IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 4 June 2022, accepted 20 June 2022, date of publication 6 July 2022, date of current version 15 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3188798

RESEARCH ARTICLE

A Modified Key Sifting Scheme With Artificial Neural Network Based Key Reconciliation Analysis in Quantum Cryptography

CHITRA BISWAS, MD. MOKAMMEL HAQUE, AND UDAYAN DAS GUPTA
Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Chattogram 4349, Bangladesh
Corresponding author: Md. Mokammel Haque (mokammel@cuet.ac.bd)

This work was supported in part by DRE Project of Chittagong University of Engineering and Technology under Project CUET/DRE/2020-21/CSE/021.

ABSTRACT Quantum Cryptography emerged from the limitations of classical cryptography. It will play a vital role in information security after the availability of expected powerful quantum computers. Still many quantum primitives like quantum money, blind quantum computation, quantum copy protection, etc. are theoretical as they require a completely functional quantum computer for their implementation. But one prominent quantum cryptographic primitive, the Quantum Key Distribution (QKD) is possible with current technology. The QKD is a key establishment system having several stages namely raw key generation, key sifting, key reconciliation, and privacy amplification. In this paper, an efficient key sifting scheme has been developed. Successful simulation has shown that the proposed modified key sifting scheme requires less time to build the sifted key compared to the sifted key in conventional BB84 protocol in most cases. This paper also represents Tree Parity Machine (TPM) based key reconciliation analysis using different learning algorithms such as Hebbian, Anti-Hebbian, and Random-Walk. This reconciliation analysis helps to choose the optimum learning algorithm for Artificial Neural Network (ANN) based key reconciliation in future Quantum Key Distribution systems.

INDEX TERMS Artificial neural network, BB84 protocol, key sifting, key reconciliation, learning algorithms, quantum key distribution, quantum cryptography.

I. INTRODUCTION

Information and security are inextricably intertwined because without security different types of attacks such as hacking, malware invasion, etc. make it easy for the intruders to have unauthorized access to the information [1]. In this situation, cryptology, the science of making and breaking the codes, plays a vital role in the case of information security. A strong cryptographic algorithm can secure all the information while cryptanalysis can create insecurity. So, developing an efficient algorithm against an adversary with unlimited resource power can be a holy grail of cryptography. But, the cryptographic feats were achieved when Gilbert Sandford Vernam first invented one-time pad (OTP) encryption in 1917 [2]. Claude Shanon has proved that a one-time pad (OTP) is secure because the random key is just used only once [3]. Even if OTP is theoretically unbreakable, it can be insecure in the practical situation because of two unachievable phenomena of classical physics. One phenomenon is truly random number generation and another one is the secure key distribution over insecure channels [4]. The symmetric key algorithms like DES and AES need to create a key that is shared between the sender and the receiver. Symmetric key algorithms use short keys for encrypting long messages and as a consequence reduce the utilization of random keys. So, these symmetric algorithms are not as protected as one-time pad (OTP). When the secret key is transmitted through an insecure channel, there is a possibility of copying or stealing by the intruder. This is a huge problem of key distribution in the case of symmetric key algorithms [4].

For finding out the solution to the key distribution problem asymmetric key algorithms or public key algorithms

The associate editor coordinating the review of this manuscript and approving it for publication was Chien-Ming Chen.

were invented. In the renowned asymmetric key algorithm, RSA scheme (named after its inventors, Ron Rivest, Adi Shamir, and Leonard Adleman), the receiver, Bob, creates two keys: one is the public key and another is the private key. The public key is broadcasted by Bob. The sender, Alice encrypts her message with the public key of Bob and sends it through an insecure channel. At the receiving end, Bob can decrypt the message with the corresponding private key [5]. So, public-key cryptography can solve the key distribution problem. Because of providing high protection to financial transactions, military communications, e-mails, medical data, websites, etc., public-key cryptosystems such as RSA [5], Elliptic Curve Cryptography (ECC) [6], Diffie-Hellman (DH) [7] are widely being used in Transport Layer Security (TLS) supported traditional computers and devices [8]. The security of public-key cryptography depends on some mathematical assumptions. As an instance, the security of RSA depends on the difficulty of factoring a large composite integer. The security of RSA algorithm would be compromised if there exists an algorithm that can efficiently factor an arbitrarily large integer. So, the possibility of the invention of a fast factoring algorithm cannot be ignored and if this happens the security of most of the public key cryptographic algorithms would be compromised. Already there exists a factoring algorithm known as the Shor's algorithm which can be executed on a quantum computer. Shor's algorithm [9] requires polynomial time to solve Integer Factorization Problems (IFP) and Discrete Logarithm Problems (DLP) and thus introduces quantum cryptanalysis. This indicates that after two decades when quantum computers will be available on a large scale all the public-key cryptographic algorithms will be collapsed [10].

As quantum computers would be able to break the public key cryptosystem, a key distribution technique is required to resist quantum attacks. So, Quantum Cryptography utilizing the laws of quantum mechanics introduces a successful key distribution system known as Quantum Key Distribution (QKD) [11]. The idea of QKD was first introduced by Stephen Wiesner who gave the concept of information transmission through polarized photons [12]. In 1984, Bennett and Brassard utilized this idea to develop the first QKD protocol known as BB84 protocol by which unconditionally secure shared keys are created between two geographically apart entities [13]. Maintaining the laws of quantum physics, QKD provides a key distribution technique where a cryptographic key is shared between two remote parties with utmost security. Traditional cryptography applies mathematical computational difficulties to restrict an intruder while QKD depends on laws of physics [14]. QKD has a special feature, where the presence of an eavesdropper (Eve) can be easily detected by two communicating parties (Alice and Bob) [15]. The key developed by the QKD technique can be then used in the OTP encryption technique or other encryption techniques to increase the security of information. The QKD-developed key is generated after the successful completion of the stages namely raw key generation, key sifting, key reconciliation, and privacy amplification. According to the QKD stages, in the BB84 protocol, the sender transmits a series of polarized photons (qubits) according to the randomly selected basis over the quantum channel. The receiver measures the qubits according to the random basis and generates a sequence of bits. Then through the public channel, the receiver reports to the sender about what types of bases he used for each of the bit measurements. The sender reports to the receiver through the public channel about which bases were correct. After that, the sender and the receiver delete the measured bits whose bases do not match. Thus using the matched basis bits, the sender and the receiver generate the sifted key [31].

Environmental disturbances, equipment imperfections, and eavesdropping can introduce errors in the sifted key bits [16]. The process of correcting the errors is called error reconciliation or information reconciliation or key reconciliation [17]. Bennett and Brassard introduced the most renowned error reconciliation protocol in QKD which is known as Cascade [18] and it is the refined version of the previous protocol BBBSS [19]. Cascade, which has been the most preferable error reconciliation protocol in most QKD applications, is a highly interactive protocol and involves both parity checks and binary searches in order to correct errors. But in the practical situation, this high interactivity degrades the performance [20]. Buttler *et al.* proposed the Winnow protocol in 2003 which was able to decrease the number of interactions [21]. Winnow also utilizes parity checks similar to Cascade. In the Winnow protocol, a Forward Error Correction (FEC) method based on Hamming Codes is used to replace the binary search of Cascade. Thus Winnow requires less number of interactions to correct the errors compared to Cascade. Recently, an error reconciliation method has been popular and is known as the Low-Density Parity Check (LDPC) code which requires an absolute minimum number of interactions. Gallager first proposed the LDPC code in the 1960s as his doctoral dissertation at MIT [22], [23]. LDPC is a powerful and efficient code and it was realized after 40 years when McKay brought it to light again in 1999 [24]. LDPC codes, a genre of FEC codes, are able to correct all the errors in the transmitted key with the help of only one interaction. This communication efficiency requires large computational complexity compared to Cascade and Winnow protocols. Regarding low computational cost, neural cryptography is a new section of cryptography that utilizes a neural network called Tree Parity Machine (TPM). By synchronizing two TPM networks with identical structures, two users can build a cryptographic key by interchanging the inputs and outputs of these networks where the synaptic weights are retained secret [25]–[27]. Neural cryptography can also be used to correct errors in the quantum key distribution of quantum cryptography. Using such an idea, Niemiec developed an error correction method for quantum cryptography. It is based on an artificial neural network, TPM to correct the errors that occurred during the communication in the quantum channel [28].

In the proposed work, there will be contributions in two sections: one is in the key sifting stage and another one is in the key reconciliation stage. First, a new scheme of key sifting in the quantum key distribution has been developed so that the time required for the proposed scheme can be analyzed in comparison to the traditional key sifting stage in BB84. Secondly, using the sifted key as weights two TPM machines have been created on both the sender and receiver sides. Then synchronization of both the TPM machines has been done using different learning algorithms such as Hebbian, Anti-Hebbian, and Random-Walk. Here, synchronization performance using different learning algorithms has been analyzed. So, the modified key sifting scheme based on basis distribution finds an alternative to the traditional key sifting scheme of BB84 protocol and the key reconciliation analysis helps to choose the optimum learning algorithm on the basis of synchronization performance in the case of artificial neural network based key reconciliation of QKD.

The remaining paper construction is as follows. Section II introduces the preliminaries of QKD to establish a final key. Section III illustrates the BB84 key sifting with an example. Section IV describes the idea of proposed key sifting with example. Section V interprets artificial neural network based key reconciliation. Simulation results have been presented in Section VI. Section VII analyzes the security of TPM based key reconciliation with the help of different learning algorithm based simulation results. Finally, Section VIII concludes the paper.

II. THE QUANTUM KEY DISTRIBUTION PRELIMINARIES

Quantum key distribution is a promising key establishment technology that provides unconditional security. Quantum key distribution either uses Heisenberg's uncertainty principle or violation of Bell's inequalities in entanglement based schemes to detect the presence of an adversary.

