A robust tripartite quantum key distribution using mutually shared Bell states and classical hash values using a complete-graph network architecture

Preprint · May 2019

DOI: 10.13140/RG.2.2.27559.39844

CITATIONS
19

READS
894

3 authors:

Kounteya Sarkar
Indian Institute of Technology Kharagpur
3 PUBLICATIONS **20** CITATIONS

SEE PROFILE

Bikash K. Behera
Bikash's Quantum (OPC) Pvt. Ltd.
174 PUBLICATIONS **1,070** CITATIONS

SEE PROFILE

Prasanta K. Panigrahi
Indian Institute of Science Education and Research Kolkata
631 PUBLICATIONS **5,900** CITATIONS

SEE PROFILE

Noname manuscript No.
(will be inserted by the editor)

A robust tripartite quantum key distribution using mutually shared Bell states and classical hash values using a complete-graph network architecture

Kounteya Sarkar · Bikash K. Behera · Prasanta K. Panigrahi

Received: date / Accepted: date

Abstract Quantum Key Distribution (QKD) is one of the major applications in the field of quantum cryptography. For security reasons, all the QKD designs have to be resilient against any attackers, who may try to passively or actively attack the quantum or classical link between the parties and gain hold of illegitimate information. Here, we propose a three party QKD using the three-qubit entangled states (GHZ state) and a complete graph type network architecture comprising both quantum and classical channels among four (or more) participants. The network architecture itself provides sufficient confusion to any potential attacker. Furthermore, we use the non-local correlations provided by two-qubit Bell states through the quantum channels, and the concept of hash values and a request-acknowledge type communication through the classical channels to add to the robustness of the scheme. We explicate the proposed algorithm in detail along with the analysis on the security of the system.

Keywords Quantum Key Distribution, Complete Graph Network Architecture, Entangled States, Bell States, Hash Values

Kounteya Sarkar
Department of Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani (Hyderabad campus), Hyderabad 500078, Telangana, India
E-mail: kounteya1000@gmail.com

Bikash K. Behera
Department of Physical Sciences, Indian Institute of Science Education and Research Kolkata, Mohanpur 741246, West Bengal, India
E-mail: bkb18rs025@iiserkol.ac.in

Prasanta K. Panigrahi
Department of Physical Sciences, Indian Institute of Science Education and Research Kolkata, Mohanpur 741246, West Bengal, India
E-mail: pprasanta@iiserkol.ac.in

1 Introduction

Quantum Cryptography and Quantum Key Distribution (QKD) are two of the foremost and realistic areas of application of quantum computing today. QKD involves the task of sharing a secret key between parties in a communication using qubits and other quantum mechanical resources. Most commonly, it involves the sharing of a secret key between two parties, which by convention, it is taken to be Alice and Bob (Alice wants to share a key with Bob), but it can also be extended to three-party (tripartite) or multi-partite QKD schemes. The two most common resources used in QKD algorithms are the mutual orthogonality of different measurement bases (e.g. the computation bases $|0\rangle$ and $|1\rangle$ and the diagonal bases $|+\rangle$ and $|-\rangle$) and entanglement of qubits (e.g. the Bell state $(|00\rangle+|11\rangle)/\sqrt{2}$). The famous BB84 protocol [1] and the B92 [2] protocol are based on qubit measurement in mutually orthogonal and non-orthogonal bases, while the E91 [3] protocol is based qubit entanglement and quantum non-local correlations.

In this work, we present a new tripartite QKD scheme, which we call Complete Network Quantum Key Distribution (CNQKD) where we use four communicating parties, three participating in the QKD plus an additional party, all mutually connected together with both classical and quantum channels, representing a complete graph like network among the participants. This architecture is proposed to provide more robustness to the system against attackers by confusing any potential attacker. We use the Bell non-local correlations between Bell states shared between different parties (like in E91 protocol) along the quantum channels and the principle of hashing, by passing only hashed values through the classical channels to provide security to the system. We propose to use maximally entangled GHZ state $((|000\rangle+|111\rangle)/\sqrt{2})$ or W state $((|001\rangle+|010\rangle+|100\rangle)/\sqrt{3})$ for the tripartite QKD, possibly as a direct extension of the ideas of the E91 protocol to three qubits or any other tripartite QKD already proposed and available in literature. To the best of our knowledge, any QKD or similar implementations using such a complete-graph like network among the participants and using the non-local correlations among Bell states along with the principle of hashing have not yet been proposed. We give the details of our proposal in detail with an example along with the security analysis which we perform to show how any potential attacker can get confused. We also simulate our proposed architecture using the IBM quantum experience platform (Ref. [4]) for the quantum channels and Python socket programming for the classical channels, thus verifying that our proposal works.

Since 2016, IBM provides the composer on its website which is a cloud-based quantum computing platform [5]. Any user can give a quantum circuit on the five-, and sixteen-qubit devices for a real run or simulation which is available with the help of QISKit Terra and use it by changing backend to perform a run or simulation. IBM Q Experience has now been used to perform a number of real experiments on the quantum chips. The real experiments include quantum simulation [6,7,8,9,10,11,12,13,14], developing quantum algorithms

[15,16,17,18,19,20,21,22], testing of quantum information theoretical tasks [9,15,23,24,25], quantum cryptography [26,27,28], quantum error correction [29,30,31,32], quantum applications [10,12,26,32,33,34,35,36] to name a few.

The organization of the paper is as follows. Section 2 describes the previous works and literature in the field. Section 3 describes our proposed architecture and algorithm in detail. Then we describe a working example of our proposed algorithm. In Section 4, we perform the security analysis along different directions of our proposal. Section 5 deals with simulation of our work and finally we conclude with some future directions and discussions.

2 Previous Works and Literature

A number of research works are continually going into QKD with many references available in literature. Apart from the protocols mentioned in the previous section [1,2,3], newer proposals and modifications are regularly being added. Gisin *et al.* [37] provides a good overview of Quantum Cryptography along with QKD. Since we are interested in tripartite QKD, Refs. [38] and [39] provide us two separate algorithms of three-party QKD, where the latter uses GHZ states. Ref. [40] is another tripartite QKD scheme using the GHZ states. Since we concentrate on our proposed network architecture and token passing to make a tripartite QKD more robust and not necessarily on the actual QKD process itself, the various three-party QKD schemes available can be used. Not only QKD, the GHZ states have also been used for other purposes such as Quantum Secret Sharing [41]. The quantum non-locality provided by Bell's theorem ([42]) exhibited by entangled states such as two qubit Bell state or three qubit GHZ state forms the fundamental background of providing security to our proposal. Ref. [43] is an excellent and comprehensive material covering all aspects of Bell's theorem, locality and non-locality, their properties and applications. Bell non-local correlations are used frequently in QKD, the E91 protocol uses it to identify potential attackers. Refs. [44] and [45] are the two examples using Bell correlations of entangled states to perform information and mutual authentication in a quantum environment.

