Bi-symmetric Key Exchange: A Novel Cryptographic Key Exchanging Algorithm

Shekhar Sonthalia, Trideep Mandal and Mohuya Chakraborty

Abstract The most sensitive part of any crypto graphical system is ensuring the security of key exchange. Classical crypto graphical algorithms make use of one-way complex mathematical functions, to encode the key. On the other hand, Quantum crypto graphical systems make use of Q-bits or photons to encode the key. These methods are useful to secure the key but they come with a lot of trade-offs. Classical one is complex and requires a lot of mathematical calculations and is easy to use as it does not involve much hardware but with quantum computing on the rise, it is a piece of cake for an eavesdropper to break the key and hamper the security. Quantum Cryptography ensures that safety by not allowing the eavesdropper to access the data without hampering the key. However the hardware requirements make it inaccessible. In this paper a novel algorithm of key exchange that involves the best of both quantum and classical worlds has been proposed. It is called Bi-Symmetric Key Exchange. Simulation result and subsequent performance analysis show the efficiency of this algorithm over existing key exchange algorithms with respect to average computational time for key generation.

Keywords Quantum cryptography · Classical cryptography · Photons · Q-bit Bi-symmetric · Security

S. Sonthalia (✉) · T. Mandal · M. Chakraborty
Department of Information Technology, Institute of Engineering and Management, Salt Lake, Kolkata, West Bengal, India
e-mail: sonthalia1996@outlook.com

T. Mandal
e-mail: mandal.trideep1435@gmail.com

M. Chakraborty
e-mail: mohuyacb@iemcal.com

M. Chakraborty et al. (eds.), *Proceedings of International Ethical Hacking Conference 2018*, Advances in Intelligent Systems and Computing 811,
https://doi.org/10.1007/978-981-13-1544-2_8

1 Introduction

Classical cryptology is the science of rendering information unintelligible to unintended parties. Various methods are used to encrypt and decrypt information. Classical cryptology relies on a key, which may be a mathematical function, alphanumeric string or even a matrix which is pre-decided and recorded either on paper or in digital format. Quantum physics has introduced us to qubits, or quantum bits. They lose their state when any attempt to measure their state is made. Their state can be read only once, irrespective of whether intended data can be read or not. This has sparked interest in using this counter-intuitive probabilistic nature to transmit data securely. Existing cryptographic protocols are either strictly classical or purely based upon quantum physics [1, 2]. In this paper we have proposed a new algorithm which makes use of best of both the worlds to provide a better cryptographic key exchange protocol and is called Bi-Symmetric Key Exchange.

The organization of the paper is as follows. After the introduction in Sect. 1, brief overview of existing cryptographic key exchange protocols has been given in Sect. 2. Section 3 holds description, algorithm, simulation results and performance analysis of Bi-Symmetric Key Exchange. Section 4 concludes the paper with some highlights on future works in this area.

2 Overview of Existing Cryptographic Key Exchange Protocols

2.1 BB84 Protocol

In 1984 Charles Bennet and Gilles Brassard developed the most famous quantum key distribution protocol known as BB84 [3, 4]. The system makes use of polarized photons to transmit information. Information is transmitted through fiber-optic cables or free space. According to the design, none of these channels need to be secure. Photons follow Heisenberg's Uncertainty Principle [5] and thus the state of a photon cannot be determined without affecting its state. If an eavesdropper tries to intercept the data transmission, he/she cannot do it without changing the state of the transmitted photons. As a result, presence of an eavesdropper increases error percentage in the data stream. BB84 protocol encodes the information in non-orthogonal states. Table 1 shows the encoding scheme of BB84.

Table 1 BB84 encoding scheme

Basis	0	1
+	↑	→
×	↗	↘

In this protocol, Sender decides upon a basis, where he denotes two of four polarization states of a photon by binary zero and the other two polarization states by binary one, as shown in Table 1. Sender then sends the encoded string to the Receiver over public channel. Receiver having no idea about the basis that Sender has encoded in chooses a convention of his own. Thus, he may detect the photon correctly or incorrectly. In other words, there is a 50% chance that Receiver decodes the information sent by Sender correctly and a 50% chance that he interprets it erroneously. Nevertheless, Receiver continues to interpret the string using his own basis. If an eavesdropper tries to understand the states of the photons, he would have to detect their state, thus changing their original state. This would result in 50% drop in integrity of the transmitted information. Now when Receiver would try to understand the data sent to him, he would have only 50% chance to interpret the data.