According to Heisenberg's uncertainty based protocol, measurement of a quantum state changes it. As a result, an error will be introduced by the eavesdropper into the information transfer along a quantum channel which can be detected by the protocol. In the case of entanglement based protocols, if any intruder measures the entangled quanta, the information comes into existence. So, the eavesdropper tries to put an extra quanta into the protocol, and Bell's inequalities will be violated by this extra quanta. Thus the presence of an eavesdropper will also be identified [29]. The Quantum no-cloning theorem also helps to detect the presence of eavesdropping because according to this theorem an adversary cannot copy quantum information [30].

Fig. 1 illustrates the idea of a QKD process where Alice, the sender, and Bob, the receiver use a quantum channel and a classical channel to create the secret key. The quantum channel can be an optical fiber or direct line of sight free space path and the classical channel can be any conventional network connection, i.e. internet or telephone network. On the transmitting side, using a laser source Alice prepares and transmits single polarized photons known as quantum bits

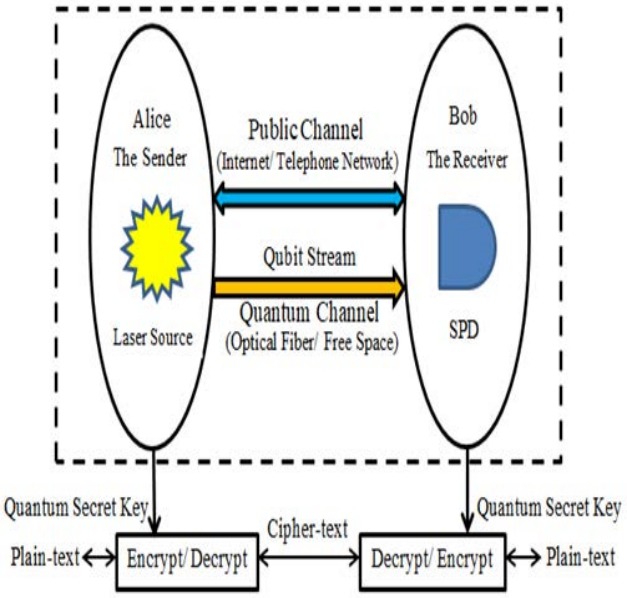


FIGURE 1. The quantum key distribution (QKD) procedure.

or ''qubits'', while on the receiving side Bob measures these photons with single photon detectors (SPD) to generate the raw key bits. Then subsequent information exchange through the classical channel generates the shared secret keys. The generated secret keys from QKD can be utilized as keys to encrypt any information such as data, audio, or video [13].

There are several stages for generating a final key in a QKD protocol have been pictured in Fig. 2:

i. Raw Key Production
ii. Key Sifting
iii. Key Distillation (Error Reconciliation & Privacy Amplification)

i) Raw Key Production: The only quantum section of quantum key distribution in which some quantum states or quantum information are passed through a quantum channel from Alice to Bob with or without the presence of the adversary, Eve [31]. All of the subsequent exchanges known as 'classical post processing' will be held only through a classical channel.

ii) Key Sifting: Alice and Bob determine which measurements will be taken to lead to the secret key and this is done through the classical channel. The protocol which will be used sets the rules for this decision making. The measurements which do not match are discarded by Alice and Bob in a process known as key sifting.

iii) Key Distillation: Bennett *et al.* determined the further processing steps after the key sifting stage for lossy practical channels so that the protocols can even work with transmission errors [19]. Bennett *et al.* [32] also showed how information losses can be repaired from a defective private channel by utilizing an authenticated public channel. As a consequence, the key distillation phase of the classical post-processing requires two steps: error reconciliation and privacy amplification. After these two steps authentication is done finally to elude man-in-the-middle attacks.

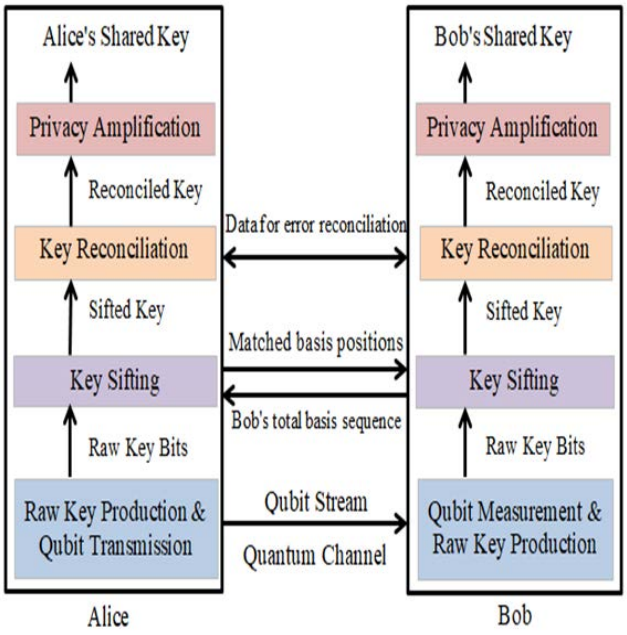


FIGURE 2. Quantum key distribution phases for generating final key.

(a) Error Reconciliation: The quantum bit error rate (QBER) is determined by a classical error correction protocol. QBER calculates the actual error rate after transmission. Either the presence of an eavesdropper or the noise on the quantum channel can create the error. But it is assumed that all errors occur because of eavesdropping. If the value of QBER is below a certain threshold value, then the secret key is forwarded to the upcoming step of key distillation. If QBER is above a certain threshold value, then the secret key is discarded because a high QBER confirms the presence of an eavesdropper. As a result, a new session of QKD is started again.

(b) Privacy Amplification: Privacy amplification is done to thwart the knowledge that Eve has gained on the raw key. In privacy amplification, the key material is compressed by an appropriate factor, depending on the QBER. If the QBER is high, more compression is needed to remove the number of key bits that Eve has acquired. Depending on the two-universal hash functions [33], [34], there are privacy amplification processes that make the key unconditionally secure.

The length of the useable key is reduced with the continuation of the various stages of the QKD protocol.

There are many QKD protocols that are based on two schemes: prepare-and-measure based QKD protocols and entangle-based protocols [35]. In prepare-and-measure based QKD protocols, the sender (Alice) prepares information in the form of polarized photons and the receiver measures the sent photons. Prepare-and-measure based QKD follows Heisenberg's Uncertainty Principle and Quantum No-Cloning theorem. These principles help to detect the

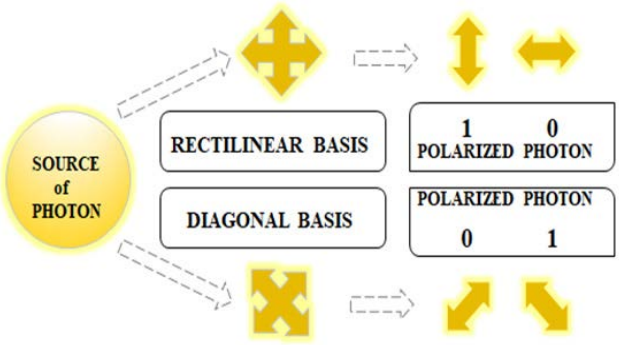


FIGURE 3. Basis and polarized states used for BB84 protocol.

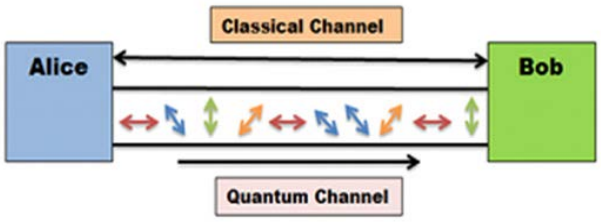


FIGURE 4. Channels used in BB84 protocol.

presence of an eavesdropper by calculating the error that occurs during the transmission from Alice to Bob. Some of the prepare and measure based QKD protocols include BB84 [36], B92 [37], SSP [38], SARG04 [39], S13 [40]. Entangle-based protocols utilize the entangled photons principle for sharing the key between Alice and Bob as well as to detect the presence of an eavesdropper [41]. The quantum entanglement based QKD protocols are E91 [42], BBM92 [43], DPS [44], COW [45].

III. KEY SIFTING IN BB84 PROTOCOL

The first quantum key distribution protocol is BB84 and the idea of BB84 was established by Bennett and Brassard in 1984 [36]. But in 1991 it was practically implemented by using two bases to polarize the single photons according to the random bit sequence. This protocol makes use of four polarization states created by two bases: one basis is a rectilinear basis and another one is a diagonal basis. The rectilinear basis encodes logic 0 as a 0° polarized photon and logic 1 as a 90° polarized photon. According to the diagonal basis logic 0 is represented by a 45° polarized photon and logic 1 as a 135° polarized photon. Fig. 3 represents a pictorial demonstration of the bases and the polarization states according to the bases.

The polarized photons are sent through the quantum channel from Alice to Bob with or without the presence of an eavesdropper. The quantum channel can be either optical fiber or free space. Then all the subsequent exchanges known as 'classical post processing' are carried out through the authenticated secure two-way classical

channel. Fig. 4 illustrates the channels used in the BB84 protocol. Classical post-processing includes key sifting, and key distillation (error reconciliation, privacy amplification, authentication).

The key sifting in BB84 protocol has been described with the help of the following steps for an example.

Quantum Channel Exchange:

1. Alice generates a sequence of random bits.

Bit Position	1	2	3	4	5	6	7	8	9	10
Random Bits	1	0	1	1	0	1	0	1	1	0

2. Alice selects a sequence of random bases to create polarized photons i.e. qubits and sends the qubits to Bob.

Bit Position	1	2	3	4	5	6	7	8	9	10
Alice's Random Bits	1	0	1	1	0	1	0	1	1	0
Alice's Random Bases	R	D	R	D	D	D	R	D	D	R
Qubits	↕	↗↙	↕	↗↙	↗↙	↗↙	↔	↗↙	↗↙	↔

3. Bob chooses a sequence of random bases to measure the polarized photons sent by Alice and generates a sequence of random bits from these polarized photons according to his random bases.

