Received: November 10, 2022. Revised: January 27, 2023.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Effective Quantum Key Distribution Using Modified Key Sifting Scheme for Cloud Storage Security

Varsha Pravin Hole[1]* **Nikahat Mulla[1]** **Pravin Basavraj Hole[2]**

[1]*Department of Information Technology, Sardar Patel Institute of Technology, Mumbai, India*
[2]*Department of Computer Engineering, Terna Engineering College, Mumbai, India*
* Corresponding author's Email: varsha_hole@spit.ac.in

Abstract: Cloud storage has gained extensive attention due to its low cost, global scalability, and improved agility. Data privacy is an important issue in cloud storage because breaches of personal information or business secrets are occurred by malicious hackers or unauthorized cloud operators. To address this issue, a Modified Key Sifting Scheme (MKSS) is proposed for developing an effective Quantum Key Distribution (QKD) with Two-Way Identity Authentication (TWIA) to provide security in the cloud storage. The Tree Parity Machine (TPM) is used for proposing a MKSS which used to minimize the key distribution time based on the synchronization. This MKSS-QKDTWIA considers four different objects such as data user, data owner, server and data storage in a communication scenario. The performance of the MKSS-QKDTWIA is analyzed by means of key computation time, processing time, encryption & decryption time, and memory utilization. Existing research such as Modified Random Fibonacci Cryptographic (MRFC), Software Stack-QKD (SS-QKD) and Enhanced Cloud Security Model based on QKD Protocol (ECSM-QKDP) are used to evaluate the performance of MKSS-QKDTWIA. The key computation time of MKSS-QKDTWIA for 50 attribute sets in each key is 16.7s which is less when compared to the MRFC, SS-QKD and ECSM-QKDP.

Keywords: Cloud storage, Data Privacy, Key computation time, Modified key sifting scheme, Quantum key distribution, Two-way identity authentication.

1. Introduction

Nowadays, cloud infrastructure is one of the developing technology in the field of Information and Technology (IT). Further, cloud computing is one of the emerging computational representations in the IT field. The global availability of computing resources such as operating systems, software applications, network infrastructure and storage devices are indicated by cloud metaphors and Internet technologies. The main goal of cloud computing is to deliver configurable computing resources according to the demand and suitability of users [1, 2]. Cloud computing alters the idea of sharing, processing, and saving data from conventional IT into integrated services provided by cloud providers [3]. The on-demand routing method is easy for sharing routes broadcast or non-broadcast information, and it allows users for utilizing the required services of cloud computing without any contact among the consumers and service providers [4]. Customers who are using the cloud services don't require to know about the exact location of their data in the cloud or be conscious of which machine in the network is developing its computational tasks [5]. The primary service delivered by cloud computing is cloud storage which allows users for storing their data from anywhere on cloud servers [6, 7].

Cloud storage recognizes multi-regional, multi-domain, and large-scale data distribution. The advantages of cloud storage are low-cost, easy user maintenance, on-demand resources, on demand storage and easy storage management [8]. But, data security is a crucial issue, because of the severe susceptibility and hitch in cloud storage [9, 10]. In conventional approaches, encrypting data before transmitting and saving it as ciphertext in the cloud is an extensively applied approach for performing data privacy protection [11, 12, 13]. Conventional cryptographic cloud storage needs the management of a huge amount of keys. Generally, the data owner is required for generating a key pairs for an each file and forward the matching decryption keys for the files which are shared with the desired users. The storage management and dissemination of a huge amount of keys are very challenging in the conventional cryptography method [14]. Hence, the quantum cryptography scheme, also referred to as quantum key Distribution is considered in this work for enhancing the security of cloud storage. This quantum cryptography depends on principles of quantum mechanics for performing diverse data operations [15].

The contributions of this research are summarized as follows:

- An effective quantum key cryptography namely QKD with TWIA is developed for enhancing the security among the data owner (Alice), data user (Bob), Center (Server) and data storage of cloud infrastructure.
- The time required during key generation is minimized based on the synchronization process obtained using the TPM based MKSS. The variation in the registration information at each time helps to enhance the robustness against unauthorized users.

The remaining paper is organized as follows: section 2 delivers the related works about cloud security. Detailed information on the MKSS-QKDTWIA method is provided in section 3. The outcomes of the MKSS-QKDTWIA are given in section 4 followed by the conclusion presented in section 5.

2. Related work

Zhu [16] presented the quantum fully homomorphic encryption (QFHE) based on the quantum information processing framework. The developed QFHE depended on the universal quantum circuit that permitted arbitrary quantum transformation. The encryption key used by the QFHE was generated using a GHZ-like state. The quantum one-time pad was used by both the encryption and the decryption key whereas the decryption key was different from the encryption key. The encryption of data was permitted by QFHE without any decryption that was used to enhance the data security. The developed QFHE was required to concentrate on the key generation and encryption time for further improving the security performances.