Now, Receiver tells Sender about the basis used by him/her over the public channel. If his basis is compatible with the photon generated by Sender, then Receiver responds with "MATCH", else replies "DISCARD". The photons for which both of them have used the same basis is converted to its equivalent binary 0 or 1, as shown in Table 1.

Then, Receiver calculates his match percentage. If the match percentage is above 50%, the generated key is used for encrypting data for further communication. On the other hand, if match percentage is below 50%, it implies the presence of an Eavesdropper in the channel. Sender and Receiver discard the present key and proceed to generate a new key until they have a better match percentage. Figure 1 shows the workflow diagram of BB84.

2.2 Diffie-Hellman-Merkle Protocol

Diffie-Hellman-Merkle key exchange (DHM) [6, 7, 8] is one of the first public-key protocols as designed by Ralph Merkle and named after Whitfield Diffie and Martin Hellman. It is a method of securely exchanging cryptographic symmetric keys over a public channel. Both Sender and Receiver share the same key which is used to encrypt data for future transmissions. This algorithm is based upon discrete logarithm problem solving, which takes way too long to be calculated by the fastest classical computing system [9, 10, 11].

In DHM protocol, Sender and Receiver decide beforehand the multiplicative group of integers modulo c, and d, which is a primitive root modulo c. According to the design, c is a prime number. Values of c and d are chosen such that the shared key lies between 1 and $c - 1$. Now, Sender chooses a secret number, say a. Receiver too chooses a secret number, say b.