Bit Position	1	2	3	4	5	6	7	8	9	10
Captured Qubits	↕	↗↙	↕	↗↙	↗↙	↗↙	↔	↗↙	↗↙	↔
Bob's Random Bases	D	D	R	D	R	R	D	R	D	R
Bob's Bases matches Alice's Bases		√	√	√					√	√
Bob's measured received bits	0	0	1	1	1	0	1	0	1	0

Public Channel Exchange:

4. Bob sends his entire sequence of random bases to Alice.
5. Alice collects Bob's sequence of random bases and compares it with her random bases. Alice generates the sifted key with the matched basis position bits.

Bit Position	1	2	3	4	5	6	7	8	9	10
Alice's Random Bits	1	0	1	1	0	1	0	1	1	0
Alice's Random Bases	R	D	R	D	D	D	R	D	D	R
Bob's Random Bases	D	D	R	D	R	R	D	R	D	R
Alice's Sifted Key		0	1	1					1	0

6. Alice reports to Bob about the matched bases with positions.
7. Bob collects the positions that hold the matched bases and from these position bits, he generates the sifted key.

Bit Position	1	2	3	4	5	6	7	8	9	10
Bob's Received Bits	0	0	1	1	1	0	1	0	1	0
Bob's Random Bases	D	D	R	D	R	R	D	R	D	R
Alice's sent matched bases with positions		D_2	R_3	D_4					D_9	R_{10}
Bob's Sifted Key		0	1	1					1	0

IV. PROPOSED MODIFIED KEY SIFTING SCHEME

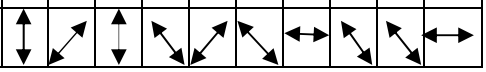
In the proposed modified key sifting scheme, the quantum channel exchange will be carried out similarly to the conventional BB84 protocol, but the modification has been done in the public channel exchange. In the public channel exchange of the proposed modified key sifting scheme, basis comparison and most of the actions on both the sender and receiver sides are carried out simultaneously. As a consequence, this proposed key sifting scheme requires less time than the conventional BB84 key sifting in most cases. The following sequential steps with examples describe the proposed modified key sifting scheme.

Quantum Channel Exchange:

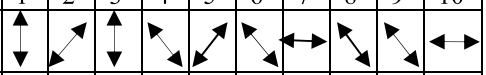
1. Alice generates a sequence of random bits.

Bit Position	1	2	3	4	5	6	7	8	9	10
Random Bits	1	0	1	1	0	1	0	1	1	0

2. Alice selects a sequence of random bases to create polarized photons i.e. qubits and sends the qubits to Bob.

Bit Position	1	2	3	4	5	6	7	8	9	10
Alice's Random Bits	1	0	1	1	0	1	0	1	1	0
Alice's Random Bases	R	D	R	D	D	D	R	D	D	R
Qubits										

3. Bob chooses a sequence of random bases to measure the polarized photons sent by Alice and generates a sequence of random bits from these measured polarized photons according to his random bases.

Bit Position	1	2	3	4	5	6	7	8	9	10
Captured Qubits										
Bob's Random Bases	D	D	R	D	R	R	D	R	D	R
Bob's Bases matches Alice's Bases		√	√	√					√	√
Bob's measured received bits	0	0	1	1	1	0	1	0	1	0

Public Channel Exchange:

4. a. Alice marks only the rectilinear bases in her basis sequence and sends these bases with positions to Bob.

Bit Position	1	2	3	4	5	6	7	8	9	10
Alice's Random Basis	R	D	R	D	D	D	R	D	D	R

b. Bob marks only the diagonal bases in his basis sequence and sends these bases with positions to Alice simultaneously.

Bit Position	1	2	3	4	5	6	7	8	9	10
Bob's Random Basis	D	D	R	D	R	R	D	R	D	R

5. a. Alice only compares Bob's sent diagonal basis with the corresponding basis in her basis sequence and takes only those bit positions at which both the bases are diagonal.

Bit Position	1	2	3	4	5	6	7	8	9	10
Alice's Random Basis	R	D	R	D	D	D	R	D	D	R
Bob's Diagonal Basis	D_1	D_2		D_4			D_7		D_9	

b. Simultaneously Bob only compares Alice's sent rectilinear basis with the corresponding basis in his basis sequence and takes only those bit positions at which both the bases are rectilinear.

Bit Position	1	2	3	4	5	6	7	8	9	10
Bob's Random Basis	D	D	R	D	R	R	D	R	D	R
Alice's Rectilinear Basis	R_1		R_3				R_7			R_{10}

6. a. Alice sends her matched diagonal basis with positions to Bob.
b. Bob sends his matched rectilinear basis with positions to Alice simultaneously.

Bit Position	1	2	3	4	5	6	7	8	9	10
Alice's Raw Key	1	0	1	1	0	1	0	1	1	0
Alice's matched Diagonal Basis with positions		D_2		D_4					D_9	
Bob's matched Rectilinear Basis with positions			R_3							R_{10}
Alice's Sifted Key		0	1	1					1	0

7. a. Alice constructs the sifted key from her matched diagonal basis position bits and Bob's matched rectilinear basis position bits.
b. Bob simultaneously constructs the sifted key from his matched rectilinear basis position bits and Alice's matched diagonal basis position bits.

Bit Position	1	2	3	4	5	6	7	8	9	10
Bob's Raw Key	0	0	1	1	1	0	1	0	1	0
Bob's matched Rectilinear Basis with Positions			R_3							R_{10}
Alice's matched Diagonal Basis with Positions		D_2		D_4					D_9	
Bob's Sifted Key		0	1	1					1	0

From the above discussions, it is clear that, during the public exchange of the proposed modified key sifting scheme, Alice and Bob simultaneously perform almost all the actions.

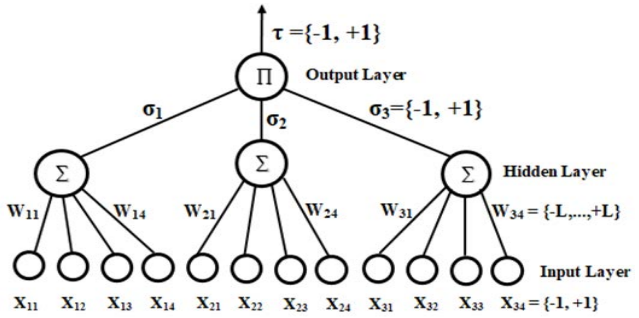


FIGURE 5. General structure of a Tree Parity Machine with K=3 and N=4.

More importantly, in the proposed scheme Alice or Bob has to compare only one type of basis (either diagonal or rectilinear) simultaneously. As a consequence, these simultaneous actions and one type of basis comparison of the proposed scheme help to decrease the sifted key generation time compared to the conventional BB84 key sifting time.

V. ARTIFICIAL NEURAL NETWORK BASED KEY RECONCILIATION

In the key reconciliation stage, a popular artificial neural network called a tree parity machine (TPM) will be used for the purpose of error correction. The tree parity machines used by the sender, receiver, and attackers are the multi-layer feed-forward networks. It contains K hidden units. Each hidden unit has N input neurons. This network has only one output neuron [28]. Fig. 5 illustrates a tree parity machine (TPM) with K=3, N=4.

The input values are binary,

$$x_{i,j} \in \{-1, +1\}$$

and the weights defining the connections between the input layer and the hidden layer are the integers from the range [−L, +L].

$$w_{i,j} \in \{-L, -L+1, \ldots\ldots, +L\}$$

Here, $i = 1, \ldots..,$ K denotes the ith hidden unit of the tree parity machine, and $j = 1, \ldots\ldots.,$ N is the number of inputs into each neuron in the hidden layer.

The output value of ith neuron in the hidden layer is calculated by the value of input x and weight w,

$$\sigma_i = \text{sgn}\left(\sum_{j=1}^{N} x_{i,j} \cdot w_{i,j}\right)$$

where the signum function is:

$$\text{sgn}(z) = \begin{cases} -1, & if\ z \leq 0 \\ 1, & if\ z > 0 \end{cases}$$

The output value, τ of the tree parity machine determined by the product of the output values of the hidden layer neurons is as follows:

$$\tau = \prod_{i=1}^{K} \sigma_i$$

The sender, Alice, and the receiver, Bob use two above mentioned TPM which can be synchronized after mutual learning [25]. To synchronize the TPMs, users produce the same random input, and the output from each TPM is calculated. If the output of the sender's TPM is identical to the receiver's TPM, they initiate the learning process for the TPMs. If the outputs are not the same, then they have to produce another input [28]. So, if outputs of both the tree parity machines are equal then Hebbian, Anti-Hebbian and Random-Walk are the three suitable learning algorithms that are used for weight synchronization in tree parity machine based neural networks. These learning algorithms are suitable because these three learning algorithms are applied in the three special cases of output value utilization of the tree parity machine [46]. The three learning algorithms along with the cases for synchronization have been described in the following:

Hebbian Learning Rule: When both the networks of sender and receiver are trained according to the output value then the Hebbian learning rule is used.

$$w_{i,j}^{+} = g(w_{i,j} + x_{i,j}\tau\theta(\sigma_i\tau)\theta(\tau^A\tau^B))$$

Anti-Hebbian Learning Rule: In the case of the anti-Hebbian learning rule, the networks update weight with the opposite of the output.

$$w_{i,j}^{+} = g(w_{i,j} - x_{i,j}\tau\theta(\sigma_i\tau)\theta(\tau^A\tau^B))$$

Random-Walk Learning Rule: If the synchronization of the participating neural networks is not dependent on output as the output is identical for all participating neural networks, then the random-walk learning rule can be used.

$$w_{i,j}^{+} = g(w_{i,j} + x_{i,j}\theta(\sigma_i\tau)\theta(\tau^A\tau^B))$$

where,

$$\theta(\sigma_i\tau) = \begin{cases} 0, & if\ \sigma_i \neq \tau \\ 1, & if\ \sigma_i = \tau \end{cases}$$

The learning algorithms must ensure that the weights have to be in between −L and +L. If any weight is out of the range, then it is changed and set to the closest value of ±L. In each learning algorithm, the function g(z) keeps the weights in the range [−L, +L].

$$g(z) = \begin{cases} -L, & if\ z \leq -L \\ z, & if\ -L < z < L \\ L, & if\ z \geq L \end{cases}$$

Thus, the synchronization process will be continued until all the weights of the TPMs of both the sender and the receiver are identical i. e. $w_i^A = w_i^B$ [46].