The principle of hashing is one of the most popular and powerful concept frequently used in both classical and quantum cryptographic algorithms. The most intriguing fact of hashing is that hash values are one-way functions, i.e. we can easily compute the hash values of any given data easily but extracting the original data from the hash values is next to impossible with a good hash algorithm. Hash values are also specific and even a single bit of difference in the original data can significantly alter the hash value. Therefore it is absolutely not easy for an attacker to change the original data but have the same hash value. Chapter 11 of Ref. [46] gives a comprehensive idea about cryptographic hash functions and their usage. MD5 ([47]) and Secure Hash Algorithm (SHA) ([48]) are the two most popular and frequently used hash algorithms, but there are many more. Not only in classical computing domain, but quantum hash functions have also been proposed and under active research. Refs. [49] and

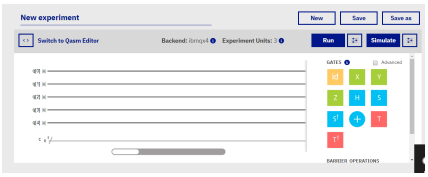


Fig. 1 The interactive GUI of the IBM Quantum Experience. Users can design, build and test their own quantum circuits, both as a simulation and on actual quantum processors.

[50] give us two such proposals of quantum hashing. However, we will be using the classical hash functions, as we propose to use the hash values only for communications along the classical channels of our network (Figure 2) and not the quantum channels.

Among the various tools available for simulation and real-life processing for quantum algorithms, the IBM Quantum Experience platform (Refer [4]) must have a special mention, since it has brought a revolution in the field of testing and simulating quantum algorithms. IBM Q offers an over-the-cloud quantum computing experience, allowing the users to create quantum circuits using a very interactive graphical user interface and test those circuits, both as a simulation (done on a classical computer) and on actual quantum processors. A number of researchers have been benefited from this unique quantum experience provided by IBM. Figure 1 shows the workplace for the IBM Q experience where users may design, build and test quantum circuits.

3 Our proposal

In our proposed three-party Quantum Key Distribution (CNQKD), the following are the salient features of our proposal. Figure 2 depicts the visual representation of the network.

- Although it is a tripartite QKD, to make our scheme resilient against attackers we use four parties connected together in a complete graph-like network (Figure 2), where each of the parties share a quantum and classical channel with each other. Our proposal in general can be extended to more number of parties and not restricted to four parties. We name our four parties as 'A', 'B', 'C' and 'D'. Without loss of generality let us assume that 'A', 'B' and 'C' are the three parties involved in key distribution.
- All the parties share maximally entangled two-qubit Bell states $((|00\rangle+|11\rangle)/\sqrt{2}$ or $(|01\rangle+|10\rangle)/\sqrt{2}$ or likewise) between each other, i.e., A and B share the states between themselves, B and C share, A and D share and so forth. Thus in general if we have 'n' parties, then we have $\binom{n}{2}$ pairs of parties who share Bell states among themselves. This mutual sharing of Bell states is from where our scheme derive its robustness.
  Furthermore the three parties involved in the QKD process share among themselves maximally entangled three qubit GHZ states (or W states)

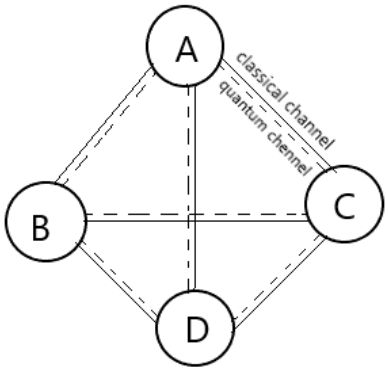


Fig. 2 The complete graph-like network among the four participants. All the four parties have both a quantum channel and a classical channel between themselves. Solid line represents classical channel while the dashed line represents quantum channel

which they use for the actual key distribution process. The process by which the key distribution takes place is open to implementation. A direct three-qubit analogue of the E91 protocol or the references provided in the previous section ([38,39,40]) can be used. In our work, we focus more on how the proposed network makes the QKD system more robust.

– Among the three parties, A, B and C who actually take part in the QKD scheme, we assume that while two parties (say A and B) use the shared key to communicate, the third party (C) may be considered to be an arbitrator between A and B in case any dispute may arise in the future between the latter parties regarding the key. For example, after the key has been shared, A and B may disagree about the shared key, so they may consult C, who also share the same key to settle the dispute. C thus acts as a trusted third party who may be consulted for any arbitration. Trusted third party schemes are very common in key distribution, both in the classical and quantum realm. The Kerberos (See Ref. [54]) architecture is an example of trusted third-party scheme in classical cryptography, while Ref. [38] and Ref. [55] are two examples in quantum cryptography using the concept.
– While A, B and C are involved in the actual QKD, the fourth party D also plays a key role in the network, as although D does not take part in the QKD, it functions in the mutual Bell state measurement around the network discussed next, which provides robustness to the system and confuses any potential attacker. In fact, without the participation of all the four participants our proposed algorithm is not complete. A step-by-step example of our algorithm is given in the next section, including how the three parties for the actual QKD is selected, the left out candidate then automatically becomes the fourth party in the process.
– As already mentioned, the parties mutually share two qubit Bell states $((|00\rangle+|11\rangle)/\sqrt{2})$ between them. In regular intervals of time, each of the

nodes sends one of the qubit of the two-qubit entanglement to any of the other party via the direct quantum channel that all the nodes have among themselves. Upon receiving the qubit, the two nodes (i.e. the sender and the receiver of the Bell state qubits) perform a Bell inequality measurement on their qubits. If the inequality satisfies the Bell's condition (or the CHSH condition for a tighter bound [43]) then by Bell's theorem the correlation between the two qubits does not remain non-local anymore and the nodes are aware that the quantum channel must have been compromised by an attacker. These two parties then take a note of this fact, i.e. their shared quantum channel is no longer secure. If on the other hand, the inequalities fail, then the correlations remain non-local and the quantum channel is secure. This process is carried out repetitively by each of the nodes with each of the other nodes in regular intervals to ascertain the security of the shared quantum channel. Note that because each node has a direct channel with every other node, this process can be carried out independently between different nodes and also between one nodes with two or more other nodes at the same time. Our complete-graph like architecture achieves robustness in this manner. How this same approach confuses any attacker is discussed shortly. It is important to note that this mutual sending of Bell state and non-local correlation measurement among each of the nodes continues to happen regularly over the quantum channels and independently of any messages being transmitted through the classical channel. Thus even though at any time no actual QKD is under way the Bell measurements never cease but keep on in regular intervals.