Wang [17] developed the identity-based broadcast encryption (IBBE) approach to develop the data-sharing system through the lattice. In the developed data broadcasting, no one has an idea about the authenticated data receiver’s identity except the data owner. The IBBE achieved identity privacy and data confidentiality simultaneously during the data transmission. The developed IBBE was processed with constant ciphertext length, public key size and private key size. In this IBBE, the existing receiver does not updated their privacy, even when new receiver was joined in the cloud. A frequent privacy update was essential to secure the data transmitted among the users.

Sumathi, and Sangeetha [18] presented the group key-based attribute encryption based on the MRFC for cloud storage. The random numbers were selected using the random Fibonacci cryptography that was used to choose the random number which was used as a private key for cloud storage. The elliptic curve technique using MRFC was used to enhance the resistance of group key identification against the adversary. The error correction was required in MRFC to minimize the key distribution time.

Pedone [19] designed the software stack (SS) for introducing the quantum key distribution (QKD) systems in a cloud environment. The developed software stack was combined with the modern infrastructures and controls a cloud-native method for improvement. Further, the software stack allowed the monitoring, integration, and management of numerous QKD systems in the infrastructure. This SS-QKD was used to enhance the security, but two way authentication was not considered in the cloud storage. For an effective communication, two way authentication was much essential to authenticate the users in both the transmitter and receiver.

Sundar [20] presented an enhanced cloud security model based on QKD protocol (ECSM-QKDP) to provide cloud storage security. This research assumed the communication scenario among three objects such as data owner, cloud server and Legitimate User (LU) whereas the quantum keys are exchanged in two steps. The 1st step utilized BB84 QKDP and the second step used the secure authentication protocol (SAP) that was developed according to the distance bounding and secure keys. The key discovery time was high when the depth of the hierarchical access tree used in the encryption is high at cloud storage.

3. MKSS-QKDTWIA method

In this research, the security in the cloud storage is ensured by developing the MKSS for QKD with

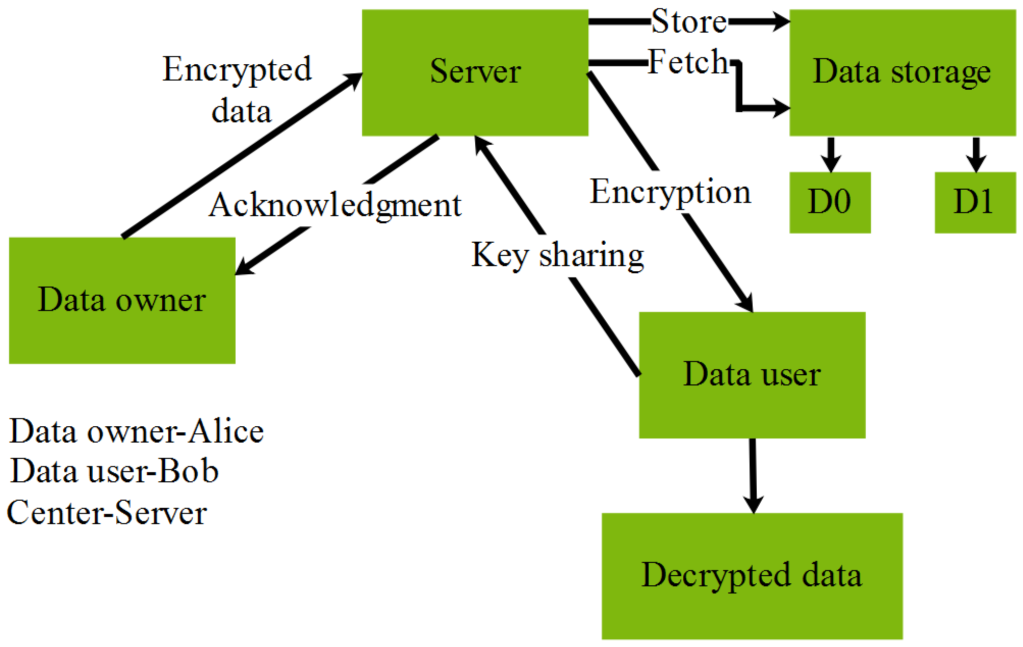

Figure. 1 Block diagram of the MKSS-QKDTWIA

TWIA. The major elements used in this MKSS-QKDTWIA are data owner (Alice), data user (Bob), Center (Server) and data storage. Since, the TPM based MKSS is used to achieve the synchronization that helps to minimize the operating time. Further, the changes in the registration information each time is used to improve the robustness against unauthorized users. The block diagram of the MKSS-QKDTWIA is shown in Fig. 1.