After the transmission of the qubits, the key sifting process is accomplished and then QBER is determined. If the QBER is below a certain threshold value, then key reconciliation is carried out to resolve the errors that are introduced during the transmission of qubits. Using the above concept of TPM synchronization, a secure key reconciliation process in quantum

TABLE 1. The values of K, N and L for different key lengths.

Key Length	K	N	L
128	4	8	7
192	6	8	7
256	8	8	7

cryptography can be performed. The key reconciliation using TPM is performed by the following steps:

1. The sender, Alice, and the receiver, Bob use their own TPMs where they convert their sifted key bits into weights in the range [−L, L]. The weights become the connection between the input layer neurons and hidden layer neurons. Alice and Bob use the identical TPM by choosing the same number of hidden layer neurons, K, and the input neurons for each hidden layer neuron, N. Thus the value of K and N becomes public. Since both the TPMs have the identical structure and the weights are got from the sifted key, a few errors in weights are found keeping most of the weights same. These errors can be resolved by the synchronization of the TPMs.
2. For synchronization, at first Alice produces a random input string of KN length and sends this string to Bob.
3. Both Alice and Bob generate their TPM output value. Alice let to know Bob about her TPM output value.
4. If the output values are the same then the synchronization process starts using one of the three above mentioned learning algorithms. If the output values are not the same then Alice has to generate another input string.
5. After a proper number of iterations, synchronization of both the TPMs is achieved and all the weights of both the TPMs become the same.
6. Finally the weights in the range [−L, L] are converted back to the string of bits. Since both the TPMs are synchronized, all the weights of both the TPMs are the same. As a result, Alice's string of bits and Bob's string of bits become identical. Thus errors can be resolved and the key reconciliation process in quantum cryptography can be accomplished. The reconciled key can be used as a final cryptographic key.

VI. SIMULATION RESULT

The GUI (Graphical User Interface) of the Python Implementation of Key Sifting and Reconciliation has been developed where there are three tabs: Process, KS_Analysis, and KR_Analysis. In the process tab, the required parameters for key sifting and key reconciliation have been provided for the purpose of proper simulation results. For example, the developed GUI for 128 bit 10,000 keys has been pictured in Fig. 6 as follows:

The values of K, N, and L for different key lengths are given below in Table-1.

A. KEY SIFTING SIMULATION RESULTS

For 128-bit key generation, from the GUI in Fig. 7, It is apparent that a total of 40,000 cases i.e. qubit (different qubit

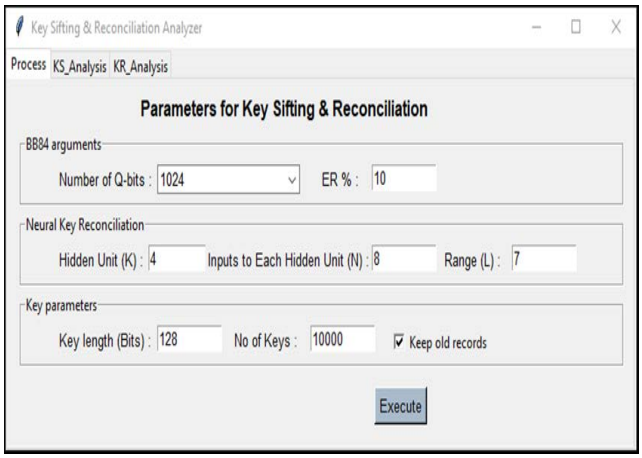


FIGURE 6. Developed GUI dedicated for 128 bit 10,000 keys.

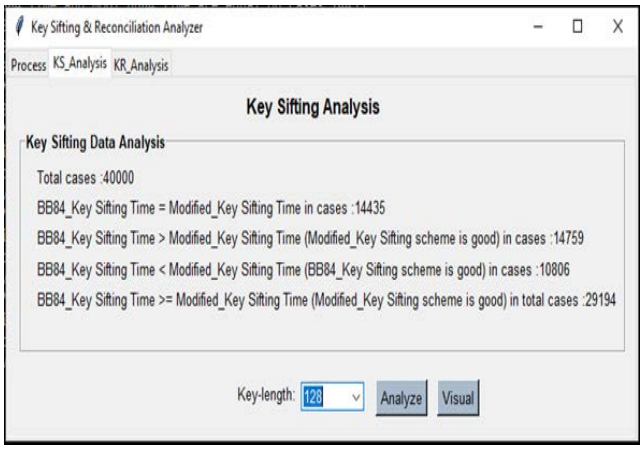


FIGURE 7. GUI for comparative analysis between BB84 and proposed modified key sifting scheme in light of sifted key generation time for 128 bit 40,000 keys.

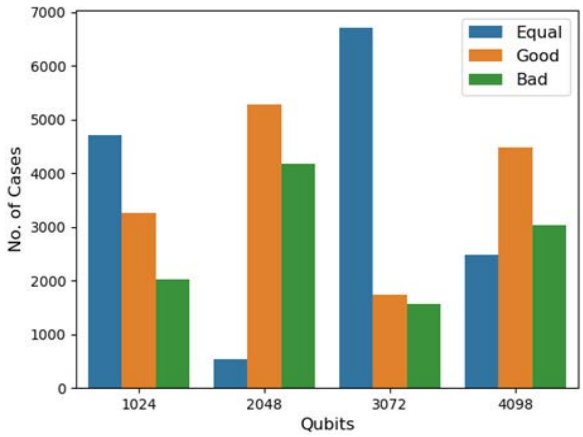


FIGURE 8. For 128 bit 40,000 keys, qubit length-wise performance analysis of modified key sifting scheme with respect to BB84 key sifting scheme.

lengths: 1024, 2048, 3072, 4098) sets have been taken from which raw key is generated on both sender and receiver sides. Each raw key set is manipulated according to the BB84 key sifting scheme and the proposed modified key sifting scheme. According to the GUI, it is seen that in 14,435 cases the time

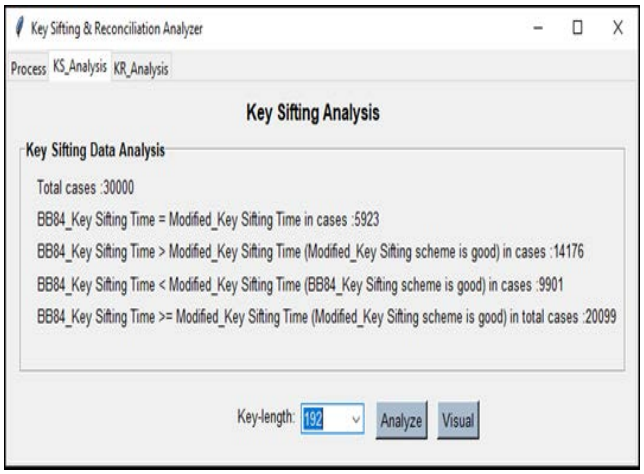


FIGURE 9. The GUI for key sifting data analysis between BB84 and modified key sifting scheme in case of 192 bit 30,000 keys.

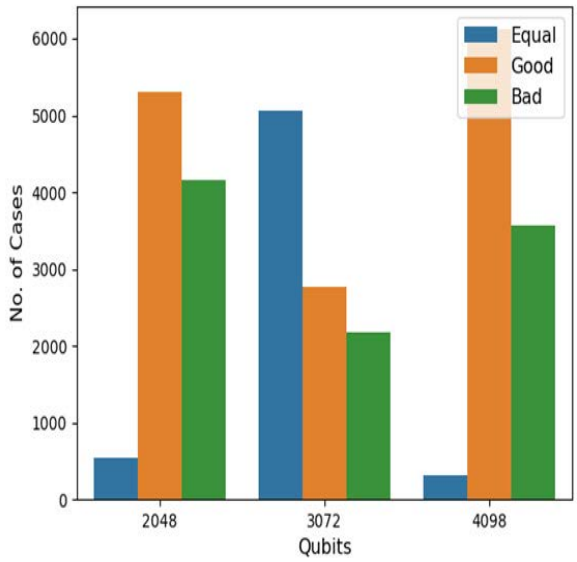


FIGURE 10. Qubit length-wise key sifting performance analysis of modified key sifting scheme with respect to BB84 key sifting scheme in case of 192 bit 30,000 keys.

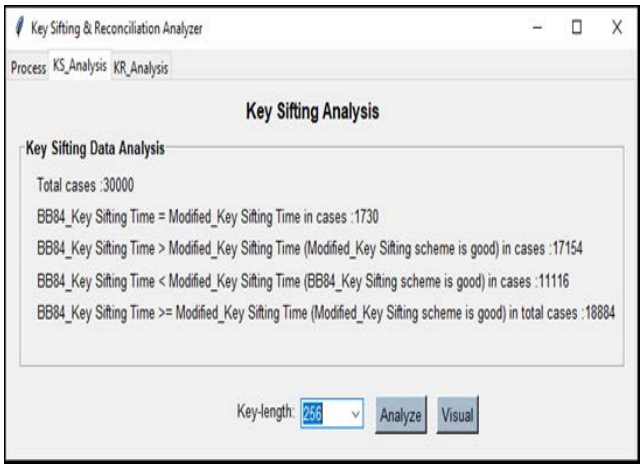


FIGURE 11. The GUI development for comparative key sifting data analysis between BB84 and modified key sifting scheme for 256 bit 30,000 keys.

