths. In our case, we have used BB84. Therefore, the efficiency will be similar to that of BB84, which is about 50%

5.3. Generated key length

The average key length was determined across all protocols, and a graphical representation was generated, plotting the initial bit length utilized during the simulation. Analysis of the graph (Figure 12) reveals that the novel QKD algorithm exhibits the longest key length, with BB84 and B92 algorithms following. As the length of the generated key is based on probability distribution (due to the inherent quantum uncertainty), it is difficult to arrive at a particular value. Hence, it is recommended that multiple simulations be performed and then an average value taken. To calculate the average key length, fifty simulations were conducted for each specific set of initial bits. Various sets of initial bit lengths (8, 16, 32, 64, 128, 256, 512, 1024, and 2048) were employed for this analysis.

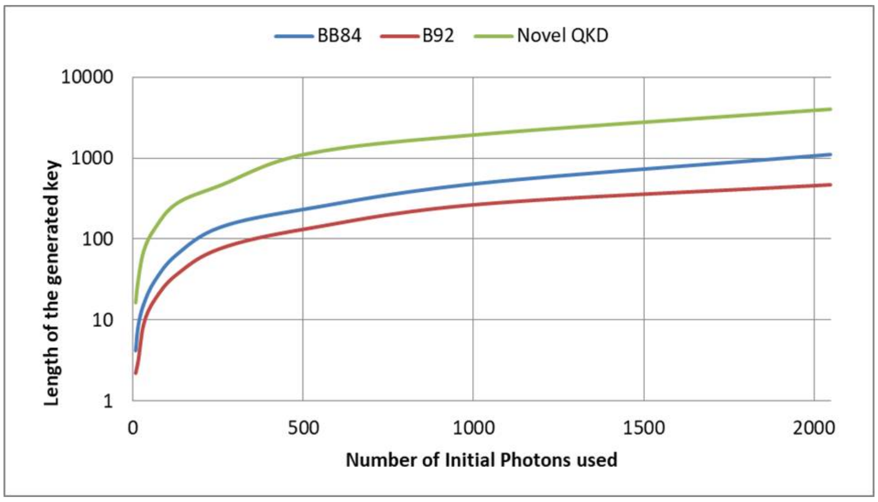

Figure 12. Generated key length versus the initial bit length used. Graph presents the results for the comparative study conducted between BB84, B92 and novel QKD algorithms. The graph is plotted between the number of initial photons used to start the key generation (on the horizontal axis) and the average length of the generated key (on the vertical axis, in Log scale).

It can be seen from the graph that the average length of the generated key in B92 is nearly half of what was achieved using BB84. Further, the key generated by the novel QKD was four times longer than BB84. Hence, it can be said that using multiple channels to conduct parallel QKD(s) in a network results in a much larger key. The length of the key can be estimated by using equation 5.2.

5.4. Rate of key generation

The key generation rate is a critical parameter in QKD protocols. This study conducted simulations for 120 seconds to determine the key generation rates of BB84, B92 and the novel QKD protocol (utilizing four channels). A comparative graph (refer to Figure 13) was plotted to illustrate the key generation rates against time for the various protocols.

It was observed that the key generation rates were stable with time. The key generation rates for B92 were the lowest, nearly half of the BB84. Further, the key generation rate for the novel QKD algorithm was multiple times higher in comparison to BB84 and B92. This was in accordance with previous results. The key generation rate is proportional to efficiency and key generation length.

However, in the novel QKD algorithm, some additional time is utilized to identify the suitable path: *First,* before starting the key distribution process. *Second,* once the eavesdropper is detected and it is required to close the channel, causing the algorithm to search for the suitable paths again. Both situations add an overhead to the total time required to complete the key generation. In our simulation,