3.1 QKD with TWIA

The steps processed in the MKSS-QKDTWIA method are given as follows:

- The N length of key and M amount of bases are used by the key distribution that is required to be understood by the server, data owner and data user. The Pseudo-random operations $P_A()$ and $P_B()$ are exchanged by the data owner, data user and server. The keys s_a and s_b are transferred by using the data owner, data user and server. The method of choosing related keys is exchanged in MKSS-QKDTWIA. data owner/data user chooses all the exchanged data and utilizes the one-way hash function for hashing it as the Pseudo Random Number (PRN)'s seed.
- The PRN computed by the data owner is $k_a = P_A(s_a)$, where k_a's length is $l = |P_A(s_a)|$. The equivalent PRN is computed by the server, where $l/m \geq N, m = \lceil \log_2 M \rceil$. The same information is exchanged with the server from Bop for performing the same thing.
- The random numbers $a_i, b_i, i \in \{1,2,\dots,N\}$ generated by the data owner and Bop. Choose the related base for encoding the true random number $|\varphi_i(k_a) + a_i \pi/2\rangle, |\varphi_i(k_b) + b_i \pi/2\rangle$ in control of the PRN, where $\varphi_i(k)$ is defined by the $m \times (i-1) + 1$ to $m \times i$ bits of k and simultaneously exchange it to the server.
- The server randomly alters the data owner and data user based on the variation among the data owner and data user based on the coin random number. The bases chosen by the two are reliable. $|\varphi_i(k) + a_i \pi/2\rangle, |\varphi_i(k) + b_i \pi/2\rangle$, where $k = k_a$ or k_b.
- Only $D0$ or $D1$ are considered as success actions. The quantum states given by both sides are reliable when the $D0$ is successful. On the other hand, the quantum states given by both sides are opposite, when the $D1$ is successful. If $D1$ is successful, noise is generated while matching the keys which causes an error in the authentication process.
- The communication is terminated, when the error is the step 5 cross beyond the definite value during the verification of authentication. Else, it processes the first five steps again until adequate data is gained.
- A definite amount of raw keys are chosen by the data owner and data user and evaluated with the traditional channel. The communication is terminated, when an amount of errors crosses a defined limit. Hence, the server is either blocked/ attacked from broadcasting.

- The error correction technique is exchanged by the data owner and data user over the classical channel and errors are mitigated in the raw key. The error correction is achieved by using the TPM based MKSS. Further, privacy amplification is used to create the final key.

3.2 Key generation rate

The data owner and data user have a similar base for quantum states discovered by the server, when both the data owner and data user are authentic users. The rate of key generation is expressed in Eq. (1).

$$R \equiv Q^{\varphi}(1 - f_e(E^{\varphi})H_2(E^{\varphi})) \quad (1)$$

Where, the rate of the raw key is Q^{φ} which server states the successful outcome while the data owner and data user broadcast server photon; the quantum bit error rate is defined as E^{φ}; an inadequacy function of error correction is represented as f_e and binary entropy function is denoted as $H_2()$.

3.3 Reliability of TWIA

The identity authentication's reliability defines how the identity is trustworthy after identity authentication. The reliability analysis is done in two levels in which the first layer defines how much identity information is proved to the identity information. Next, the information of identity is identified as it is either forged or not in the second layer. In 1st layer, all the identity information is confirmed, which shows that verification has a trustworthiness of 100%. For instance, the verifier of symmetric verification collects all the identity data of the prover. Subsequently, whether the information is reliable with their information or not is evaluated by a verifier. The authentication of identity is successful when all the information is reliable. However, there is another approach for evaluating the information about identity authentication which is analyzing a portion of identity data. In this MKSS-QKDTWIA, all data related to identity is analyzed before providing authentication. However, some information is obtained to perform verification. The PRN Seed Key Construction (PRNSKC) is used to extract the information.

The process of PRNSKC is stated as follows:

1. From the registration data, the initial location L of the seed key is selected based on the transfer between the data owner and server at the traditional channel.
2. In the conventional channel, the data owner and server transferred the true binary random string S. The amount of locations for value 1 in the randomly chosen string S is larger when compared to the length needed by the seed key.
3. The time stamp T is transferred using the data owner and server over a public information channel.
4. Data owner and server are initiated from the location L at its registration data. Next, the random value S and registration data are used for performing the "and" operation. Further, the data from the related location 1 is taken out and deleted the amount of locations of 0 in S, and the remaining value is k'_a.
5. The time stamp T is included in the data from the previous steps, then the seed key is $k_a = k_a \parallel T$ of PRN.