Sender sends to Receiver

$$J = d^a mod\ c \tag{1}$$

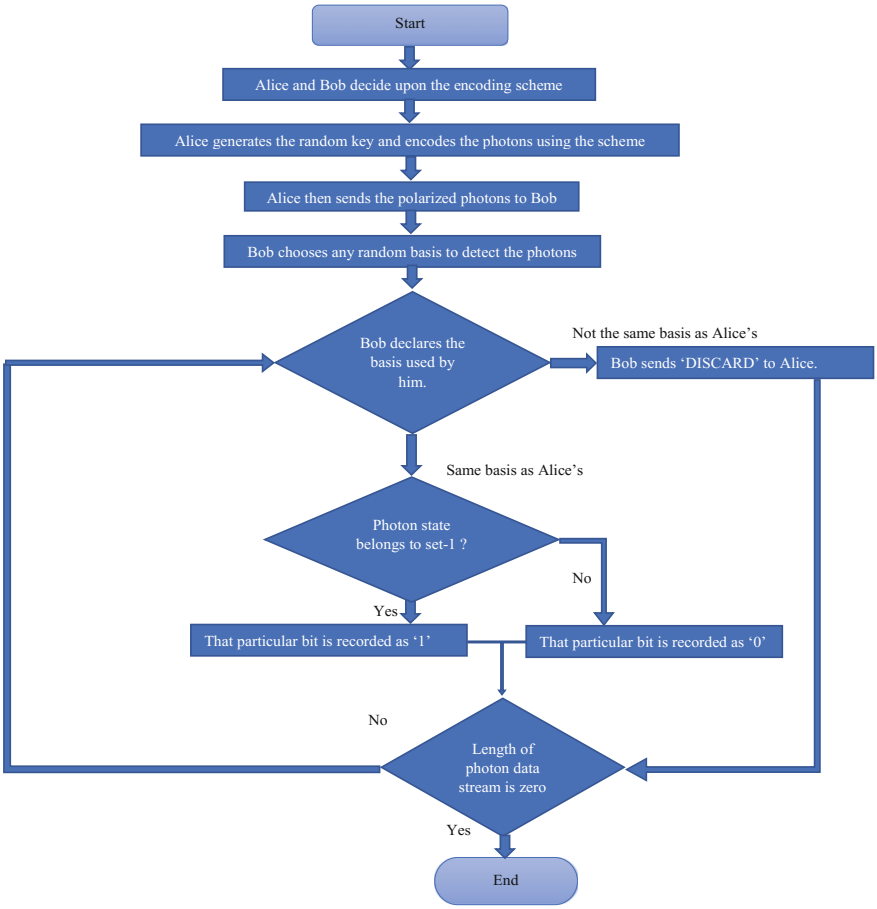

Fig. 1 BB84 flowchart

Receiver sends to Sender

$$K = d^b mod\ c \tag{2}$$

Sender determines

$$S = K^a mod\ c \tag{3}$$

Receiver determines

$$S = K^b mod\ c \tag{4}$$

Both Receiver and Sender now share the same key S, as under mod c,

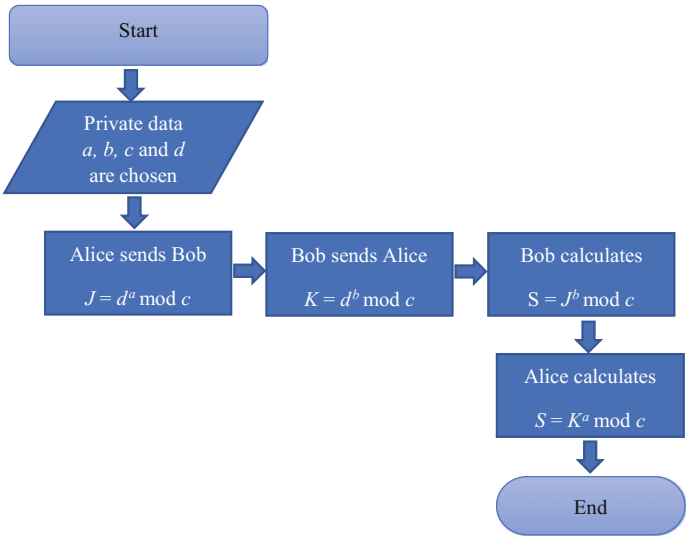

Fig. 2 Diffie-Hellman-Merkle protocol flowchart

$$K^a mod\ c = d^{ab} mod\ c = d^{ab} mod\ c = K^b mod\ c \tag{5}$$

This key 'S' can now be used sending encrypted data over the public channel. The workflow diagram of Diffie-Hellman-Merkle protocol is shown in Fig. 2.

3 Bi-symmetric Key Exchange: Proposed Cryptographic Key Exchange Algorithm

Bi-symmetric Key Exchange protocol improves upon BB84 and Diffie-Hellman-Merkle by using some of their key features along with its own. In this section we have proposed our algorithm for key distribution using concepts from both the protocols. Instead of going in with quantum bits or photons we rely upon classical bits. Tables 2, 3 and 4 demonstrate the algorithm of this protocol.

3.1 Description

The system starts with the creation of a universal set U, which is a set of symbols to be used to encode the private key by generating random strings. For the sake of explanation, we have considered a universal set U, consisting of lowercase and uppercase alphabets and numbers as given in Eq. (6).

Table 2 Algorithm for main function

Steps	Actions
Step 1.	First define the universal set U, private sets A1, B1, A2, B2 and the length L of the string which is to be generated for key transfer
Step 2.	Ask the user to choose his/her role as a Sender or Receiver
Step 3.	If user chooses Sender
Step 4.	Generate the 1st string using Generator() and pass L as parameter
Step 5.	Ask the Sender to send this string to Receiver over public channel
Step 6.	Enter the string sent by the Receiver
Step 7.	Iterate over the characters in the received string. Pass each character along with the sets corresponding to generation of 1st key. If Set_Check() returns '1', print '0' or '1' depending on set with which the function was called. If Set_Check() was called with the set corresponding to '0', print '0' and vice versa. This string of '0' and '1' is the 1st key
Step 8.	Generate the 2nd string using Generator() and pass L as parameter
Step 9.	Enter the string sent by the Receiver
Step 10.	Iterate over the characters in the received string. If the character in the generated and received string belong to the same set (the sets corresponding to generation of 2nd key), store 'M' in an array. Else, store 'D' in the array
Step 11.	Send this array to Receiver
Step 12.	For the characters lying in the same set, print the number corresponding to that set to generate the 2nd key
Step 13.	Else if the user chooses Receiver
Step 14.	Generate the 1st string using Generator() and pass L as parameter
Step 15.	Enter the string sent by Sender
Step 16.	Iterate over the characters in the received string. If the character in the generated and received string belong to the same set (the sets corresponding to generation of 1st key), store 'M' in an array. Else, store 'D' in the array
Step 17.	Send this array to the Sender
Step 18.	For the characters lying in the same set, print the number corresponding to that set to generate the 1st key
Step 19.	Generate the 2nd string using Generator() and pass L as parameter
Step 20.	Enter the string sent by Sender
Step 21.	Iterate over the characters in the received string. Pass each character along with the sets corresponding to generation of 2nd key. If Set_Check() returns '1', print '0' or '1' depending on set with which the function was called. If Set_Check() was called with the set corresponding to '0', print '0' and vice versa. This string of '0' and '1' is the 2nd key
Step 22.	Append both the keys to generate the final key