– The four parties in our consideration is known only among themselves and do not reveal their identity to outside (refer Section 1). Hence, all the four nodes appear same to any outsider or attacker with no distinction. This adds to the confusion of the attacker as to among whom the key distribution is taking place. But then how does the four nodes agree among themselves and initiate and terminate the algorithm. This is achieved using hashing and sending the hash value through the classical channel. Here we propose that each of the party in our network has with itself a table which it shares uniquely with every other party. Table 1 shows an example of two tables that A has, one with node B the other with C. To increase security we further propose, as clearly seen in Table 1, the tables shared between different parties are all different, which the sharing parties may decide among themselves. The table contains some pre-decided messages and the hash algorithm (e.g MD5, SHA etc) that they intend to use between themselves. Whenever any party, say A wants to initiate a QKD with B and C, it refers to its unique table that it shares with B and C, chooses the appropriate message, calculates the hash of the message and sends the message to the respective recipient through the classical channel. Because of the wonderful one-way property of the hash function (ref Section 1 and Table 3), any attacker snooping in the classical channel gets no way to extract the meaning of the hash even though holding of the hash message.

Table 1 A possible example of two tables that A shares each with B and C, NOTE that the table contents are different for similar meaning fields as required and B and C also share the same copy with A. The unique feature of the hash functions ensure that the hash values will be different even for a single bit of difference in the original inputs. The hash algorithms to be used among the parties are also different. The tables are known only among the two parties who share the tables and are kept completely secret otherwise. The parties may agree on the table contents when they first meet.

Sample table shared between A and B	
HASH-ALGO	MD5
REQUEST SHARE- KEY	
REQUEST THIRD P- C	
REQUEST THIRD P-D	
REQUEST TERMINATE KEY	
ACKNOWEDGE	
REQUEST ARBITRAR- C	
OK ARBITRAR WITH-C	
.....	
.....	

Sample table shared between A and C	
HASH-ALGO	SHA512
START QKD	
KEEP THIRD PARTY-B	
KEEP THIRD PARTY-D	
DISCARD KEY	
ACK	
BE MY ARBITRAR WITH-B	
OK WITH ARBITRAR-B	
.....	
.....	

– When the recipient of the hash message receives a message through a classical channel, it knows who is the sender. This is achieved by the fact that each node has a unique direct connection to every other node because of the complete-graph like network. The recipient then refers the table that it shares with the sender and computes the hash of the messages according to the specified algorithm and compares it with the received hash. If the hashes match then the recipient knows what is the message and sends an acknowledgement in the same manner, otherwise if there is no match the recipient does nothing. In this way two parties agree with each other over the classical channel regarding the key distribution. Furthermore the two parties, by using the same hashing technique and consulting the tables at their side, agree on their third party, and they both communicate the same to the third party regarding their decision by the same technique. The third party can then send an acknowledgement to both the other two parties and the three-part QKD can then begin through the quantum channel. Since the fourth party does not receive any hash messages over its classical channels it is unaware of any communication between the other three nodes

and continues with its Bell state sending, receiving and measuring over its quantum channel as usual.

- Although the attacker has no way of extracting the meaning of the hash message, it can still jumble up the hash to confuse the valid entities. To overcome this we propose the following. First, we impose the condition that whenever any recipient receives a hash message, if it is able to extract the meaning it should send an acknowledgement back to the receiver (using the same hash technique and referring their mutually shared table) in case it is unable to extract the meaning it considers the message to be junk and does not send a reply. The sender waits for some time, and upon not receiving any reply understands that something might be wrong. Second, as we have already mentioned in the previous points, the exact structure and the content of the table which the parties refer for constructing/extracting the meaning of the hash and share is different for each parties, hence the tables that A shares with B is different from what it shares with C or D. The tables are necessarily quite long as it must contain every possible valid conversation between the two parties (Table 1) and maintaining such expansive tables at each node for each other node in the network is definitely a challenge, especially if the number of nodes are large. However, this is a necessary overhead that we require to increase the security. In this way, even if by some mean an attacker gets hold of the information of one table, it cannot know the contents of the other tables as they are all different.
- Even though the attacker has no means of decoding the hash message, it can nevertheless still snoop into the channels and figure out the parties in communication by identifying the source and destinations. To prevent this, as a final security measure we propose that, like the regular sharing of Bell states through the quantum channels, all the parties regularly share garbage hash values with each other to confuse the attacker. A participant who receives a garbage hash, can immediately verify its authenticity by checking his table, and upon non finding any match will understand it as a garbage. This sending and receiving of garbage hash also goes on regularly independent of other communications. For any valid hash, the senders and recipients will be able to decode the message, whereas for garbage hash values they will simply ignore.
- If by any reason the quantum channel between a sender and receiver has been compromised, they both can detect it by measuring the non-locality correlations as discussed above. In such a scenario even if a receiver receives a valid hash from a sender, it does not send an acknowledgement back. The sender will also be aware of the compromised quantum channel through the measurement on its own qubit and also when it does not receive any acknowledgement from the recipient. They then immediately cease any operation that were doing, such as sharing of GHZ states for the QKD and try again later when conditions may improve.
- When the third party gets selected to act as the arbitrator by two other parties, and all the respective hash messages and their acknowledegements in this regard have been transmitted, upon sensing the quantum channel to

be free of any intruder by Bell measurement, the third party now prepares three-qubit GHZ states, and sends two qubits of each state to each of the two other nodes and keeps the third with itself. After all the GHZ state qubits have been transferred, the three parties may then mutually agree on a key based on their shared qubits of the GHZ states by any method mentioned in previous literature or as a direct extension of the E91 protocol to three qubits. If in this process at any time any party detects an intruder in their quantum channels by Bell measurement, they immediately stop the protocol.