The k_a length is determined by using the following ways: If t is the pseudo-random function $P_A()$ for cyclic node, then the length of preferred PRN is l and the length of k_a is l_{ka}. The following conditions given in Eq. (2) are satisfied by l_{ka}

$$l_{ka}.t.\alpha > 1 \quad (2)$$

Where, the security parameter is $0 < \alpha < 1$. Eq. (3) shows the needed length for k_a.

$$2^{l_{ka}.t.\alpha} > 1 \quad (3)$$

The PRNSKC guaranteed that the data broadcasted by the data owner and server is dissimilar each time. Subsequently, the dissimilar information is authenticated and transferred to the cloud. The registration information selection is altered in position and length at the same time which effectively maximizes the robustness against eavesdroppers from identifying the registration information. Eavesdroppers also steal the identified data by mimicking like data owner. However, the Eavesdropper only discovers the quantum state and identity information of the data user, even though it has the identity information of the data owner. Hence, it is confirmed that the registration data of other users isn't disclosed because of the exposure of information from a few users. Eq. (4) expresses the difficulty estimation for Eve's eavesdropping to steal the data owner's registration data on the server.

$$R \xrightarrow{PRNSKC} k_a \xrightarrow{P()} 1 \xrightarrow{l/m} \varphi(N) \quad (4)$$

Where, the registration information of the data owner in the server is denoted as R; the PRN seed is represented as k_a ; pseudo-random function is defined as $P()$; the PRN which identifies the quantum base is denoted as 1 and the transmitted final quantum state is denoted as $\varphi(N)$. The quantum states of $\varphi(N)$ are computed by Eve. The information exchanged between data owner and server is changed each time, hence it is difficult for an unauthorized person to identify the registration data using quantum state.

3.4 Security of TWIA

In this two-way authentication, the PRNs used by the data owner and data user are known by the server. The server confirms that each computation utilizes the same base in a normal scenario, therefore it has the capacity for obtaining an effective discovery result each time. The main difference between the proposed method and from existing QKD protocol is that it doesn't broadcast the encoding base and computing base in the traditional channel, so eavesdroppers can't achieve the information about identity used to protect the authentication information.

The data owner, data user and server have the keys, the server varies the base among data owner and data user to a similar base before the computation. Consider the data owner is an unauthorized user, where the quantum state which is broadcasted by the data owner is $f\varphi_i$. The authorized user is the data user and the quantum state broadcasted by data user is $f\varphi_j$. Here, the base utilized by the data owner and data user is denoted by i and j respectively. Consider, the random numbers are a & b and server's rotation operation based on the coin random number is denoted as U^{coin} . Fix $|\varphi_i'\rangle = U^{coin}\left|\varphi_i + a.\frac{\pi}{2}\right\rangle, |\varphi_j'\rangle = U^{coin+1}\left|\varphi_j + b.\frac{\pi}{2}\right\rangle$ and the Eq. (5) shows the rate of key generation.

$$R' \equiv \frac{1}{M}\left(Q^{\varphi} - Q^{\varphi} f_e(E^{\varphi}) H_2(E^{\varphi})\right) \tag{5}$$

R' is very rough evaluation, because probability of choosing base is an average probability. Server defines the whether data owner or data user is an unauthorized user over the computation result. If both the data owner and data user is unauthorized user, then the probability is less in MKSS-QKDTWIA.

3.5 Error Correction using TPM for MKSS

In the proposed modified key sifting scheme (MKSS), the TPM [21] is used to perform the error correction TWIA process. The quantum channel exchange is performed similar to the conventional BB84 [20], but the modification is accomplished in public channel exchange. The ECSM-QKDP [20] is performed using pseudo random number, but the proposed MKSS synchronizes the random number in cloud storage. This synchronization of random numbers using MKSS is used to minimize key distribution time than the conventional BB84 key sifting of ECSM-QKDP [20]. Here, the TPM is used to generate the BB84 with MKSS by using the loss function of Anti-Hebbian rule. In the key reconciliation stage, a famous artificial neural network namely TPM is used to perform the error correction. The TPM is used by the sender, receiver and attackers are the multi-layer feedforward networks. The developed TPM has K hidden units and O amount of input neurons where this network has only one output neuron.

The data owner (Alice), data user (Bob), center (Server) and data storage are used the TPM that are synchronized after performing the mutual learning. The same random output is generated by user for synchronizing the TPM and the output is computed from an each TPM. The learning process of the TPM is started when the data owner's TPM output matches with data user's TPM output. The data user required to generate another input, when the outputs aren't identical to each other. Hence, the Anti-Hebbian rule expressed in Eq. (6) is used for weight synchronization when the output of both the TPMs are identical in cloud.

$$w_{i,j}^{+} = g(w_{i,j} - x_{i,j}\tau\theta(\sigma_i\tau)\theta(\tau^A\tau^B)) \tag{6}$$

Where, the updated weight is denoted as w^+; $i = 1,2,\dots,K$ represents the ith hidden unit of TPM; $j = 1,2,\dots,O$ represents the amount of inputs in each neuron at the hidden layer; TPM output is denoted as τ; output of hidden layer for neuron i is σ_i and input values is $x_{i,j}$. Eq. (7) expresses the θ value.

$$\theta(\sigma_i\tau) = \begin{cases} 0, & if\ \sigma_i \neq \tau \\ 1, & if\ \sigma_i = \tau \end{cases} \tag{7}$$

The TPM based MKKS is accomplished based on the below steps:

Received: November 10, 2022. Revised: January 27, 2023.