Table 3 Algorithm for generator function

Steps	Actions
Step 1.	Take length L as an argument
Step 2.	Apply the correct seeding for random generator function
Step 3.	Select a random character from the universal set of characters and store it in an array
Step 4.	Repeat the previous step until the length of array created is not equal to L

Table 4 Algorithm for Set_Check function

Steps	Actions
Step 1.	Accept a character and a character array as parameters
Step 2.	If the character exists in the character array, return '1'
Step 3.	Return '0' as a default case

$$U = \{ABCDEFGHIJKLMNOPQRSTUVWXYZ\,abcdefghijklmnopqrstuvwxyz\,0123456789\} \tag{6}$$

Before the sender and receiver start the exchange of keys, they decide upon four private sets which are subsets of the universal set U. These four sets, by design, would be disjoint in nature and their union would contain fewer elements than the universal set. Let us name these sets as A_1, B_1, A_2 and B_2.

$$\{A_1, A_2, B_1, B_2\} \subset U \tag{7}$$

$$A_1 \cap B_1 = \phi \tag{8}$$

$$A_2 \cap B_2 = \phi \tag{9}$$

The size and symmetry of the above four sets would be limited only by the above stated requirements and the users' needs. These four sets would be used in pairs of two, the first pair, A_1 and B_1 will be used first to generate the first private key (Key 1) and the other pair, A_2 and B_2 will be used to generate the second private key (Key 2). So, at any given instant only one of the two pairs will be in use as the private set for generation of the key (be it Key 1 or Key 2).

These sets are similar to the polarization conventions used in the BB84 protocol. In BB84 protocol, two out of the four photon states signify binary zero and the other two states represent binary one (as shown in Table 1). Similarly, in our system, the sender and the receiver will decide upon which of the above two sets (i.e. A_1 and B_1) would denote to binary zero, making the other denote to binary one. This is done for generating the first key, similarly for the second key, one of the two sets (i.e. A_2 and B_2) would denote to binary zero, making the other denote to binary one.

Let us assume that in our system set A_1 and A_2 represent binary 0, and set B_1 and B_2 represent binary 1. All of this is done privately and is user dependent. Once Sender and Receiver have decided on the private sets, they can start the exchange of keys. Instead of transmitting any photons over free space, both of them will generate

a string of X characters on their systems. This string of X characters will be generated over the Universal Set U, where the characters can be repeated. Let Sender be the sender and Receiver the receiver. Sender then will transmit her string to Receiver. Let *i* be the position of the *i*th element in both Sender's and Receiver's string. Receiver upon receiving the string, checks if the *i*th character in his string belongs to the same set containing Sender's *i*th character. Since they are generating the first key (Key 1) so Receiver will be checking the first pair of sets, A_1 and B_1. If both the *i*th elements belong to the same private set (either A_1 or B_1) then Receiver prints 'M' which stands for "MATCH". If both the characters don't belong to the same set or are not lying in the private set A_1 or B_1, he prints 'D' which stands for "DISCARD". In this way Receiver goes through all the characters of Sender's string and generates his own string of X characters containing 'M' and 'D' which is known as "MATCH-DISCARD" string. This string is sent over to Sender. Now both Sender and Receiver have their own string of X characters and the "MATCH-DISCARD" string which is also of X characters. They then determine the characters in their string which are at the same position as that of 'M' in the "MATCH-DISCARD" string. If their character belongs to A_1, they record it as binary 0, else they record it as binary 1. This string of 0's and 1's forms the first key (Key 1).