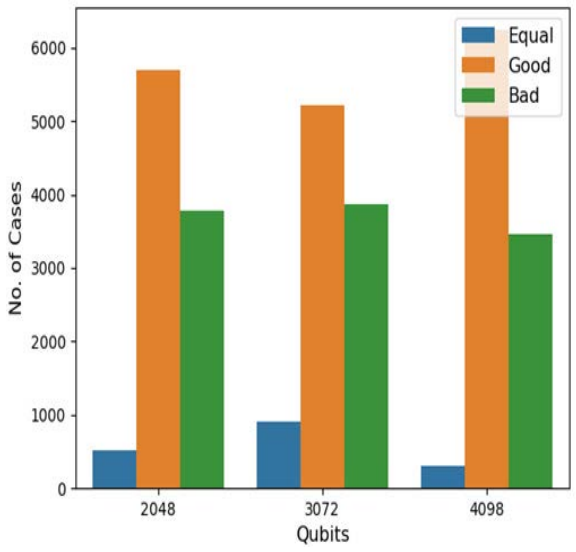


FIGURE 12. Qubit length-wise modified key sifting performance analysis with respect to BB84 key sifting scheme for 256 bit 30,000 keys.

required to generate the sifted key in the BB84 scheme is equal to the time required to generate the sifted key in the proposed modified scheme. There are 14,759 cases in which BB84 key sifting time is greater than the proposed modified key sifting time. As a consequence, it can be said that in these 14,759 cases modified key sifting scheme is better than the BB84 key sifting scheme. Then, the time required to generate the sifted key according to the BB84 key sifting scheme is less than the modified key sifting scheme in 10,806 cases which clearly states that in these 10,806 cases BB84 key sifting scheme is better than the modified key sifting scheme. So, in light of the above analysis, it can be claimed that there is a total of 29,194 cases out of 40,000 cases in which the time required to generate the sifted key according to the proposed modified scheme is less than or equal to the time required to generate the sifted key according to the BB84 scheme. So, the modified key sifting scheme performance concerning sifted key generation time is better than the BB84 key sifting scheme. Fig. 8 shows the qubit length-wise performance analysis of the modified key sifting scheme with respect to the BB84 key sifting scheme for 128-bit 40,000 keys.

Similarly, for 192-bit key generation, a total of 30,000 cases i.e. qubit (different qubit lengths: 2048, 3072, 4098) sets have been taken. Here, for the 192-bit key generation, 1024 qubit length has not been taken because in the key sifting stage after the random basis matching and sifting, the remaining bits are insufficient for building the sifted key. The GUI for key sifting data analysis and key sifting performance analysis between BB84 and modified key sifting scheme in the case of 192-bit 30,000 keys have been illustrated in Fig. 9 and Fig. 10 respectively. Here it is clearly visible that the modified key sifting scheme performance concerning

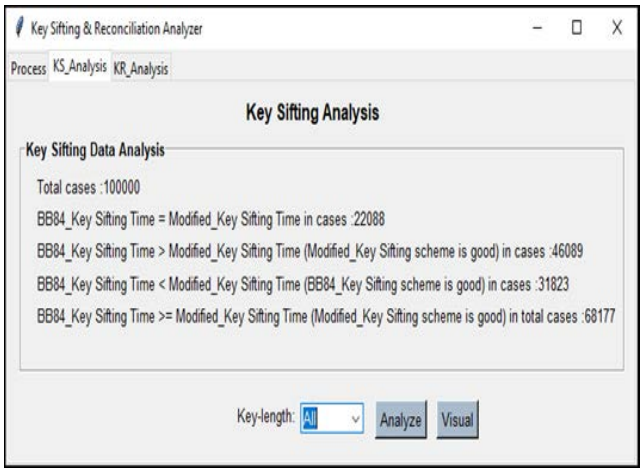


FIGURE 13. The GUI for key sifting data analysis between BB84 and modified key sifting scheme in case of 1,00,000 keys of all key lengths(128, 192, 256 bit).

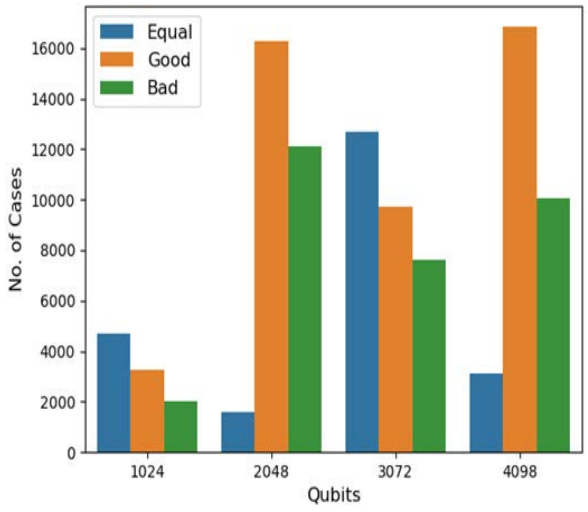


FIGURE 14. Qubit length-wise performance analysis of modified key sifting scheme with respect to BB84 key sifting scheme in case of 1,00,000 keys of all key lengths(128, 192, 256 bit).

sifted key generation time is better than the BB84 key sifting scheme for 192-bit 30,000 keys.

In the same manner, for 256-bit 30,000 keys, a total of 30,000 cases i.e. qubit (different qubit lengths: 2048, 3072, 4098) sets have been taken. From these case analyses, the GUI for key sifting data analysis and qubit length-wise performance analysis in the case of 256-bit 30,000 keys have been illustrated in Fig. 11 and Fig. 12 respectively. Both the figures illustrate that in the case of 256-bit 30,000 keys modified key sifting scheme performs better than the BB84 key sifting scheme regarding key sifting time.

So, if we aggregate all the cases for different key lengths, then the GUI for key sifting data analysis and qubit length-wise performance analysis in the case of 1,00,000 keys are given in Fig. 13 and Fig. 14 respectively.

So, from the overall key sifting scheme performance analysis it can be summarized that concerning the Qubit length the proposed modified key sifting scheme shows

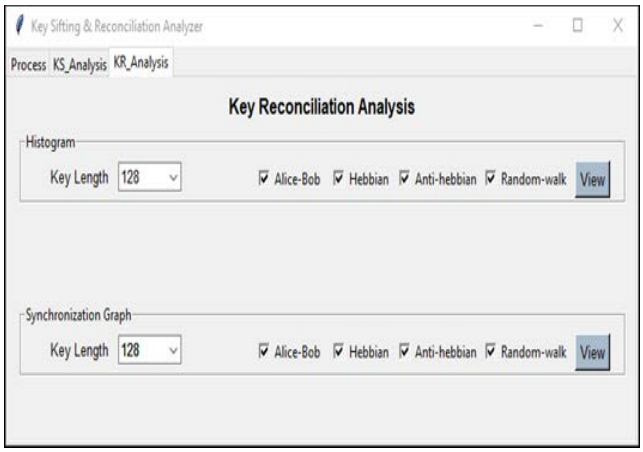


FIGURE 15. The GUI for histogram and synchronization graph according different learning algorithms for 128 bit keys.

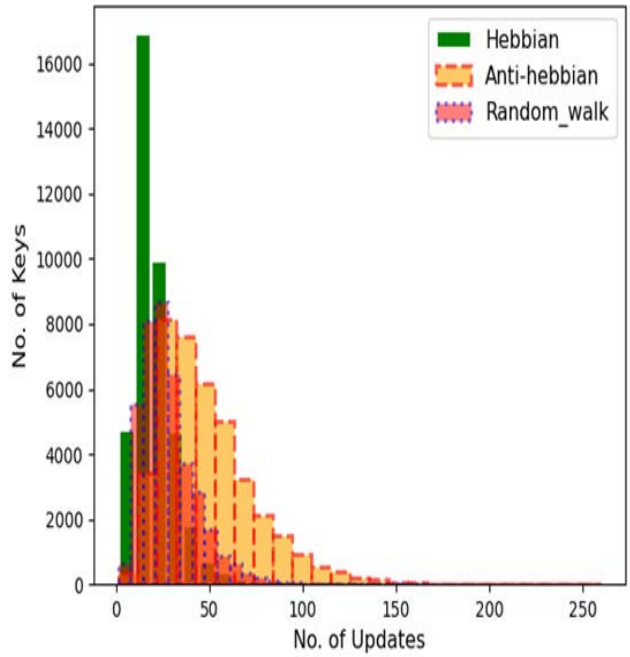
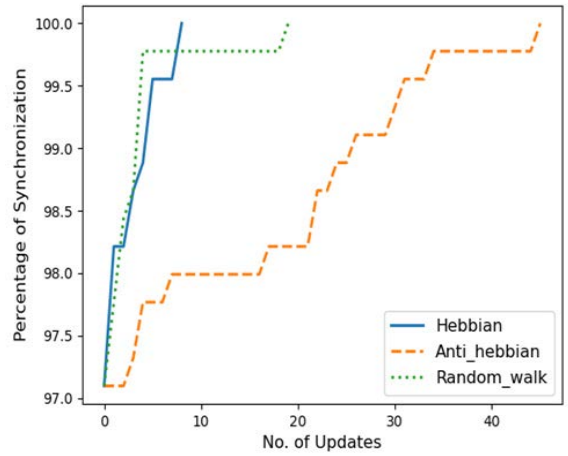
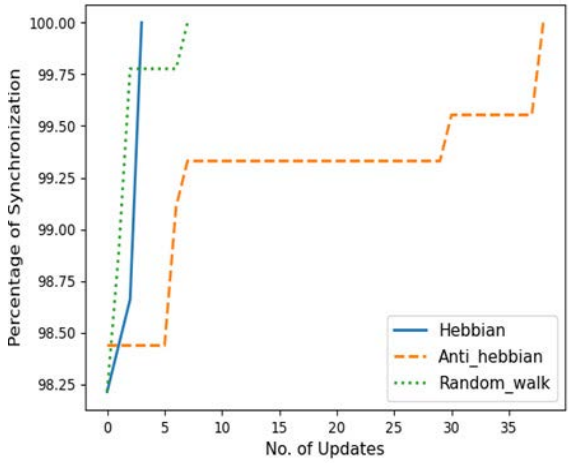


FIGURE 16. Histogram of no. of updates vs. no. of keys for 128 bit 40,000 keys according to different learning algorithms.