Table 1. Analysis of key computation time

Number of attributes set in each key	Key computation time (Seconds)			
	MRFC [18]	SS-QKD [19]	ECSM-QKDP [20]	MKSS-QKDTWIA
10	5.5	5.2	4.6	3.4
20	11.0	10.8	8.5	6.1
30	17.3	16.5	14.6	9.7
40	28.6	25.3	21.8	13.9
50	31.7	30.1	25.8	16.7

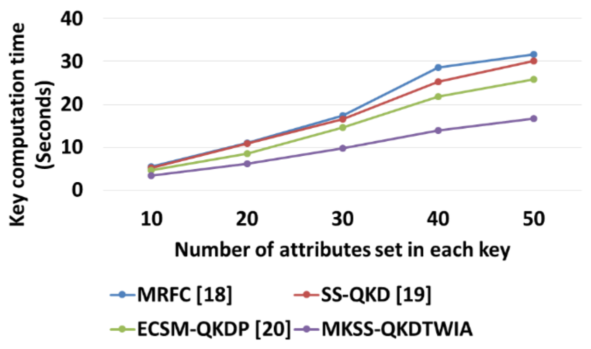

Figure. 2 Comparison of key computation time

- The data owner and data user are utilized their TPMs where those owner and user transformed the sifted key bits as weights in the range of $[-WL, WL]$. A few errors exist in the weight are identified that keep most of weight as same value, because of similar architecture of TPM and weights received from sifted key. The TPM is used to resolve these errors.
- Data owner generates random input and sends to the data user to perform synchronization.
- The outputs of TPM are created for data owner and data user. Data owner informed the data user about its output of TPM.
- The Anti-Hebbian rule based learning is initialized, when the outputs are similar to each other; otherwise, the data owner is required to create another input, when the outputs are different.
- The synchronization of the TPMs is obtained and all weights of TPM are similar, once the iterations are completed.
- At the end, the weight value generated in the limit of $[-WL, WL]$ is again transformed into bit string. All weight values of TPMs are identical, because synchronization is done for both the TPM. Thus, the bit string of data owner is equal to bit string of data user. Therefore, the errors are solved and key reconciliation is achieved in the QKD. The restored key is used as final cryptographic key in the cloud storage.

This two-way authentication proved that there is a rapid decrement in key generation rate along with the increment in cardinalities increment, when there is authorized user exists in the cloud storage. Hence, the unauthorized user in the cloud storage is discovered, when the probability of detection is abnormal.

4. Results and discussion

The outcomes of this MKSS-QKDTWIA method is detailed in this section. The design and simulation of this MKSS-QKDTWIA method is done using the Python 3.7 software where the system is operated with i3 processor and 8GB RAM. The proposed MKSS-QKDTWIA is used to provide security in the cloud storage.

4.1 Performance analysis

The performance of the MKSS-QKDTWIA is analyzed by means of key computation time, processing time, encryption & decryption time and memory utilization. Here, the performances are analyzed based on number of attribute set, file size and a number of files.

The QFHE [16] has to focus on the key generation and encryption time to additionally improve the security. The developed MKSS-QKDTWIA is concentrated on the various time measures such as key computation time, processing time and encryption & decryption time. On the other hand, IBBE [17] doesn't provide the frequent security update for the users of cloud. However, the MKSS-QKDTWIA based key distribution enables the frequent security update among the users of cloud security. Therefore, an existing researches such as MRFC [18], SS-QKD [19] and ECSM-QKDP [20] are used to evaluate the performance of MKSS-QKDTWIA.

Fig. 2 and Table 1 show the analysis of key computation time for MKSS-QKDTWIA with MRFC [18], SS-QKD [19] and ECSM-QKDP [20]. From the analysis, it is known that the computation time of keys for MKSS-QKDTWIA is less than the MRFC [18], SS-QKD [19] and ECSM-QKDP [20]. For example, the computation time of keys of MKSS-QKDTWIA with 50 attribute sets is 16.7s, whereas the ECSM-QKDP [20] obtains 25.8s. The key generation of MKSS-QKDTWIA uses the attribute

Table 2. Analysis of processing time

File size in kb	Processing time (Seconds)			
	MRFC [18]	SS-QKD [19]	ECSM-QKDP [20]	MKSS-QKDTWIA
100	7.1	6.1	4.5	2.8
200	9	7.4	4.7	3
300	10.8	9	6.9	5.4
400	13.5	12.2	9.1	7.3

Table 3. Analysis of encryption time

Number of files	Encryption time (Seconds)			
	MRFC [18]	SS-QKD [19]	ECSM-QKDP [20]	MKSS-QKDTWIA
20	3.2	2.8	1.9	0.6
40	6.4	5.9	3.3	1.9
60	9.8	8	6.2	4.7
80	11.5	9.1	5.6	5
100	14.3	12.7	9.5	7.3