– Finally we state two important assumptions for our scheme. First we assume that communication between two nodes, whether classical or quantum can only happen through the direct channel shared between the nodes as we have a complete network among the nodes. For example if A wants to send something to C, it can only send via the direct channel it has to C, not by routing through B or D. Secondly, we must address the issue of how the tables that we have talked about gets created in the first place, especially when all are different. For this we assume that during the formation of the network, or when a new node joins the network, it must physically interact with all the other nodes in a secure way and at that time of interaction all the nodes can decide upon their unique table, as well as establish both the classical and quantum channel. No doubt this condition of physical interaction at the beginning can seem as a disadvantage, but once the tables have been shared and the channels established, the nodes can communicate securely.

Table 2 Table showing how the communication takes place along the two types of channels

Quantum channels	Qubits, both Bell state qubits and GHZ state qubits
Classical channels	Only hash values, both garbage values to confuse the attacker and valid messages

Having listed the salient points of our scheme, let us take an example and see how the algorithm runs from the beginning. Again, without loss of generality we assume that A wants to initiate the QKD with B and they both want C to act as the third party, D does not directly take part in the actual QKD scheme. Figure 2 and the Table 1 can be observed for consultation. Table 2 shows in brief the entities that actually are transferred through the quantum and classical channels.

3.1 CNQKD: An Example

1. A decides that it wants to share a key with B where C acts as the arbitrator. A checks its table that it shares with B, selects the appropriate item (for

example "I want a key with you" as according to Table 1), hashes it with algorithm mentioned in the table and sends the hash to B along the classical channel that it shares with B.

2. As B receives the hash, it checks its own table that it shares with A, computes the hashes of the table items according to the specified algorithm one by one and compares the same with the hash value that it has received from A. If it matches with "I want a key with you", going by the example in the previous point, it knows that A wants to share a key with itself. B then sends the hash of the acknowledgement to A.
3. Having received the acknowledgement from B, A now wishes to communicate B regarding its desire to have C as their trusted third-party. In a similar manner, A now consults its table and sends B the hash of the corresponding message. B receives the hash, extracts its meaning and if it is willing to have C as the third entity, sends its acknowledgement to A, otherwise sends an error/regret message.
4. A and B both are in agreement about C, so now both A and B sends their request, (something like "I want you to be the arbitrator between me and B" from A's perspective, and similar from B's perspective) to C. After receiving *the messages from both A and B*, C sends an acknowledgement to A and B. Note that C will send its acknowledgement to both A and B *only* after it receives the request messages from both, otherwise it does not send anything. Since in our algorithm an acknowledgement is a must, if A and B, even after sending the request to C does not receive a reply, they know that something must be wrong and they forfeit their current process. If both receives acknowledgement from C then A and B further sends a ready-to-receive message (of course not the original but the hash) to each other and to C, indicating that they are now ready to receive the qubits from C.
5. C now prepares maximally entangled three qubit states (e.g. GHZ states) and sends two of the qubits to A and B and keeps the third with itself. However, sending to A and B, C ensures that the quantum channel between it and A and B are not compromised by performing a Bell inequality test on the qubits it had already shared with each of A and B. After sending the first pair of GHZ qubits, C then repeats the process over and over again, until sufficient qubits have been shared from which the required n-bit secret key can be extracted. As said before the exact process of the key distribution using the GHZ states have been left open for implementation. It can be a direct extension of the E91 protocol, like each of the three parties A, B and C measure the GHZ state qubit that they hold in three separate measurement bases, and later publicly compare and choose only those qubits for which all the three parties measured in the same basis, and discard the other qubits. Thus now A, B and C share a unique secret key among themselves, and A and B can now use the key to communicate with each other and consult C if any dispute may arise in the future regarding the key.

6. In all the while the fourth party D did not play any role in the actual QKD. Nevertheless D, along with the other three parties regularly checks for the integrity of the various quantum channels of the network by sharing and measuring non-local correlations of Bell states as already mentioned. Thus all the while A, B and C were engaged in QKD, they were all communicating with D and among themselves through the quantum channels to test the channel's integrity.

-

Having listed the steps of our algorithm in detail, in the next section we analyze the security of our proposal.

4 Security Analysis

Having given the details about our algorithm (CNQKD) and an example, let us now examine the most important factor of our algorithm, the robustness that it provides against a potential attacker. We will analyze the security of our approach in three ways, first by the very nature of our proposed complete-graph like network, second by the mutually Bell state qubits shared by the nodes through the quantum channels and finally by the use of hash values only through the classical channels. We show that our proposal is indeed robust and secure.

4.1 Security provided by the network architecture

We have proposed a complete-graph like network architecture (Figure 2) for our approach. What security does this particular architecture imparts to the system? To understand this consider Figure 3 that shows a typical QKD network among two parties A and B. This network has a vulnerable architecture, as there is only one direct link between the parties. It is easy for an attacker to compromise the network, since all it has to do is either snoop in or manipulate the direct link between the nodes.

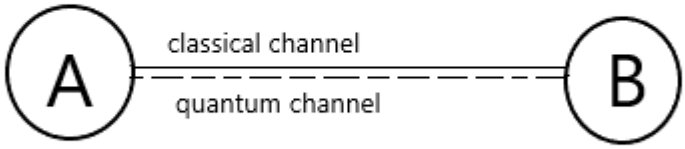
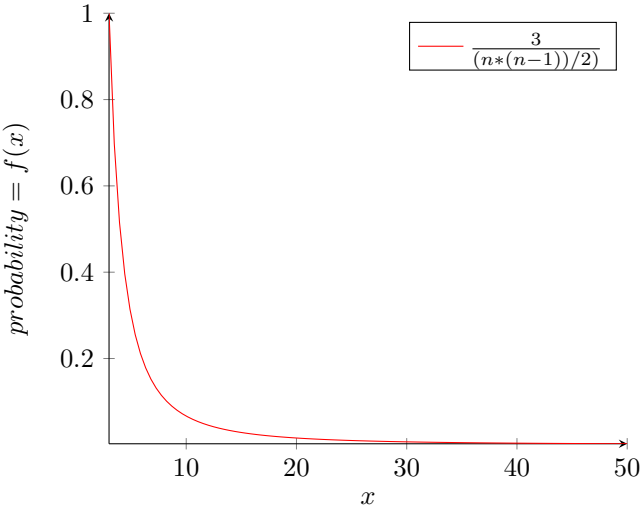


Fig. 3 A typical network of classical and quantum channels between two parties.