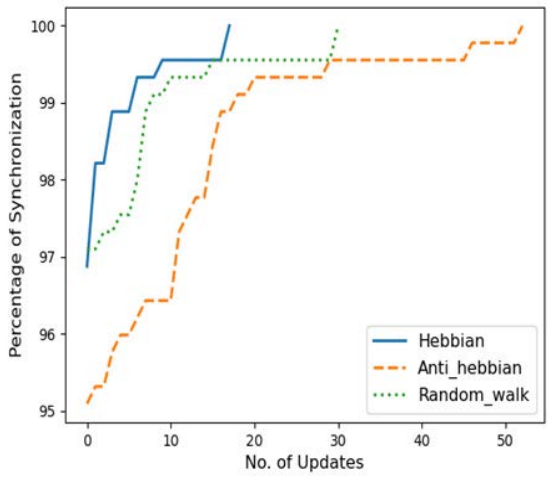
good performance in the number of cases is greater than the cases of the BB84 key sifting scheme. In some cases, both the schemes perform the key sifting operations in equal time. As a consequence, the modified key sifting scheme performance concerning sifted key generation time is better than the BB84 key sifting scheme. Moreover, the modified key sifting scheme needs less basis comparison and hence less memory since this scheme utilizes the idea of only one kind of basis (diagonal or rectilinear) comparison in each of the sender and receiver sides. More importantly, most of the actions like basis transmission and basis comparison on the sender and receiver sides according to the modified key sifting scheme are carried out in parallel which saves time. At last, regarding these advantageous aspects, it can be



(a) Synchronization graph for 128 bit Random Key 1



(b) Synchronization graph for 128 bit Random Key 2



(c) Synchronization graph for 128 bit Random Key 3

FIGURE 17. Updates vs. percentage of synchronization graph for 3 random 128 bit keys according to different learning algorithms.

concluded that the proposed modified key sifting scheme can be preferably used as an alternative to the BB84 key sifting scheme.

TABLE 2. Average No. of Updates and Synchronization time according to different learning algorithms for 128 bit 40,000 keys.

Learning Algorithms	Average No. of Updates	Average Synchronization Time (s)
Hebbian	20.4612	0.03593947442173958
Anti-Hebbian	48.719575	0.08920939985513687
Random Walk	28.090625	0.05016005570292473

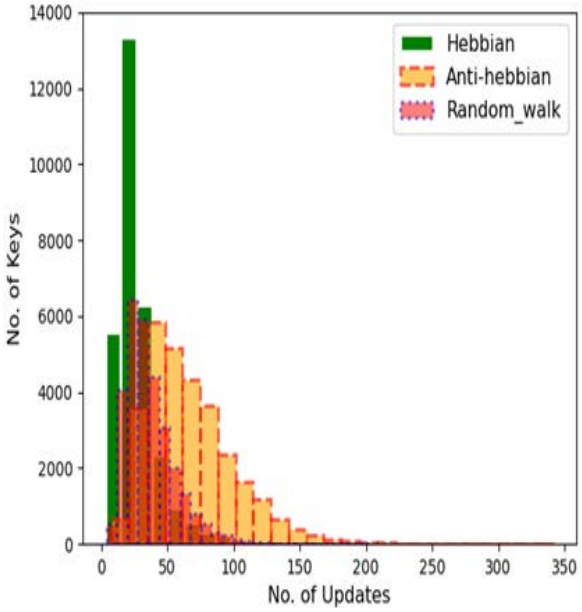


FIGURE 18. Histogram for no. of updates vs. no. of keys in case of 192 bit 30,000 keys according to different learning algorithms.

TABLE 3. Average No. of Updates and Synchronization time for 192 bit 30,000 keys according to different learning algorithms.

Learning Algorithms	Average No. of Updates	Average Synchronization Time (s)
Hebbian	27.1986	0.0768769008477529
Anti-Hebbian	67.29136666666666	0.20278750456968944
Random Walk	36.851166666666664	0.10174079774220784

TABLE 4. Average no. of updates and synchronization time generated for 256 bit 30,000 keys with respect to different learning algorithms.

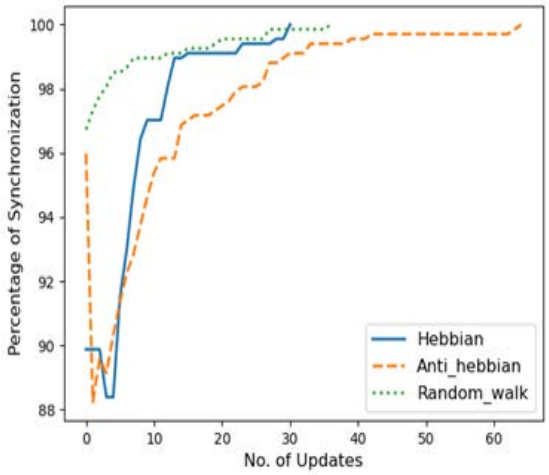
Learning Algorithms	Average No. of Updates	Average Synchronization Time (s)
Hebbian	34.857033333333334	0.1333453301111857
Anti-Hebbian	86.6337	0.3399557679653168
Random Walk	45.995066666666666	0.15958568471272785

B. KEY RECONCILIATION SIMULATION RESULTS

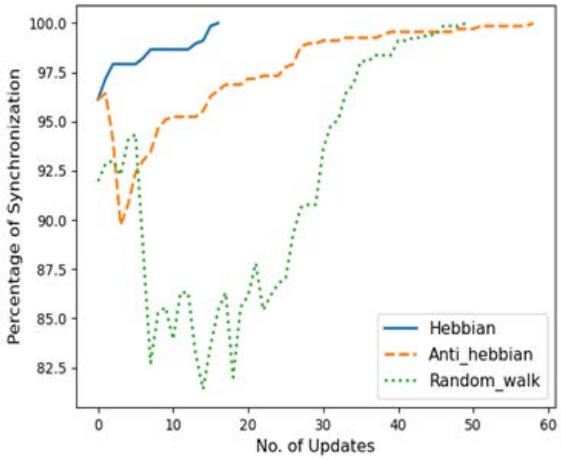
For key reconciliation results, the key length inputted in the KR_Analysis tab is shown in Fig. 15.

The related histogram and synchronization graphs between Alice and Bob according to the different learning algorithms regarding the inputted key length of 128 bits are found which are shown in Fig. 16 and Fig. 17 respectively.

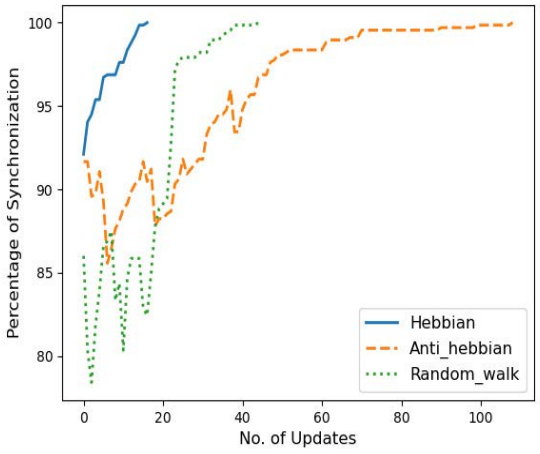
From the histogram in Fig. 16, synchronization graphs in Fig. 17, and Table-2 for 128 bit 40,000 keys, it is apparent that for reconciliation purposes Hebbian learning algorithm requires a less average number of updates and synchronization time than the Anti-Hebbian and Random-Walk learning algorithms. Random-Walk and Anti-Hebbian learning



(a) Synchronization graph for 192 bit Random Key 1

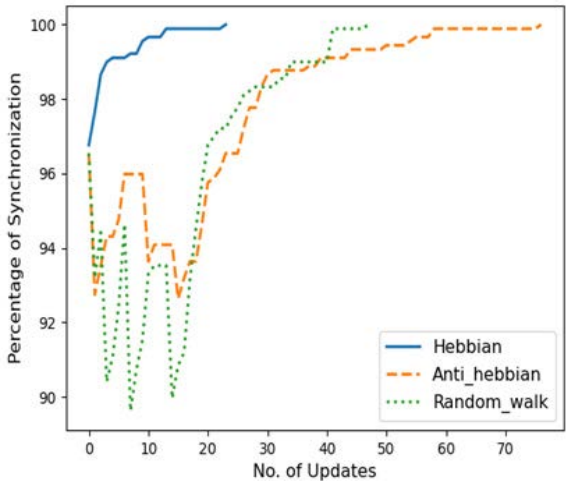


(b) Synchronization graph for 192 bit Random Key 2

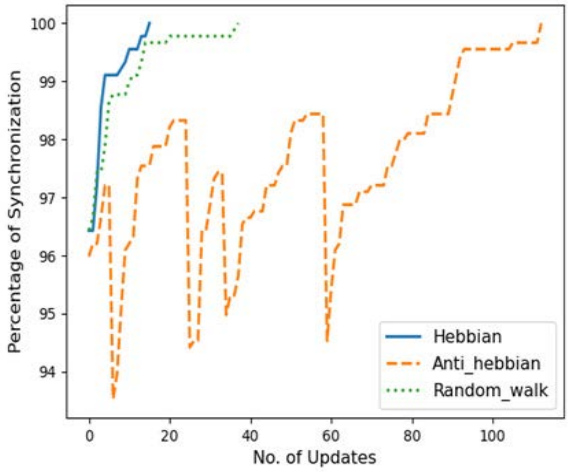


(c) Synchronization graph for 192 bit Random Key 3

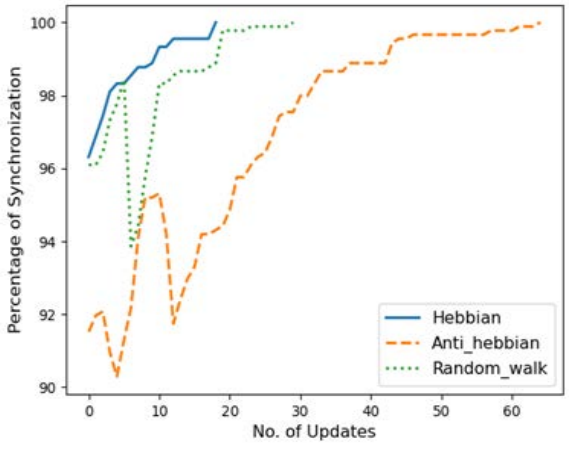
FIGURE 19. Updates vs. percentage of synchronization graphs for 3 random 192 bit keys according to different learning algorithms.