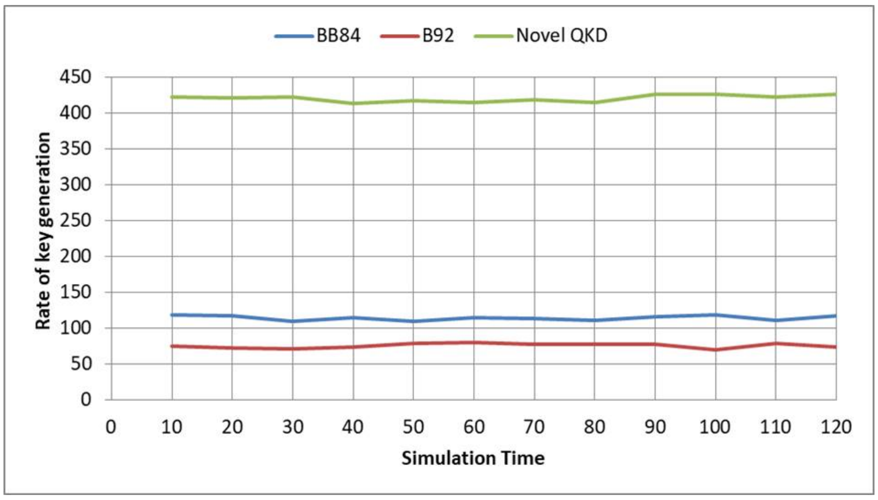

Figure 13. Key generation rate versus time. The graph shows the key generation rate measured over a period of time (120 s). A simulation environment was created, and the key generation rate was recorded for all three algorithms under similar conditions (1024 initial bits). The channels were assumed to have no channel noise and were free from any eavesdropper.

this overhead was negligible and not recorded. Having said that, for a complex network, it could be significant and would require further investigation in the future.

5.5. Error percentage in generated key

In QKD, the error rate is determined by comparing a small portion of the generated key. Ideally, the error rate for BB84 should be below 1% in the absence of noise in the channel. However, in the presence of noise, an acceptable error rate should be below 11%. If the error rate exceeds this threshold, the generated keys are discarded, and the QKD process is restarted.

For comparative analysis (refer to Figure 14), BB84, B92, novel QKD, and novel QKD (with an eavesdropper present in one of the channels) were simulated using four channels for 120 seconds. The average error rate was plotted against time to compare the performance of these protocols. Although standard practice recommends halting communication if the error rate exceeds an acceptable limit, for comparison, communication was not terminated in this study. For BB84, B92 and novel QKD (without eavesdropper), it was assumed that all the errors are because of the inherent properties of the protocol, and no eavesdropper is present. Even for novel QKD (with one eavesdropper), the total error was recorded, which includes the error introduced by an eavesdropper.

It was observed that B92 has a higher error rate in comparison to BB84 and novel QKD (no eve) algorithm. It can be attributed to the fact that B92 uses only two non-orthogonal states, making them less distinguishable and increasing the likelihood of measurement errors. In contrast, BB84 uses four states, which reduces the chance of errors by providing more options for the correct basis choice during measurement. Additionally, B92's simplicity leads to a higher sensitivity to noise, further increasing the error rate.