Because there is only one quantum and one classical channel, all communication that happens between the nodes must happen through these channels only. The network architecture of Figure 3 itself does not provide any security, the two parties have to rely only on the algorithm they are using to provide them the security. Now consider our proposed network (Figure 2). The very architecture of the network consists of $\frac{n(n-1)}{2}$ channels (each for classical and quantum) for a network with 'n' nodes (property of a complete graph). The nodes completely hide their details from the external world, and are only known to themselves. In addition, they continuously send each other Bell state qubits along the quantum channel and hash values (garbage hash in case there is no actual communication) from time to time. All these make a potential attacker confused. To see this consider an attacker snoops into the channels between A and B. The attacker detects some qubits in transition through the quantum channel and some hash values through the classical channel. But since A and B continuously send each other qubits and hash values, the attacker has no way of knowing whether the qubits are the GHZ qubits to be used in the actual QKD, or the Bell qubits. The attacker cannot decode the meaning of the hash value also (one-way property), neither it has a copy of the tables (1 so it cannot know whether the hash value is a garbage or some valid message. At most, the attacker can capture and measure the qubits and send some other qubits and also distort the hash message. However, A and B will immediately know of the attackers' presence by the failure of Bell's nonlocality when they measure on their qubits, as well as when they find the hash does not match with any valid message. Thus compared to Figure 3, where an attacker has a 100% chance of attacking the channels, in our proposal (Figure 2) it has $\frac{3}{[n(n-1)]/2}$ (because of tripartite QKD) probability of correctly attacking the required channels. If we increase the number of nodes ('n'), then the system becomes even more robust, as the probability of correctly guessing the

channel to attack decreases fast, referred to the below Graph. The horizontal axis shows the number of nodes in the network and the vertical axis shows the probability. It is to be noted that since this is a tripartite QKD if we have three parties, the attacker has 100% chance of correctly attacking, but it decreases rapidly with increase in nodes.



Another advantage of our network architecture is that it is *resilient* upto a certain factor even after being attacked. For example consider the actual QKD which is under process between A, B and C, when an attacker attacks the system. As said before, in this case it has a $\frac{3}{6}$ (4 nodes) of correctly guessing a channel to attack. Suppose that it guesses wrong and it attacks the channel between B and D. Both B and D immediately becomes aware of the B-D channel being compromised, but as the actual QKD is under process along the A-B-C channels, the three parties can still continue to carry on the process. This is something which is not possible in the simple architecture (Figure 3).

4.2 Security along the quantum channels

For security along the quantum channels, we have used the Bell non-local correlations provided by the entangled qubits in Bell states. As the parties in the network continuously exchange Bell qubits and measure the Bell inequality for non-locality, the quantum channels are always under active scrutiny by all the parties. Any action by the attacker will be immediately exposed to the parties, who can then take appropriate measures. The resilience of our proposed network discussed in the previous section ensures that a QKD can still keep on happening, even if the attacker gains access to some channels other than those which are being used for the actual QKD.

4.3 Security along the classical channels

Finally, we analyze the security of the classical channels. The nodes send only hash values along the classical channels, both garbage and valid messages. The attacker can snoop into the classical channels and get hold of the hash values, but it cannot extract the meaning from the hash values. It cannot differentiate between a garbage and a valid hash value. At most, it can distort the hash values and send the distorted value to the nodes. However, since any valid hash message must correspond to one of the messages in the tables that the nodes share, and because it is almost impossible to have the same hash value for two different messages (property of hash value), if a node receives a distorted value, it simply treats the message as another garbage value, and does nothing. Without the access to the tables itself (which the nodes keep secret and private) the attacker can never compromise the classical channels also.

5 Example and Simulation

In this section, we go ahead to present a sample working example of our architecture and also provide a simulation of our proposal.

5.1 Quantum channel simulation

According to our complete-graph network, all the nodes are connected with a classical as well as a quantum channel. Through the quantum channel the parties share only entangled qubits among themselves, either Bell qubits or GHZ state qubits. To emulate such a behavior we assume that the respective nodes have a quantum circuit to generate a Bell pair or a GHZ state extended among themselves. Figure 4 is a circuit for generating Bell states and Figure 5 is the circuit for generating GHZ states. These circuits have been simulated on the IBM Quantum computer, the quantum-computing-over-the-cloud platform provided by IBM ([**?**]).

In the two circuits shown, the entire circuit, as said before is extended across the relevant communicating parties, each of the qubits shared by each relevant party. For the Bell state circuit, for example the sender has the top qubit in its possession while the receiver has the bottom one in its possession. For the GHZ state circuit, which is used for the actual key transfer, the arbitrator may be the creator of the GHZ state, who has the top most qubit in its possession, while the two other nodes, among whom the actual QKD takes place has the bottom two qubits in its possession.

In order to show that the above two circuits actually produce the required outcomes, i.e. the Bell and the GHZ state, we have also tested our circuit on the IBM quantum experience platform. Figure 6 and 7 show the result outcomes of the above two circuits. Figure **??** shows the result of the GHZ state

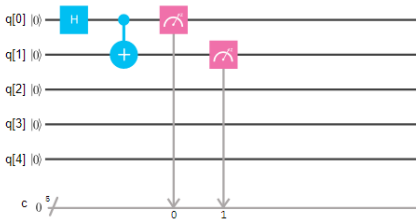


Fig. 4 A circuit for generating a Bell pair. The entire circuit is extended from, say node A to node B, A has the possession of the upper qubit while B has the possession of the lower qubit. The two measurement gates are added to verify that the circuit indeed produces Bell state.

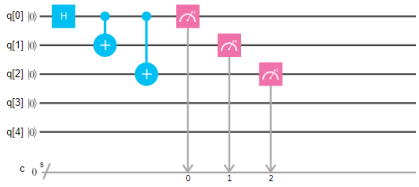


Fig. 5 A circuit for generating a GHZ state. The entire circuit is extended from the generator of the GHZ state, say node C (acting as the arbitrator) to nodes A and B, who are the actual parties in the QKD. C has the possession of the upper qubit while A and B has the possession of the lower qubits. The three measurement gates are added to verify that the circuit indeed produces the GHZ state

circuit executed on the actual quantum computer. It is to be noted that in the actual execution of a quantum circuit there are always some finite probability of getting results other than those desired. Both the circuits have been tested for multiple number of times, and as we see, the circuits indeed produce the required results. For the Bell state circuit, half of the time we get '11' corresponding to $|11\rangle$ of the Bell and the other half we get '00' corresponding to $|00\rangle$. Similarly, for the GHZ state, where half of the time we get $|000\rangle$ and the other half of time we get $|111\rangle$ upon measurements.