(a) Synchronization graph for 256 bit Random Key 1



(b) Synchronization graph for 256 bit Random Key 2



(c) Synchronization graph for 256 bit Random Key 3

FIGURE 20. Visual for no. of updates vs. percentage of synchronization for 3 random 256 bit keys according to different learning algorithms.

algorithms hold second and third positions respectively concerning less average no. of updates and synchronization time.

The histogram in Fig. 18, synchronization graphs in Fig. 19, and Table-3 generated from 192 bit 30,000 keys clearly state that Hebbian, Random-Walk, and Anti-Hebbian learning algorithms hold first, second, and third positions respectively in terms of less average no. of updates and synchronization time.

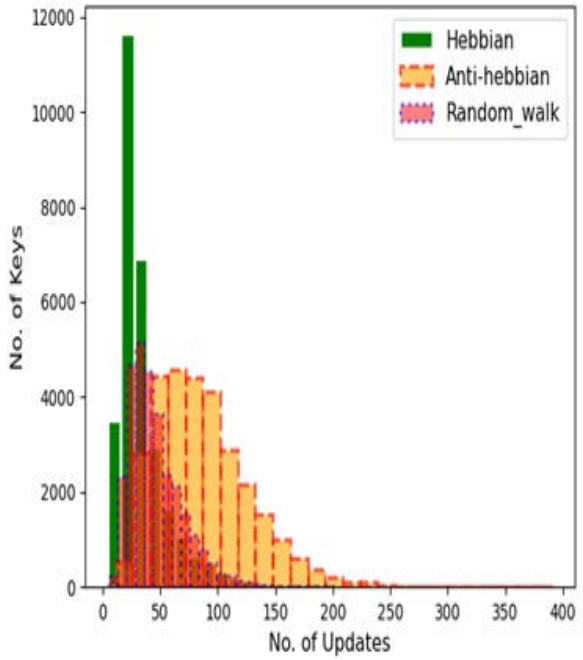


FIGURE 21. No. of updates vs. no. of keys histogram for 256 bit 30,000 keys according to different learning algorithms.

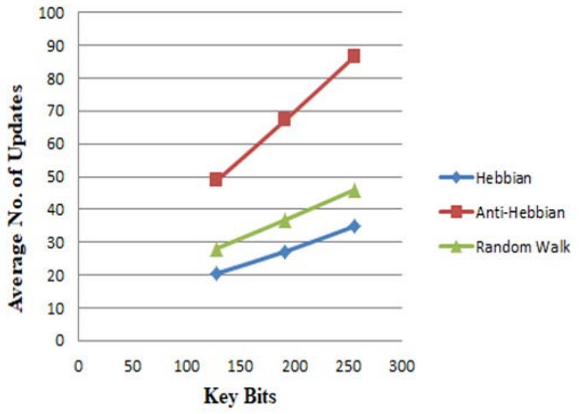


FIGURE 22. Key Bits vs. Average No. of Updates according to different learning algorithms.

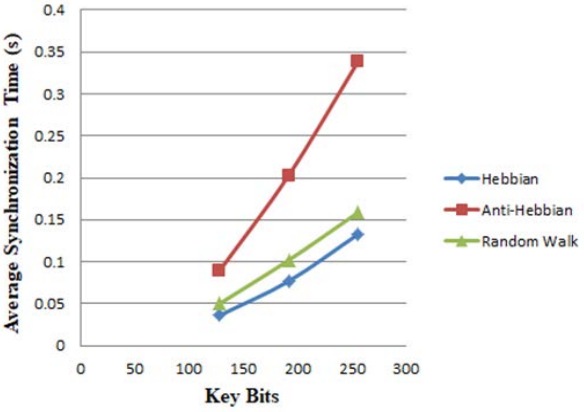
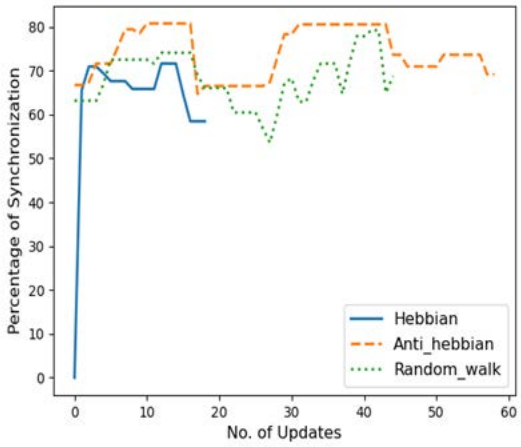
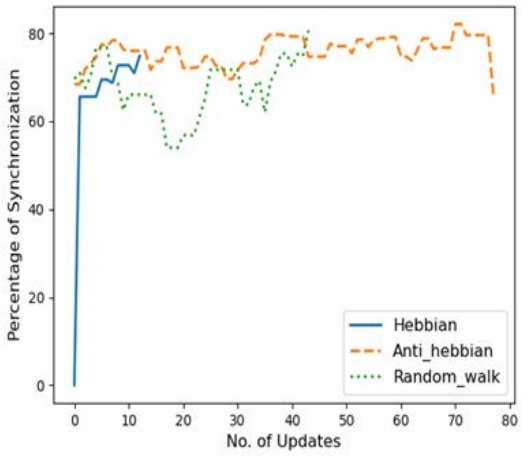


FIGURE 23. Key Bits vs. Average Synchronization Time according to different learning algorithms.

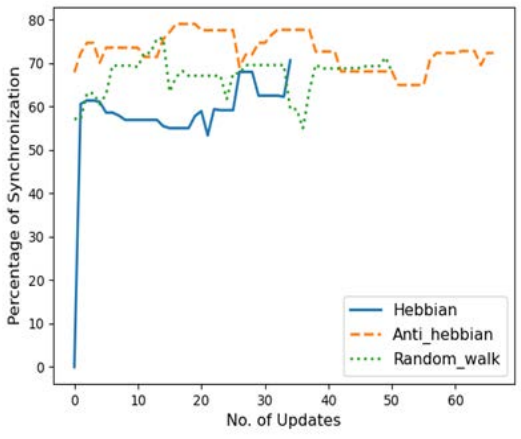
The above synchronization graphs in Fig. 20, histogram in Fig. 21, and Table-4 created from 256 bit 30,000 keys



(a) Eve's Synchronization graph for a 128 bit Random Key



(b) Eve's Synchronization graph for a 192 bit Random Key



(c) Eve's Synchronization graph for a 256 bit Random Key

FIGURE 24. Eve's synchronization graphs according to different learning algorithms for 3 random keys of 128 bit, 192 bit and 256 bit respectively.

apparently interpret the situation again that in case of reconciliation Hebbian algorithm performs better than Anti-Hebbian and Random-Walk because it needs less average no. of updates and synchronization time than others. Then Random-Walk holds the second position because it requires

an average no. of updates and synchronization time more than the Hebbian learning algorithm but less than Anti-Hebbian. Finally, the Anti-Hebbian learning algorithm needs the highest average no. of updates and synchronization time.

From the above observations, it is clear that for reconciliation Hebbian learning algorithm performance is better than Anti-Hebbian and Random-Walk learning algorithms. Another observation is that with the increase of key bits average no. of updates and synchronization time increase gradually according to different learning algorithms which have been shown in the above Fig. 22 and Fig. 23 respectively.

VII. SECURITY ANALYSIS

From the synchronization graphs of Eve for 3 random keys of different key lengths according to different learning algorithms illustrated in Fig. 24, it is clear that Eve tries to synchronize with Alice or Bob. Since Eve's sifted key is wholly random, Eve cannot achieve 100 percent synchronization after the reconciliation process while Alice and Bob achieve 100 percent synchronization after a few updates because of Alice and Bob's almost identical sifted keys.

From Eve's synchronization graphs in Fig. 24, it is evident that Eve's percentage of synchronization is always below 90 percent according to different learning algorithms. As a consequence, Eve cannot generate the expected similar keys like Alice and Bob after the reconciliation process. So, this aspect has been illustrated by artificial neural network based key reconciliation analysis using different learning algorithms which confirms the effectiveness and security of the proposed modified key sifting scheme along with artificial neural network based key reconciliation of QKD.

VIII. CONCLUSION

In BB84, during the public discussion, Alice has to compare her whole basis sequence to the basis sequence sent by Bob. In the proposed key sifting stage Alice only compares Bob's diagonal basis with her corresponding basis and Bob only compares Alice's rectilinear basis with his corresponding basis. These comparisons and most of the actions on both sides are carried out simultaneously in the proposed scheme. As a result, this proposed key sifting stage requires less time than the key sifting of conventional BB84 protocol in most cases. This feature of the proposed modified key sifting scheme makes it an alternative to the traditional BB84 key sifting scheme and thus helps to improve the performance of the quantum key distribution (QKD) system in quantum cryptography. The simulation result of the reconciliation process between Alice and Bob using different learning algorithms shows that the Hebbian algorithm requires the least average number of updates and synchronization time. From the simulation, it is also apparent that in the case of the less average number of updates and synchronization time requirement Random-Walk and Anti-Hebbian algorithms hold the second and third position respectively. So, this reconciliation analysis helps in optimum learning algorithm selection in the case of artificial neural network based key reconciliation of QKD. Moreover, from the synchronization graphs between Alice and Eve using different learning algorithms, it is proved that Eve cannot achieve 100 percent synchronization with Alice which verifies the security proof of artificial neural network based key reconciliation in quantum cryptography. So, this paper interpreting the proposed modified key sifting scheme along with the artificial neural network based key reconciliation analysis using different learning algorithms is beneficial for the performance up-gradation of the Quantum Key Distribution (QKD) system in Quantum Cryptography.