Now Sender and Receiver interchange their roles. Sender becomes the receiver and Receiver becomes the sender. They repeat the above process with sets A_2 and B_2 as their private sets. In this way they end up generating the second key (Key 2). The final private key is produced by applying any complex one-way mathematical function between Key 1 and Key 2 (over here we are appending both the keys).

Similarities with BB84 protocol. Bi-symmetric key exchange protocol is inspired from two aspects of BB84 protocol. First the basis and second the Match-Discard verification process. BB84 used four non-orthogonal polarization states of a photon as the smallest package of information and were used to transmit information from Sender to Receiver. Two of these states were mapped to binary 0 and the other two to binary 1, forming the basis. Our system uses a set of 62 characters comprising of lower case alphabets, upper case alphabets and numbers. Each of these 62 characters is the smallest package of information, and a combination of these characters is used to generate strings and private sets. The private sets A1, B1, A2 and B2 function in a way similar to basis in BB84 protocol. The strings are used to exchange data among the Sender and Receiver. This exchanged data is verified using the private sets through a process similar to the Match-Discard verification process of BB84 protocol.

Similarities with Diffie-Hellman-Merkle protocol. In contrast to BB84 protocol, where data transmission is directed from Sender to Receiver, Diffie-Hellman-Merkel protocol allows exchange of data among Sender and Receiver over a public channel to generate a symmetric key. Bi-symmetric key exchange protocol is inspired from this exchange of data among Sender and Receiver, as it allows bi-directional transfer of data to generate two symmetric keys, which are then used to generate the final key.

Table 5 String size and average key size (Key 1 or Key 2) comparison for a private set of 15 elements

Length of string used	100	200	300	400	500	600	700	800	900	1000
Average length of key generated	25	36	53	72	92	117	128	132	155	185

3.2 Bi-symmetric Key Exchange Flowchart

Figure 3 shows the workflow of Bi-Symmetric key exchange algorithm.

3.3 Algorithm of Bi-symmetric Key Exchange Protocol

See Tables 2, 3 and 4.

3.4 Performance Analysis

Figure 4 shows the plot of average computational time for key generation. It indicates that computational time required to generate a key from a string of different lengths, based on pre-specified private sets is more or less constant, with a difference of about 500 μs. In Table 2, we get an idea of the size of generated keys for different lengths of strings used for key generation. Looking at Table 2 and Fig. 1, we can conclude that generation of a key for practical use takes a time of 0.5–1 ms only. This is in contrast with other algorithms where time taken to generate a key rises exponentially with increase in size of keys. This low variance in the time taken for generation of larger keys and non-exponential nature of increase in the time required to generate larger keys gives the user flexibility to generate keys of any length, based on his/her security requirements. They wouldn't have to think twice before increasing the size of their key as the increase in computation time is considerably less when compared to other protocols. The overall time required to generate keys of 150–190 characters is around a millisecond. As a result, Sender and Receiver have the flexibility to generate a new key in real time. This would eliminate the need to wait too long or pause exchange of information to secure the communication with a new key (Table 5).