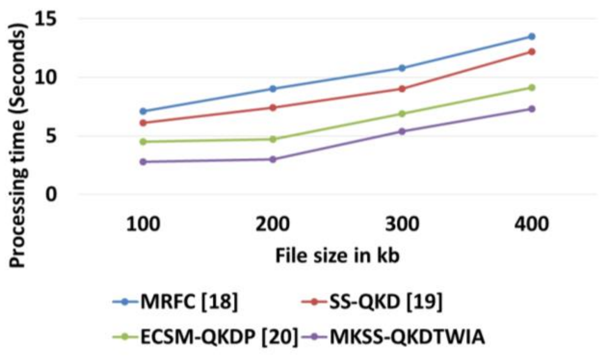

Figure. 3 Comparison of processing time

set of users for generating the PRN which makes key generation easier and makes the key generation time lesser than the other existing methods. Moreover, the synchronization of PRN is used to minimize the key computation time.

The comparison of processing time for MKSS-QKDTWIA with MRFC [18], SS-QKD [19] and ECSM-QKDP [20] is shown in Fig. 3 and Table 2. This analysis shows that the processing time of MKSS-QKDTWIA is less than the MRFC [18], SS-QKD [19] and ECSM-QKDP [20]. For example, the processing time for MKSS-QKDTWIA with a 400kb file is 7.3s, whereas the ECSM-QKDP [20] obtains 9.1s. The MKSS developed in the cloud storage is used to minimize the processing time based on the synchronization of TPMs for both the data owner and data user. The processing time of MKSS-QKDTWIA is reduced due to the authentication of multiple key shares with a fixed length between the two entities of the data owner and data user.

Fig. 4 and Table 3 show the analysis of encryption time for MKSS-QKDTWIA with MRFC [18], SS-QKD [19] and ECSM-QKDP [20]. From the analysis, it is known that the encryption time of MKSS-QKDTWIA is less than the MRFC [18], SS-QKD [19] and ECSM-QKDP [20]. For example, the encryption time for MKSS-QKDTWIA with 100 files is 7.3s, whereas the ECSM-QKDP [20] obtains 9.5s. The attribute set-based key generation using MKSS-QKDTWIA generates the ciphertext with the signature on user attributes which helps to minimize the encryption time.

The comparison of decryption time for MKSS-QKDTWIA with MRFC [18], SS-QKD [19] and ECSM-QKDP [20] is shown in Fig. 5 and Table 4. This analysis shows that the decryption time of MKSS-QKDTWIA is less than the MRFC [18], SS-QKD [19] and ECSM-QKDP [20]. For example, the decryption time for MKSS-QKDTWIA with 100 files is 0.5s, whereas the ECSM-QKDP [20] obtains 2.1s. In MKSS-QKDTWIA, decryption time gets lesser based upon a lesser error in the key formation from the shares of the key generated and both user and data owner has a less complex computable relation on the defined key shares.

Fig. 6 and Table 5 show the analysis of memory utilization for MKSS-QKDTWIA with MRFC [18], SS-QKD [19] and ECSM-QKDP [20]. From the analysis, it is known that the memory utilization of MKSS-QKDTWIA is less than the MRFC [18], SS-QKD [19] and ECSM-QKDP [20]. For example, the memory utilization of MKSS-QKDTWIA with 300kb file is 101.85kb, whereas the ECSM-QKDP [20] obtains 167.85kb. In MKSS-QKDTWIA, the memory usage of the data is reduced with the help of definite key length sharing and reduces the weightage on the memory of encrypted data.

The comparison of verification overhead for MKSS-QKDTWIA with MRFC [18], SS-QKD [19] and ECSM-QKDP [20] is shown in Fig. 7 and Table 6. This analysis shows that the verification overhead of MKSS-QKDTWIA is less than the MRFC [18], SS-QKD [19] and ECSM-QKDP [20]. For example, the verification overhead for MKSS-QKDTWIA

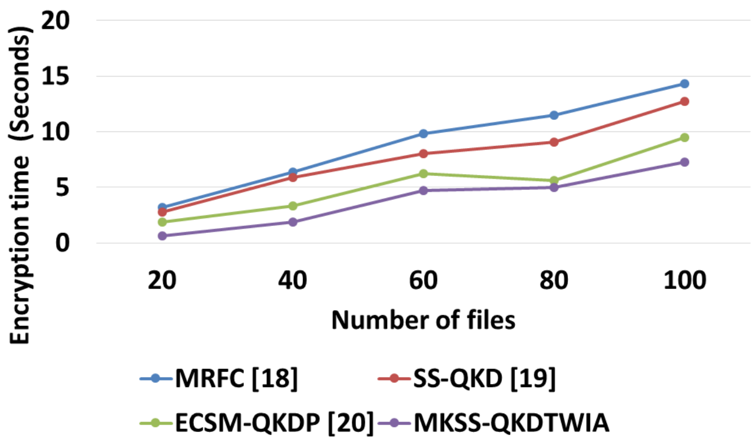

Figure. 4 Comparison of encryption time

Table 4. Analysis of decryption time

Number of files	Decryption time (Seconds)			
	MRFC [18]	SS-QKD [19]	ECSM-QKDP [20]	MKSS-QKDTWIA
20	-1.9	-2.5	-4.8	-6.7
40	-0.9	-1.3	-3.1	-5
60	1.4	0.9	-2.2	-3.3
80	4.5	2	0.2	-1.8
100	5.1	3.9	2.1	0.5