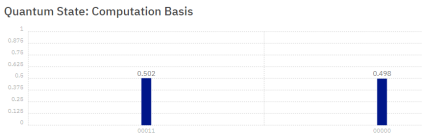


Fig. 6 Graph showing the outcome of the Bell state circuit after multiple runs. The results are to be read from right (LSB) to left (MSB). The graph shows that the circuit indeed produces the expected results.

In our architecture we have a quantum channel between each of the participants. We assume that the said participants have the respective quantum circuits shared among themselves along with the respective qubits. To have

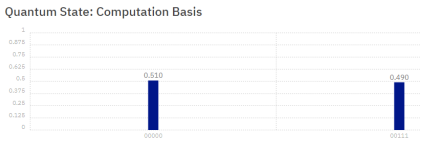


Fig. 7 Graph showing the outcome of the Bell state circuit after multiple runs. The results are to be read from right (LSB) to left (MSB). The graph shows that the circuit indeed produces the expected results.

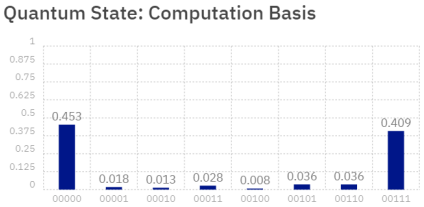


Fig. 8 Graph showing the outcome of the GHZ state circuit run on an actual quantum processor. It is to be noted that outputs other than those desired are also present, as a real life quantum processor is susceptible to noises.

an idea, let us consider the following example of a communication along the quantum channel between A and B. We will consider the mutual sharing of both the Bell state qubits and GHZ state qubits between A and B.

i. Bell state - Like all other nodes in the network, A and B continuously share between them one qubit each of a Bell pair to verify the integrity of the circuit. A and B has a quantum circuit to generate the Bell pair (4) which extends from A to B. A is in possession of the top qubit, while B has the bottom qubit in its possession. Whenever A wants to send one qubit of a Bell pair to B, it initializes two qubits to $|0\rangle$, creates and extends the Bell circuit all over to B, makes the circuit to act on the two qubits and finally sends the bottom entangled qubit to B. A and B then both perform a Bell inequality measurement on the qubits held by them to ascertain the integrity of the quantum channel.

ii. GHZ state - The working and the use of the GHZ state is similar to above. Let us now consider that A, B and C are the three parties in the actual QKD, and that A and B are the two parties who actually share a quantum secret key and that C is the arbitrator. Whenever the three parties agree to start the key distribution process, the arbitrator, in this case C, creates the GHZ state circuit (Fig 5) and extends its across itself and the other two parties, A and B. C has with itself the top most qubit while A and B have the bottom two qubits. C now initializes the three qubits in $|0\rangle$ state, applies the GHZ circuit to the three qubits to prepare the entanglement and finally keeps the top qubit while sending the other two qubits to A and B. Now A, B and C can initiate any entanglement based three-party QKD with the GHZ qubits.

5.2 Classical channel simulation

We use the well defined client server architecture to simulate the classical channels and use Python programming language to demonstrate our idea. We assume that each node acts both as the client and the server. Whenever a node wants to send some information to any other node through the classical channel, that node acts as the client and the other node acts as the server. Whenever any node wants to reply to messages it has received from any other node, then this node acts as the server and the other node acts as the client. For simplicity we write separate client-server Python programs for each of the nodes and run them on local host instead of running them on physically separate computers as logically both approaches are the same. Each Python code has both a client and server component and has the necessary functions to implement all the functionalities for communication of the nodes along the classical channels.

In order to fully appreciate the working of the classical channel, let us first see the power of hash values. Table 3 below shows the MD5 (one among the various hashing algorithms) hash values for three binary bit strings as input. All the three bit strings differ with each other by only one bit value, but notice how the corresponding hash values are so widely different.

Table 3 Table showing MD5 hash values in hexadecimal corresponding to different bit strings as input. Notice how the hash values are starkly different even when only one bit is changed in the input.

Inputs	MD5 Hash values
10001110011	258a0defbc1e0a6af22e51ec29f3ad66
10001100011	440d528bf2e407e5235fb9aa0d52c65a
10001100001	61939c3f4161a6c91a1a0d095e67f770

From the above table **??**, we observe that even a small change in the input makes a significant change in the hash values. This provides one security aspect of the hash values. The other security aspect of the hash values is its one-way property, i.e. while it is easy to compute the hash value of any given input, it is difficult for the reverse function, i.e. given a hash value it is difficult to get back the original input. These two features make the use of only hash values along the classical channel robust, as even if an attacker has access to the channel and gets the hash values, the attacker has no way of knowing the meaning of the hash value unless he/she has access to the communication tables shared between various parties. The attacker can at most capture the ongoing hash messages and distort them with the hope of confusing the legitimate parties. However, since the hash values are so sensitive to their inputs it will be almost difficult for the attacker to have exactly the same hash vales for any other message and if the hash values are indeed distorted, the moment the legitimate

receiver of the hash value finds that the received hash value does not match with any one on its table, it will simply discontinue with the communication.

As said before we simulate the behaviour of each nodes by writing a separate client and server programs for each node. Each node when wanting to send some requests acts as the client and when wanting to respond to some requests acts as the server. A sample program for one such node, say node 'A' in our example is given at the end. Both the client and server programs are shown for node A. For simplicity only the communication between A and B has been simulated and the results are shown. The other communications between the nodes are similar. Figure 9 shows the output of the run of a client program and Figure 10 shows the action of the server on receiving a client request.

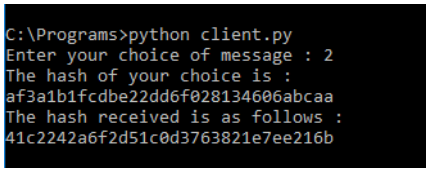


Fig. 9 The output of a single run of the client side program. Note how only hash values are the ones that are sent.

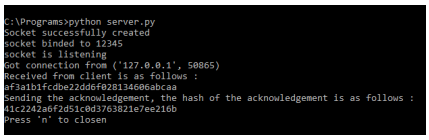


Fig. 10 The output of a single run of the server side program showing the responses by a server upon receiving some request from the client.

We can observe that the client sends only the hash value of a valid message taken from one of the messages in the unique table shared between the nodes, while the server replies with an acknowledgement which is also in the hash value form. In order to truly appreciate that only hash values are exchanged through the classical channel we show the screenshot of Wireshark network capture while the communication along the classical channels was in progress. Figure 11 can be referred to see the capture. The actual data that was transmitted has been highlighted. Compare the data captured by Wireshark with the output of the server (Figure 10) to see that indeed the hash values only are transmitted along the classical channel.