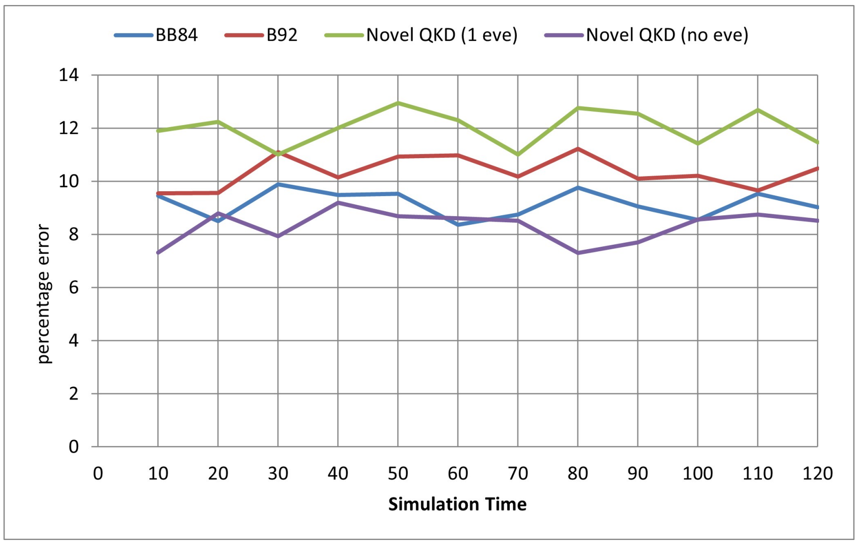

Figure 14. Error in generated key versus time. A simulation program was used to simulate all three algorithms, and their error rates were compared. Novel-QKD was simulated twice: *first*, without the eavesdropper and four channels, *and second,* with the eavesdropper in one of the four channels.

5.6. *Resilience to cyber-attacks*

Cyber attacks on networks are increasingly sophisticated and challenging to detect and mitigate. Introducing redundancy in QKD systems enhances their resilience against eavesdropping attacks. Implementing the algorithm over a software-defined network allows for seamless path changes during communication, making it difficult for attackers to exploit vulnerabilities such as MITM and DOS attacks. However, if the entire network is compromised by an eavesdropper capable of attacking all connections simultaneously, the situation becomes critical.

- DOS attacks, for instance, can overload the network, leading to a collapse. Common remedial measures include blocking the node that causes the disturbance or restarting the entire network. In the classical BB84 QKD, overloading can be easily detected, but since there is only one connection available, switching it off results in network downtime. In contrast, with multiple communication paths available, such attacks can be easily avoided by simply changing the communication path.
- In a MITM attack, the eavesdropper impersonates both the sender and the receiver, generating two keys simultaneously—one with the sender and the other with the receiver. Consequently, both the sender and the receiver remain unaware of the eavesdropper's presence. However, this attack cannot be avoided without pre-authentication. When using redundant keys, users can validate the generated keys against each other to detect the presence of an eavesdropper.
- Photon number split attacks can be thwarted by carefully generating photon pulses in such a way that only one photon is produced per pulse. If a single pulse contains more than one photon, the eavesdropper can intercept one photon to generate its own key, resulting in data theft. However, this can be prevented by precisely generating single-photon pulses.

6. Conclusions

The prevalent QKD algorithms, including BB84 and B92, suffer from various limitations. These include very low key generation rates, higher error rates, and sensitivity to channel attacks such as MITM and DOS attacks. To mitigate these limitations, researchers propose an innovative adaptive QKD algorithm aimed at overcoming these shortcomings and improving its practical applicability in secure communication.

Using computer simulations, it was proven that the novel algorithm's key generation rate substantially increased, accompanied by a substantial reduction in error rates. In addition, the algorithm's resilience against MITM and DOS attacks was proven to be better than that of its predecessors.

Additionally, the study revealed an interesting finding: The novel QKD algorithm's utilization of multiple channels for key generation enabled it to tolerate higher channel losses in some channels without compromising overall system security and efficiency. This discovery opens up new possibilities for QKD realization in various communication environments.

7. Limitations and further work

As proved in the results section, the novel QKD algorithm has a superior rate of key generation. However, in order to achieve this higher key generation, an interconnected network is required. In the novel QKD algorithm, the nodes are required to communicate through multiple channels simultaneously, which requires a very complex hardware setup. The novel QKD algorithm provides us with an opportunity for future improvements. The protocol can be improved to generate an exponentially large key by finding more efficient methods to combine the redundant keys. Further, innovative techniques can be utilized to reduce hardware complexity and to reduce cost.

Author contributions

Jai Sukh paul Singh: Methodology; Hardeer Kaur: Software, Writing – original draft, Validation. All authors have read and agreed to the published version of the manuscript.