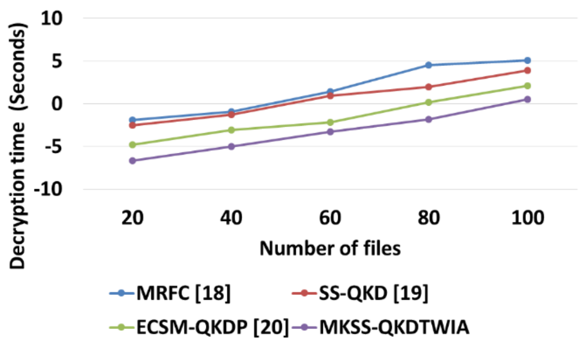

Figure. 5 Comparison of decryption time

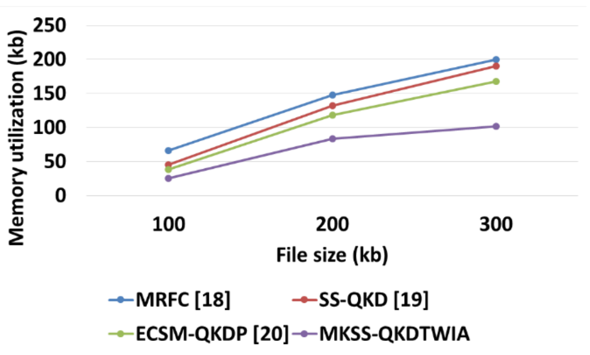

Figure. 6 Comparison of memory utilization

Table 5. Analysis of memory utilization

File size (kb)	Memory utilization (kb)			
	MRFC [18]	SS-QKD [19]	ECSM-QKDP [20]	MKSS-QKDTWIA
100	66.18	45.29	38.25	25.31
200	147.61	132.33	118.01	83.78
300	200.08	190.47	167.85	101.85

with file size 100kb is 0.14s, whereas the ECSM-QKDP [20] obtains 0.58s. The verification overhead of the MKSS-QKDTWIA is minimized because of the lesser processing time achieved by authentication of multiple key shares with a fixed length and less memory usage based on the definite key length sharing in the cloud.

4.2 Attack analysis

The developed MKSS-QKDTWIA is analysed with inside attack, outside attack and brute force attack for evaluating the efficiency of security.

4.1.1. Inside attack

The sensitive attribute (SA) is divided by $n+1$ groups and each group is encrypted by using a separate group key (GK) based on data owner. The encrypted data $G(SA)$ is uploaded in cloud service provider, once the encryption is done. Therefore, it is difficult to identify GK used in encryption process for a certain group which used to create the robustness against the inside attacks.

Table 6. Analysis of verification overhead

File size (kb)	Verification overhead (s)			
	MR FC [18]	SS-QKD [19]	ECSM-QKDP [20]	MKSS-QKDTWIA
100	0.6 8	0.61	0.58	0.14
200	0.7 7	0.72	0.60	0.21
300	0.8 3	0.75	0.62	0.27
400	1.1	0.82	0.69	0.33

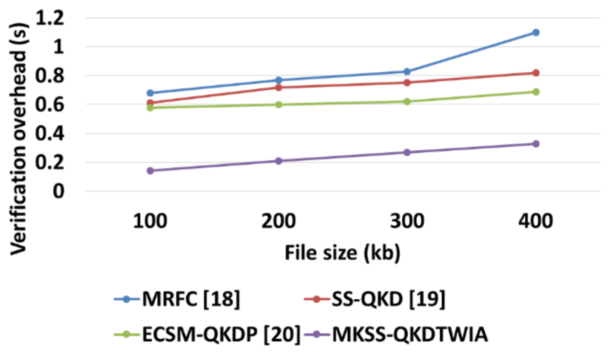

Figure. 7 Comparison of verification overhead

4.1.2. Outside attack

The group admin (GA) accesses the group key based access methodology and it is also possible by gaining the remaining users incorporated in a certain organization. In order to overcome the aforementioned limitation, the $GA(P_r)$ is utilized for generating the GK. Consequently, the P_r of GA is essential while decrypting the certain $G(SA)$, where P_r denotes the private key. Therefore, no unauthorized user can obtain the information about group.