We thus see that through the use of hash values as the only means of communication through the classical channel, the classical channels remains secure. Of course, as we have said earlier, by our algorithm not only hash

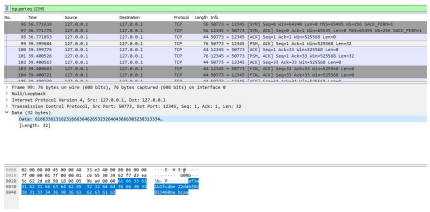


Fig. 11 Screenshot of a Wireshark capture of a communication between the client and server. The actual data that was transmitted has been highlighted. Notice that it is also a hash value.

values of valid messages but also dummy hash values are continuously passed among the nodes to confuse any attacker even more.

6 Conclusion and Discussions

Having seen the details of our proposal CNQKD, along with the security analysis and simulation along both classical and quantum channels here we present some discussions about our work and provide the conclusion along with the prospect of future work. We have given a robust three-party QKD proposal whose robustness is based on a complete-graph like architecture among all the participating nodes. Although it is a three-party QKD we can have more nodes in our architecture to provide security. We have shown only with four nodes, but our architecture can be extended with even more number of nodes if required. As clearly seen from the graph given in the 'Security Analysis' section, the probability of a successful attack by an attacker asymptotically reaches zero with an increase in the number of nodes. Even if we have 10 nodes in our system the probability of a successful attack drops to 0.066 which is quite low. The actual algorithm to be employed for the tripartite QKD is not discussed as there are many algorithms available in the literature for the QKD, rather we have focused more on the complete-graph like architecture which provides the robustness to our system.

In our architecture we have both classical and quantum channels between each of the nodes. To provide the security along both the channels, we have proposed that the nodes share with each other Bell state qubits regularly and measure the Bell non-locality conditions to check for any infiltrator along the quantum channels, and only hash values along the classical channels. The hash values are not any random, but are chosen from only a set of valid messages based on an unique table shared between every two parties in our architecture listing all the valid communications that may be required by any node. The contents of the tables are agreed upon by the nodes initially while forming the network by some means, for example by physically meeting each other. To have further security the nodes send each other dummy hash values through the classical channels from time to time to prevent any replay or timing attacks by the attacker.

The future works of our proposal lie in quite a few areas, a most important being a mechanism by which the nodes can agree upon their uniquely shared tables which incurs a much less overhead. We finally conclude by saying that our proposal provides a robust tripartite Quantum Key Distribution using maximally entangled state qubits by using an unique complete-graph like architecture. We hope, that if implemented our proposal can be used in a real life tripartite QKD scenario which requires strong security and robustness.

Acknowledgments

KS would like to thank Suvadip Batabyal from BITS Pilani (Hyderabad Campus) for the immense help and motivation for the work. We also thank the IBM Quantum Experience project for using real quantum computers. The discussions and opinions developed in this paper are only those of the authors and do not reflect the opinions of IBM or IBM QE team.

References

1. C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," in Proceedings of IEEE International Conference on Computers, Systems and Signal Processing, 1984.
2. C. Bennett, Quantum cryptography using any two nonorthogonal states. Phys. Rev. Lett. **68(21)** 3121-3124 (1992).
3. A. Ekert, Quantum cryptography based on Bells theorem, Phys. Rev. Lett. **67(6)** 661-663 (1991).
4. https://quantumexperience.ng.bluemix.net/qx/editor
5. IBM Quantum Experience, URL: https://www.research.ibm.com/ibm-q/.
6. K. Halder, N. N. Hegade, B. K. Behera, and P. K. Panigrahi, Digital Quantum Simulation of Laser-Pulse Induced Tunneling Mechanism in Chemical Isomerization Reaction, arXiv:1808.00021.
7. G. R. Malik, R. P.Singh, B. K.Behera, and P. K.Panigrahi, First Experimental Demonstration of Multi-particle Quantum Tunneling in IBM Quantum Computer, DOI: 10.13140/RG.2.2.27260.18569.
8. D. Aggarwal, S. Raj, B. K. Behera, and P. K. Panigrahi, Application of quantum scrambling in Rydberg atom on IBM quantum computer, arXiv:1806.00781.
9. P. K. Vishnu, D. Joy, B. K. Behera, P. K. Panigrahi, Experimental demonstration of non-local controlled-unitary quantum gates using a five-qubit quantum computer, Quantum Inf. Process. **17**, 274 (2018).
10. M. Schuld, M. Fingerhuth, and F. Petruccione, Implementing a distance-based classifier with a quantum interference circuit, Europhys. Lett. **119**, 60002 (2017).
11. S. S. Tannu, and M. K. Qureshi, A Case for Variability-Aware Policies for NISQ-Era Quantum Computers, arXiv:1805.10224.
12. J. R. Wootton, Benchmarking of quantum processors with random circuits, arXiv:1806.02736.
13. Manabputra, B. K. Behera, and P. K. Panigrahi, A Simulational Model for Witnessing Quantum Effects of Gravity Using IBM Quantum Computer, arXiv:1806.10229.
14. O. Viyuela *et al.*, Observation of topological Uhlmann phases with superconducting qubits, npj Quantum Inf. **4**, 10 (2018).
15. D. García-Martín, and G. Sierra, Five Experimental Tests on the 5-Qubit IBM Quantum Computer, J. App. Math. Phys. **6**, 1460 (2018).