REFERENCES

[1] T. Zhou, J. Shen, X. Li, C. Wang, and J. Shen, "Quantum cryptography for the future internet and the security analysis," *Secur. Commun. Netw.*, vol. 2018, pp. 1–7, Feb. 2018.

[2] G. S. Vernam, "Cipher printing telegraph systems: For secret wire and radio telegraphic communications," *Trans. Amer. Inst. Electr. Eng.*, vol. 45, no. 2, pp. 109–115, Feb. 1926.

[3] C. E. Shannon, "Communication theory of secrecy systems," *Bell Labs Tech. J.*, vol. 28, no. 4, pp. 656–715, Oct. 1949, doi: 10.1002/j.1538-7305.1949.tb00928.x.

[4] B. Qi, L. Qian, and H.-K. Lo, "A brief introduction of quantum cryptography for engineers," 2010, *arXiv:1002.1237*.

[5] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[6] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comput.*, vol. 48, no. 177, pp. 203–209, Jan. 1987.

[7] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976.

[8] T. M. Fernandez-Carames, "From pre-quantum to post-quantum IoT security: A survey on quantum-resistant cryptosystems for the Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6457–6480, Dec. 2019.

[9] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997.

[10] M. Mosca, "Cybersecurity in an era with quantum computers: Will we be ready?" *IEEE Secur. Privacy*, vol. 16, no. 5, pp. 38–41, Sep./Oct. 2018.

[11] P. W. Shor and J. Preskill, "Simple proof of security of the BB84 quantum key distribution protocol," *Phys. Rev. Lett.*, vol. 85, no. 2, pp. 441–444, Jul. 2000.

[12] S. Wiesner, "Conjugate coding," *SIGACT News*, vol. 15, no. 1, pp. 78–88, Jan. 1983.

[13] L. O. Mailloux, D. D. Hodson, M. R. Grimaila, R. D. Engle, C. V. McLaughlin, and G. B. Baumgartner, "Using modeling and simulation to study photon number splitting attacks," *IEEE Access*, vol. 4, pp. 2188–2197, 2016.

[14] M. Mehic, O. Maurhart, S. Rass, D. Komosny, F. Rezac, and M. Voznak, "Analysis of the public channel of quantum key distribution link," *IEEE J. Quantum Electron.*, vol. 53, no. 5, pp. 1–8, Oct. 2017.

[15] M. Elboukhari, A. Azizi, and M. Azizi, "Quantum key distribution in practice: The state of art," in *Proc. 5th Int. Symp. I/V Commun. Mobile Netw.*, Rabat, Morocco, Sep. 2010, pp. 1–4.

[16] L. Hu, H. Liu, and Y. Lin, "Parameter optimization of cascade in quantum key distribution," *Optik*, vol. 181, pp. 156–162, Mar. 2019.

[17] J. S. Johnson, M. R. Grimaila, J. W. Humphries, and G. B. Baumgartner, "An analysis of error reconciliation protocols used in quantum key distribution systems," *J. Defense Model. Simul. Appl. Methodol. Technol.*, vol. 12, no. 3, pp. 217–227, Jul. 2015.

[18] G. Brassard and L. Salvail, "Secret-key reconciliation by public discussion," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 765, T. Helleseth, Ed. Berlin, Germany: Springer-Verlag, 1994, pp. 410–423.

[19] C. H. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin, "Experimental quantum cryptography," *J. Cryptol.*, vol. 5, no. 1, pp. 3–28, Jan. 1992.

[20] T. Pedersen and M. Toyran, "High performance information reconciliation for QKD with CASCADE," *Quantum. Inf. Comput.*, vol. 15, pp. 419–434, Apr. 2015.
[21] W. T. Buttler, S. K. Lamoreaux, J. R. Torgerson, G. H. Nickel, C. H. Donahue, and C. G. Peterson, "Fast, efficient error reconciliation for quantum cryptography," *Phys. Rev. A, Gen. Phys.*, vol. 67, no. 5, May 2003, Art. no. 052303.
[22] R. G. Gallager, "Low density parity check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.
[23] R. G. Gallager, "Low-density parity-check codes," in *Monograph*. Cambridge, MA, USA: MIT Press, 1963.
[24] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
[25] É. S. Dorokhin, W. Fuertes, and E. Lascano, "On the development of an optimal structure of tree parity machine for the establishment of a cryptographic key," *Secur. Commun. Netw.*, vol. 2019, pp. 1–10, Mar. 2019.
[26] S. Jeong, C. Park, D. Hong, C. Seo, and N. Jho, "Neural cryptography based on generalized tree parity machine for real-life systems," *Secur. Commun. Netw.*, vol. 2021, pp. 1–12, Feb. 2021.
[27] I. Meraouche, S. Dutta, H. Tan, and K. Sakurai, "Neural networks-based cryptography: A survey," *IEEE Access*, vol. 9, pp. 124727–124740, 2021.
[28] M. Niemiec, "Error correction in quantum cryptography based on artificial neural networks," *Quantum Inf. Process.*, vol. 18, no. 6, pp. 1–18, Apr. 2019.
[29] S. Cobourne, "Quantum key distribution protocols and applications," Dept. Math., Roy. Holloway, Univ. London, Egham, U.K., Tech. Rep. RHUL-MA-2011-05, Mar. 2011.
[30] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, pp. 802–803, Oct. 1982.
[31] D. Stebila, M. Mosca, and N. Lütkenhaus, "The case for quantum key distribution," in *Proc. Int. Conf. Quantum Commun. Quantum Netw.* Berlin, Germany: Springer, 2009, pp. 283–296.
[32] C. H. Bennett, G. Brassard, and J.-M. Robert, "Privacy amplification by public discussion," *SIAM J. Comput.*, vol. 17, no. 2, pp. 210–229, Apr. 1988.
[33] J. L. Carter and M. N. Wegman, "Universal classes of hash functions," *J. Comput. Syst. Sci.*, vol. 18, no. 2, pp. 143–154, Apr. 1979.
[34] M. N. Wegman and J. L. Carter, "New hash functions and their use in authentication and set equality," *J. Comput. Syst. Sci.*, vol. 22, no. 3, pp. 265–279, Jun. 1981.
[35] H. Singh, D. L. Gupta, and A. K. Singh, "Quantum key distribution protocols: A Review," *IOSR J. Comp. Eng.*, vol. 16, no. 2, pp. 1–9, 2014.
[36] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," in *Proc. IEEE Int. Conf. Comput., Syst., Signal Process.*, Bengaluru, India, Dec. 1984, pp. 175–179.
[37] C. H. Bennett, "Quantum cryptography using any two nonorthogonal states," *Phys. Rev. Lett.*, vol. 68, no. 21, pp. 3121–3124, May 1992.
[38] H. Bechmann-Pasquinucci and N. Gisin, "Incoherent and coherent eavesdropping in the six-state protocol of quantum cryptography," *Phys. Rev. A, Gen. Phys.*, vol. 59, no. 6, pp. 4238–4248, Jun. 1999.
[39] V. Scarani, A. Acín, G. Ribordy, and N. Gisin, "Quantum cryptography protocols robust against photon number splitting attacks for weak laser pulse implementations," *Phys. Rev. Lett.*, vol. 92, no. 5, Feb. 2004, Art. no. 057901.
[40] E. H. Serna, "Quantum key distribution from a random seed," Nov. 2013, *arXiv:1311.1582*.
[41] A. I. Nurhadi and N. R. Syambas, "Quantum key distribution (QKD) protocols: A survey," in *Proc. 4th Int. Conf. Wireless Telematics (ICWT)*, Nusa Dua, Indonesia, Jul. 2018, pp. 1–5.
[42] A. K. Ekert, "Quantum cryptography based on Bell's theorem," *Phys. Rev. Lett.*, vol. 67, no. 6, pp. 661–663, Aug. 1991.
[43] C. H. Bennett, G. Brassard, and N. D. Mermin, "Quantum cryptography without Bell's theorem," *Phys. Rev. Lett.*, vol. 68, pp. 557–559, Feb. 1992.
[44] K. Inoue, E. Waks, and Y. Yamamoto, "Differential-phase-shift quantum key distribution using coherent light," *Phys. Rev. A, Gen. Phys.*, vol. 68, no. 2, Aug. 2003, Art. no. 022317.
[45] N. Gisin, G. Ribordy, H. Zbinden, D. Stucki, N. Brunner, and V. Scarani, "Towards practical and fast quantum cryptography," 2004, *arXiv:0411022*.
[46] A. Ruttor, "Neural synchronization and cryptography," Ph.D. dissertation, Fakultät für Physik Astronomie, Inst. für Theoretische Physik Astrophysik, Würzburg, Germany, 2006.

CHITRA BISWAS received the B.Sc. (Eng.) degree in electronics and telecommunication engineering from the University of Science and Technology Chittagong (USTC), Bangladesh, in 2011. She is currently pursuing the M.Sc. degree in computer science and engineering with the Chittagong University of Engineering and Technology.

From 2012 to 2020, she was a full-time Lecturer with the Department of Electronics and Telecommunication Engineering, USTC. She is also working as a Research Assistant with the Department of Computer Science and Engineering, Chittagong University of Engineering and Technology. Her research interests include cryptography, cybersecurity, and machine learning.

MD. MOKAMMEL HAQUE received the B.Sc. (Eng.) degree in computer science and engineering from the Chittagong University of Engineering and Technology (CUET), Bangladesh, the M.Sc. degree in computer engineering from Kyung Hee University, South Korea, and the Ph.D. degree in computing from Macquarie University, Australia.

He is currently working as a full-time Professor with the Department of Computer Science and Engineering, CUET. His research interests include cybersecurity, computer networks, cryptography, machine learning, and algorithms.

UDAYAN DAS GUPTA received the B.Sc. (Eng.) degree in computer science and engineering from Mawlana Bhashani Science and Technology University, Tangail, Bangladesh, in 2010. He is currently pursuing the M.Sc. degree in computer science and engineering with the Chittagong University of Engineering and Technology, Bangladesh.

He is also a Programmer by profession. His research interests include cryptography, machine learning, data security, and data mining.