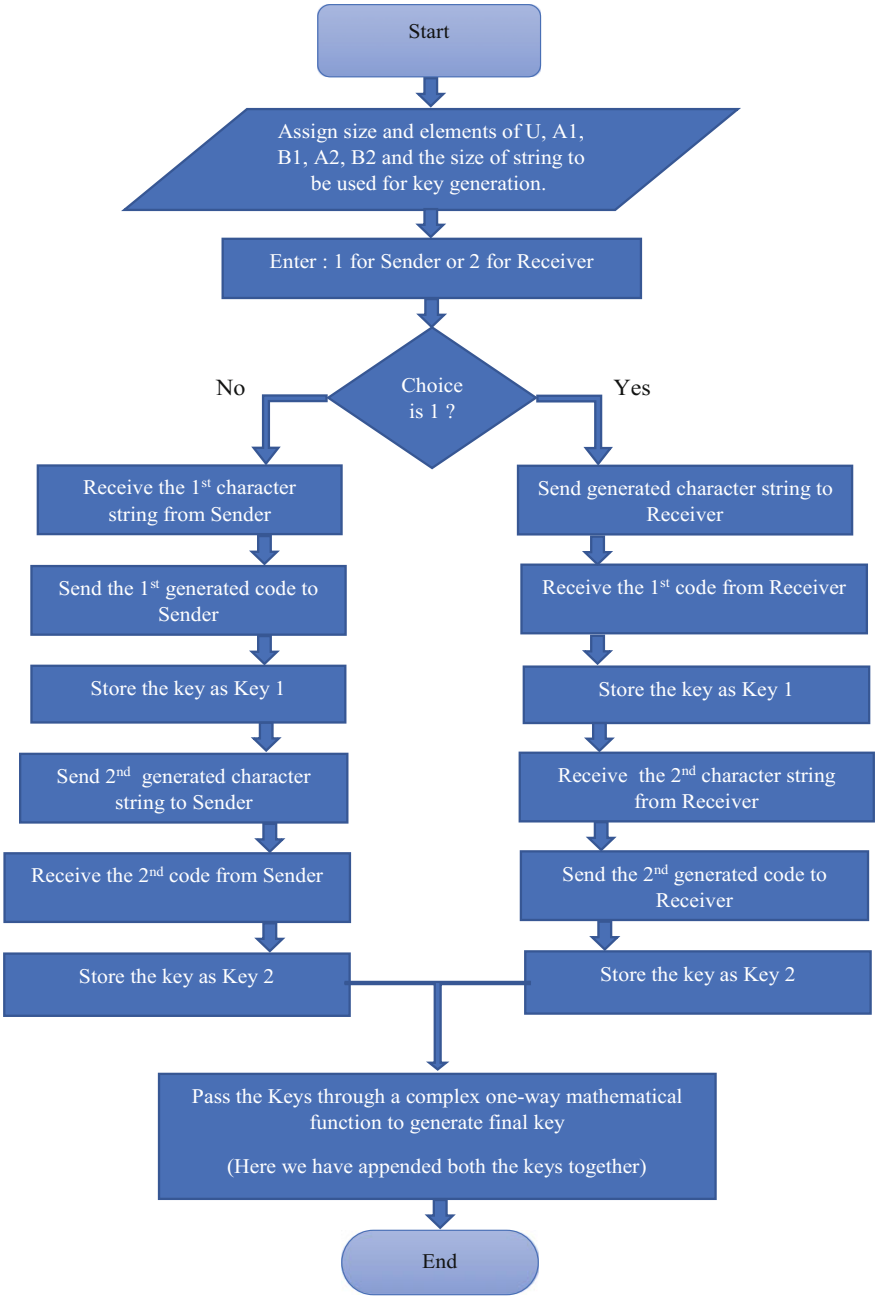

Fig. 3 Bi-symmetric key exchange flowchart

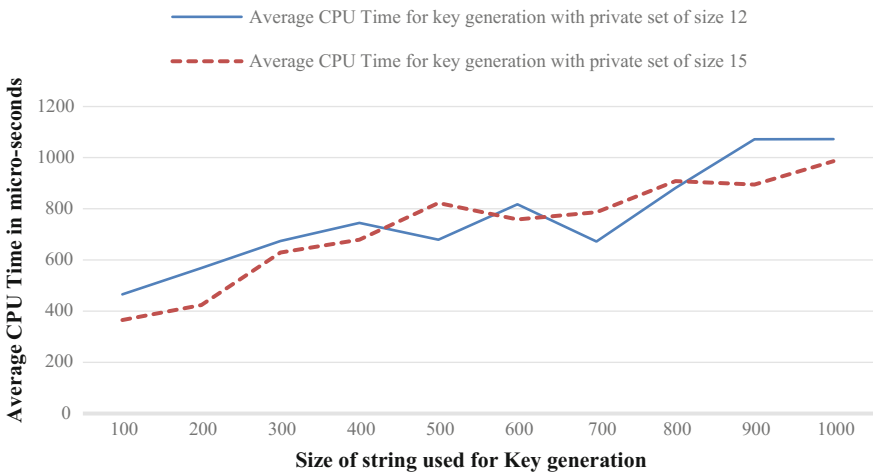

Fig. 4 The plot of average computational time required to generate key on the user's system vs the length of string used to generate the key by the user