Conflict of interest

The authors have no competing interests to declare that are relevant to the content of this article. Also all authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

References

1. Wiesner S (1983) Conjugate Coding. *ACM Sigact News* 15: 78–88. https://doi.org/10.1145/1008908.1008920
2. Bennett CH, Brassard G (2014) Quantum cryptography: Public key distribution and coin tossing. *Theoretical Computer Science* 560: 7–11. https://doi.org/10.1016/j.tcs.2014.05.025

3. Ekert AK (1991) Quantum cryptography based on Bell's theorem. *Phys Rev Lett* 67: 661–663. https://doi.org/10.1103/PhysRevLett.67.661

4. Bennett CH (1992) Quantum cryptography using any two non-orthogonal states. *Phys Rev Lett* 68: 3121–3124. https://doi.org/10.1103/PhysRevLett.68.3121

5. Bechmann-Pasquinucci H, Gisin N (1999) Incoherent and coherent eavesdropping in the six-state protocol of quantum cryptography. *Physical Review* 59: 4238–4248. https://doi.org/10.1103/PhysRevA.59.4238

6. Stucki D, Fasel S, Gisin N, Thoma Y, Zbinden H (2007) Coherent one-way quantum key distribution. *Photon Counting Applications, Quantum Optics, and Quantum Cryptography* 6583: 194–197. https://doi.org/10.1117/12.722952

7. Scarani V, Acin A, Ribordy G, Gisin N (2004) Quantum Cryptography Protocols Robust against Photon Number Splitting Attacks for Weak Laser Pulse Implementations. *Phys Rev Lett* 92: 057901. https://doi.org/10.1103/PhysRevLett.92.057901

8. Scarani V, Acin A, Ribordy G, Gisin N (2004) Quantum cryptography protocols robust against photon number splitting attacks. *Phys Rev Lett* 92: 197901. https://doi.org/10.1103/PhysRevLett.92.197901

9. Lütkenhaus N (1999) Estimates for practical quantum cryptography. *Phys Rev A* 59: 3301–3319. https://doi.org/10.1103/PhysRevA.59.3301

10. Bruß D, Lütkenhaus N (2000) Quantum Key Distribution: from Principles to Practicalities. *Applicable Algebra in Engineering, Communication and Computing* 10: 383–399. https://doi.org/10.1007/s002000050137

11. Bennett CH, Bessette F, Brassard G, Salvail L, Smolin J (1992) Experimental Quantum Cryptography. *J Cryptol* 5: 3–28. https://doi.org/10.1007/BF00191318

12. Aggarwal R, Sharma H, Gupta D (2011) Analysis of Various Attacks over BB84 Quantum Key Distribution Protocol. *International Journal of Computer Applications* 20: 28–31. https://doi.org/10.5120/2454-3313

13. Sniedovich M (2006) Dijkstra's algorithm revisited: the dynamic programming connexion. *Control and Cybernetics* 35: 599–620.

14. Dijkstra EW (1959) A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik* 1: 269–271. https://doi.org/10.1007/BF01386390

15. Bechmann-Pasquinucci H (2006) Eavesdropping without quantum memory. *Phys Rev A* 73: 044305. https://doi.org/10.1103/PhysRevA.73.044305

16. Lütkenhaus N (1996) Anosov flows with stable and unstable differentiable distributions. *Phys Rev A* 54: 97–111.

17. Slutsky BA, Rao R, Sun PC, Fainman Y (1998) Security of quantum cryptography against individual attacks. *Phys Rev A* 57: 2383–2398. https://doi.org/10.1103/PhysRevA.57.2383

18. Zhao B, Zha X, Chen Z, Shi R, Wang D, Peng T, et al. (2020) Performance Analysis of Quantum Key Distribution Technology for Power Business. *Applied Sciences* 10: 2906. https://doi.org/10.3390/app10082906

19. Bruss D, Erdélyi G, Meyer T, Riege T, Rothe J (2007) Quantum cryptography: A survey. *ACM Computing Surveys* 39: 6–es. https://doi.org/10.1145/1242471.1242474

20. Fei YY, Meng XD, Gao M, Wang H, Ma Z (2018) Quantum man-in-the-middle attack on the calibration process of quantum key distribution. *Scientific Reports* 8: 4283. https://doi.org/10.1038/s41598-018-22700-3

21. Mehic M, Rass S, Dervisevic E, Voznak M (2022) Tackling Denial of Service Attacks on Key Management in Software-Defined Quantum Key Distribution Networks. *IEEE Access* 10: 110512––110520. https://doi.org/10.1109/ACCESS.2022.3214511

22. H.-K. Lo, X. Ma, and K. Chen (2005) Decoy state quantum key distribution. *Phys Rev Lett* 94: 230504. https://doi.org/10.1103/PhysRevLett.94.230504

23. SujayKumar Reddy M, Chandra Mohan B (2023) Comprehensive Analysis of BB84, A Quantum Key Distribution Protocol. *quant-ph* eprint=2312.05609.