16. R. Jha, D. Das, A. Dash, S. Jayaraman, B. K. Behera, and P. K. Panigrahi, A Novel Quantum N-Queens Solver Algorithm and its Simulation and Application to Satellite Communication Using IBM Quantum Experience, arXiv:1806.10221.
17. M. Sisodia, A. Shukla, K. Thapliyal, A. Pathak, Design and experimental realization of an optimal scheme for teleportaion of an n-qubit quantum state, Quantum Inf. Process. **16**, 292 (2017).
18. S. Gangopadhyay, Manabputra, B. K. Behera and P. K. Panigrahi, Generalization and Demonstration of an Entanglement Based Deutsch-Jozsa Like Algorithm Using a 5-Qubit Quantum Computer, Quantum Inf. Process. **17**, 160 (2018).
19. S. Deffner, Demonstration of entanglement assisted invariance on IBM's quantum experience, Heliyon **3**, e00444 (2017).
20. İ. Yalçinkaya, and Z. Gedik, Optimization and experimental realization of quantum permutation algorithm, Phys. Rev. A **96**, 062339 (2017).
21. K. Srinivasan, S. Satyajit, B. K. Behera, and P. K. Panigrahi, Efficient quantum algorithm for solving travelling salesman problem: An IBM quantum experience, arXiv:1805.10928.
22. A. Dash, D. Sarmah, B. K. Behera, and P. K. Panigrahi, Exact search algorithm to factorize large biprimes and a triprime on IBM quantum computer, arXiv:1805.10478.
23. E. Huffman and A. Mizel, Violation of noninvasive macrorealism by a superconducting qubit: Implementation of a Leggett-Garg test that addresses the clumsiness loophole, Phys. Rev. A **95**, 032131 (2017).
24. D. Alsina, and J. I. Latorre, Experimental test of Mermin inequalities on a five-qubit quantum computer, Phys. Rev. A **94**, 012314 (2016).
25. A. R. Kalra, N. Gupta, B. K. Behera, S. Prakash, and P. K. Panigrahi, Demonstration of the No-Hiding Theorem on the 5 Qubit IBM Quantum Computer in a Category Theoretic Framework, Quantum Inf. Process. **18**, 170 (2019).
26. B. K.Behera, A. Banerjee, and P. K.Panigrahi, Experimental realization of quantum cheque using a five-qubit quantum computer, Quantum Inf. Process. **16**, 312 (2016).
27. M.-I. Plesa and T. Mihai, A New Quantum Encryption Scheme, Adv. J. Grad. Res. **4**, 1 (2018).
28. A. Majumder, S. Mohapatra, and A. Kumar, Experimental Realization of Secure Multiparty Quantum Summation Using Five-Qubit IBM Quantum Computer on Cloud, arXiv:1707.07460.
29. D. Ghosh, P. Agarwal, P. Pandey, B. K. Behera, and P. K. Panigrahi, Automated Error Correction in IBM Quantum Computer and Explicit Generalization, Quantum Inf. Process. **17**, 153 (2018).
30. J. Roffe, D. Headley, N. Chancellor, D. Horsman, and V. Kendon, Protecting quantum memories using coherent parity check codes, Quantum Sci. Technol. **3** 035010 (2018).
31. S. Satyajit, K. Srinivasan, B. K. Behera, and P. K. Panigrahi, Nondestructive discrimination of a new family of highly entangled states in IBM quantum computer, Quantum Inf. Process. **17**, 212 (2018).
32. R. Harper and S. Flammia, Fault tolerance in the IBM Q Experience, arXiv:1806.02359.
33. A. Dash, S. Rout, B. K. Behera, and P. K. Panigrahi, A Verification Algorithm and Its Application to Quantum Locker in IBM Quantum Computer, arXiv:1710.05196.
34. U. Alvarez-Rodriguez, M. Sanz, L. Lamata, and E. Solano, Quantum Artificial Life in an IBM Quantum Computer, Sci. Rep. **8**, 14793 (2018).
35. B. K. Behera, S. Seth, A. Das, and P. K. Panigrahi, Demonstration of entanglement purification and swapping protocol to design quantum repeater in IBM quantum computer, Quantum Inf. Process. 18, 108 (2019).
36. B. K. Behera, T. Reza, A. Gupta, and P. K. Panigrahi, Designing Quantum Router in IBM Quantum Computer, arXiv:1803.06530.
37. N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, Quantum cryptography, Rev. Mod. Phys. **74** 145 (2002).
38. M. Alshowkan, K. Elleithy, A. Odeh, and E. Abdelfattah, A New Algorithm for Three-Party Quantum Key Distribution, Third International Conference on Innovative Computing Technology (INTECH 2013).
39. G. L. Khym, W. Y. Chung, J. I. Kim and H. J. Yang, H. Y. Lee and C. H. Hong, Quantum Key Distribution among Three Parties Using GHZ States, J. Korean Phys. Soc. **44(6)**, 1349-1357 (2004).

40. Y. Guo, R. Shi, and G. Zeng, Secure networking quantum key distribution schemes with GreenbergerHorneZeilinger states. Physica Scripta, **81(4)**, 045006 (2010).
41. M. Hillery, V. Buzek, and A. Berthiaume, Quantum secret sharing, Phys. Rev. A, **59(3)**, 1829-1834 (1999).
42. J. S. Bell, On the Einstein Podolsky Rosen Paradox, Physics **1**, 195-200 (1964).
43. N. Brunner, D. Cavalcanti, S. Pironio, V. Scarani, and S. Wehner, Bell nonlocality, Rev. Mod. Phys. **86(2)**, 419-478 (2014).
44. X. Li and H. Barnum, Quantum Authentication Using Entangled States, Int. J. Found. Comput. Sci. **15(04)** 609-617 (2004).
45. M. Alshowkan and K. Elleithy, Quantum Mutual Authentication Scheme Based on Bell State Measurement, IEEE Long Island Systems, Applications and Technology Conference (2016).
46. W. Stallings, *Cryptography and Network Security, Principles and Practice* (5th ed., pp. 327-356). Boston: Pearson Education (2011).
47. RFC 1321, *Retrieved from* https://www.ietf.org/rfc/rfc1321.txt
48. RFC 4634, *Retreived from* https://tools.ietf.org/html/rfc4634
49. Y. Yang, P. Xu, R. Yang, Y. Zhou, and W. Shi, Quantum Hash function and its application to privacy amplification in quantum key distribution, pseudo-random number generation and image encryption, Sci. Rep. **6**, 19788 (2016).
50. F. Ablayev, and A. Vasiliev, Quantum Hashing, *Retreived from* https://arxiv.org/pdf/1310.4922.pdf(2013).
51. GeeksForGeeks https://www.geeksforgeeks.org/computer-network-controlled-access-protocols/
52. I. Suzuki and T. Kasami, A distributed mutual exclusion algorithm, ACM Trans. Comput. Sys. **3(4)**, 344-349 (1985).
53. S. Banerjee, and P. Chrysanthis, A new token passing distributed mutual exclusion algorithm, Proceedings Of 16Th International Conference On Distributed Computing Systems, (1996).
54. W. Stallings, Cryptography and Network Security, Principles and Practice (5th ed., pp. 452-469). Boston: Pearson Education, (2011).
55. Y. Kanamori, B. Hoanca, and S. Yoo, Three-Party Quantum Authenticated Key Distribution with Partially Trusted Third Party. IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference, (2008).