4 Conclusion

BB84, in spite of being a robust quantum key exchange protocol has a few major flaws which make it impractical for current use. The sender needs to have access to a quantum laser setup for generating the photons and the receiver needs to detect their polarization, both of which add to hardware cost. Moreover, the transmission of these photons takes place through optical fibers or free space [12]. Whereas Bi-symmetric Key exchange protocol improves upon these flaws by eliminating the need of any dedicated hardware setup for generation and transmission of keys. As a result, this protocol can be used to secure transmissions between mobile and low powered devices too. The time required for generation of larger keys doesn't increase exponentially, which gives the user the freedom to generate key as per his security requirements. Additionally, the user can choose any number system for implementation of this protocol for improved encryption. Here we made use of binary system where the key generated consists of both 0 and 1. Since both the Universal Set U and the private sets are user defined, he/she can choose any number system like quaternary, octal, or decimal etc. This makes our system flexible and less prone to attacks, the required changes for which would be implemented by us based on the user's demands. To sum up, Bi-Symmetric Key Exchange is a capable protocol which takes in better features of existing cryptographic algorithms and produces a powerful yet simple way of securing today's key exchange.

References

1. Singh, H., Gupta, D.L., Singh, A.K.: Quantum key distribution protocols: a review. IOSR J. Comput. Eng. (IOSR-JCE) **16**(2), 01–09 (2014). e-ISSN: 2278–0661 p-ISSN: 2278-8727
2. Asif, A., Hannan, S.: A review on classical and modern encryption techniques. Int. J. Eng. Trends Technol. (IJETT) **12**(4) (2014)
3. Bennett, C.H., Brassard, G.: Quantum cryptography: public key distribution and coin tossing. In: Proceedings of IEEE International Conference on Computers, Systems and Signal Processing, New York, vol. 175, p. 8. (1984)
4. Bennett, C.H., Brassard, G.: Quantum cryptography: public key distribution and coin tossing. Theoretical aspects of quantum cryptography—celebrating 30 years of BB84. Theor. Comput. Sci. **560**(Part 1), 7–11 (04 December 2014). https://doi.org/10.1016/j.tcs.2014.05.025
5. Heisenberg, W.: Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik. Zeitschrift für Physik (in German), **43**(3–4), 172–198 (1927). Bibcode: 1927ZPhy…43..172H, https://doi.org/10.1007/bf01397280. Annotated pre-publication proof sheet of Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik, March 21, 1927
6. Merkle, R.C.: Secure communications over insecure channels. Commun. ACM. **21**(4), 294–299 (1978). https://doi.org/10.1145/359460.359473. Accessed August 1975; Revised September 1977
7. Diffie, W., Hellman, M.: New directions in cryptography (PDF). IEEE Trans. Inf. Theory **22**(6), 644–654 (1976). https://doi.org/10.1109/TIT.1976.1055638
8. Hellman, Martin E.: An overview of public key cryptography (PDF). IEEE Commun. Mag. **40**(5), 42–49 (2002). https://doi.org/10.1109/MCOM.2002.1006971
9. Garzia, F.: Handbook of Communications Security, p. 182. WIT Press, (2013). ISBN: 1845647688
10. Buchmann, J.A.: Introduction to Cryptography, 2nd ed. Springer Science & Business Media, pp. 190–191 (2013). ISBN: 1441990038
11. Barbulescu, R., Gaudry, P., Joux, A., Thomé, E.: A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In: Advances in Cryptology—EUROCRYPT 2014. Proceedings 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques. Lecture Notes in Computer Science, Copenhagen, Denmark, vol. 8441, pp. 1–16 (2014). https://doi.org/10.1007/978-3-642-55220-5_1. ISBN: 978-3-642-55220-5
12. Ojha, V., Sharma, A., Goar, V., Trivedi, P.: Limitations of practical quantum cryptography. Int. J. Comput. Trends Technol. (2011). ISSN: 2231-2803