24. Kreutz D, Ramos FM, Verissimo PE, Rothenberg CE, Azodolmolky S, Uhlig S (2015) Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE* 103: 14–76. https://doi.org/10.1109/JPROC.2014.2371999

25. Jain R, Paul S (2013) Network virtualization and software defined networking for cloud computing: a survey. *IEEE Commun Mag* 51: 24–31. https://doi.org/10.1109/MCOM.2013.6658648

26. Casado M, Freedman MJ, Pettit J, Luo J, McKeown N, Shenker S (2007) ETHANE: Taking Control of the Enterprise. *Computer Communication Review - CCR* 37: 1–12. https://doi.org/10.1145/1282380.1282382

27. Barabasi S, Barrera J, Bhalani P, Dalvi P, Dimiecik R, Leider A, et al. (2020) Student User Experience with the IBM QISKit Quantum Computing Interface. *Lecture Notes in Networks and Systems*, 547–563. https://doi.org/10.1007/978-3-030-12385-7_41

28. Kaur K, Singh J, Ghumman NS (2014) Mininet as Software Defined Networking Testing Platform. *International conference on communication, computing & systems (ICCCS)*, 139–142.

29. Curty M, Azuma K, Lo HK (2019) Simple security proof of twin-field type quantum key distribution protocol. *npj Quantum Information* 5: 64. https://doi.org/10.1038/s41534-019-0175-6

30. Lo HK, Curty M, Qi B (2012) Measurement-Device-Independent Quantum Key Distribution. *Phys Rev Lett* 108: 130503. https://doi.org/10.1103/PhysRevLett.108.130503

31. Cerf NJ, Levy M, Van Assche G (2001) Quantum distribution of Gaussian keys using squeezed states. *Phys Rev A* 63: 052311. https://doi.org/10.1103/PhysRevA.63.052311

32. Charles H. Bennett, Gilles Brassard, and N. David Mermin (1992) Quantum cryptography without Bell's theorem. *Phys Rev Lett* 68: 557. DOI:https://doi.org/10.1103/PhysRevLett.68.557

33. Bai JL, Xie YM, Fu Y, Yin HL, Chen ZB (2022) Asynchronous Measurement-Device-Independent Quantum Key Distribution with hybrid source. *Opt Lett* 48: 3551–3554.

34. Lo HK, Ma X, Chen K (2005) Decoy State Quantum Key Distribution. *Phys Rev Lett* 94: 230504. https://doi.org/10.1103/PhysRevLett.94.230504

35. Gottesman D, Lo HK, Lutkenhaus N, Preskill J (2004) Security of quantum key distribution with imperfect devices. *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings*, 136. https://doi.org/10.1109/ISIT.2004.1365172

36. Liu WB, Li CL, Xie YM, Weng CX, Gu J, Cao XY, et al. (2021) Homodyne Detection Quadrature Phase Shift Keying Continuous-Variable Quantum Key Distribution with High Excess Noise Tolerance. *PRX Quantum* 2: 040334. https://doi.org/10.1103/PRXQuantum.2.040334

37. Sim DH, Shin J, Kim MH (2023) Software-Defined Networking Orchestration for Interoperable Key Management of Quantum Key Distribution Networks. *Entropy* 25: 943. https://doi.org/10.3390/e25060943

38. Shirko O, Askar S (2023) A Novel Security Survival Model for Quantum Key Distribution Networks Enabled by Software-Defined Networking. *IEEE Access* 11: 21641–21654. https://doi.org/10.1109/ACCESS.2023.3251649