4.1.3. Brute force attack

In brute force attack, two processes such as forging and guessing of group key are accomplished by adversary for discovering the group key and plaintext. The adversary is tried to discover the key in polynomial time for both the above mentioned processes. Next, the adversary tried all different opportunities in a polynomial time in this attack. The key length of MKSS-QKDTWIA is $2^{\{256\times(n+1)\}}$, therefore identification of such larger key in a certain time period is difficult. If an adversary detects any one GK, the other GA prediction is impossible. Because, the computed GK is not dependent on others and it mainly based on data owner and group admin. Therefore, the MKSS-QKDTWIA also has the resistance against the brute force attack.

5. Conclusion

In this paper, effective security to the cloud storage is provided by proposing the MKSS in the QKD. Here, the security is provided between the three major elements of the communication scenario such as data owner (Alice), data user (Bob), Center (Server) and data storage. The key generation efficiency is improved because of using the TPM based MKSS in the QKD for avoiding the errors during the key restoration. The synchronization of the TPMs is used to minimize the operating time. The variation in the registration information each time is utilized for enhancing the robustness against unauthorized users. The MKSS-QKDTWIA with definite key length sharing is used to reduce memory usage during processing. From the results, it is concluded that the MKSS-QKDTWIA provides better performance than the MRFC, SS-QKD and ECSM-QKDP. Further, the MKSS-QKDTWIA provides an effective robustness against the inside attack, outside attack and brute force attack. The key computation time of MKSS-QKDTWIA for 50 attributes set in each key is 16.7s which is less when compared to the MRFC, SS-QKD and ECSM-QKDP. In the future, the authentication protocol can be further improved for better confidentiality and integrity of the data.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

The paper background work, conceptualization, methodology, dataset collection, implementation, result analysis and comparison, preparing and editing draft, visualization have been done by first author. The supervision, review of work and project administration, have been done by second and third author.

References

[1] V. Prabhakaran and A. Kulandasamy, “Integration of recurrent convolutional neural network and optimal encryption scheme for intrusion detection with secure data storage in the cloud”, *Computational Intelligence*, Vol. 37, No. 1, pp. 344-370, 2021.

[2] M. Tajammul and R. Parveen, “Auto encryption algorithm for uploading data on cloud storage”, *International Journal of Information Technology*, Vol. 12, No. 3, pp. 831-837, 2020.

[3] A. S. Babrahem and M. M. Monowar, “Preserving confidentiality and privacy of the

patient's EHR using the OrBAC and AES in cloud environment", *International Journal of Computers and Applications,* Vol 43, No. 1, pp. 50-61, 2021.

[4] D. B. Salvakkam, and R. Pamula, "Design of fully homomorphic multikey encryption scheme for secured cloud access and storage environment", *Journal of Intelligent Information Systems*, pp. 1-23, 2022.

[5] B. Seth, S. Dalal, V. Jaglan, D. N. Le, S. Mohan, and G. Srivastava, "Integrating encryption techniques for secure data storage in the cloud", *Transactions on Emerging Telecommunications Technologies*, Vol. 33, No. 4, p. e4108, 2022.

[6] R. K. Bedi, J. Singh, and S. K. Gupta, "An efficient and secure privacy preserving multi-cloud storage framework for mobile devices", *International Journal of Computers and Applications*, Vol. 43, No. 5, pp. 472-482, 2021.

[7] M. M. Sandoval, M. H. Cabello, H. M. M. Castro, and J. L. G. Compean, "Attribute-based encryption approach for storage, sharing and retrieval of encrypted data in the cloud", *IEEE Access*, Vol. 8, pp. 170101-170116, 2020.

[8] B. P. Kavin, S. Ganapathy, U. Kanimozhi, and A. Kannan, "An enhanced security framework for secured data storage and communications in cloud using ECC, access control and LDSA", *Wireless Personal Communications*, Vol. 115, No. 2, pp. 1107-1135, 2020.

[9] N. Chidambaram, P. Raj, K. Thenmozhi, and R. Amirtharajan, "Advanced framework for highly secure and cloud-based storage of colour images", *IET Image Processing*, Vol. 14, No. 13, pp. 3143-3153, 2020.

[10] H. Li, L. Liu, C. Lan, C. Wang, and H. Guo, "Lattice-based privacy-preserving and forward-secure cloud storage public auditing scheme", *IEEE Access*, Vol. 8, pp. 86797-86809, 2020.

[11] C. Gong, J. Du, Z. Dong, Z. Guo, A. Gani, L. Zhao, and H. Qi, "Grover algorithm-based quantum homomorphic encryption ciphertext retrieval scheme in quantum cloud computing", *Quantum Information Processing*, Vol. 19, No. 3, pp. 1-17, 2020.

[12] M.A. Hossain, and M.A. A. Hasan, "Improving cloud data security through hybrid verification technique based on biometrics and encryption system", *International Journal of Computers and Applications*, Vol. 44, No. 5, pp. 455-464, 2022.

[13] P. Kumar and A. K. Bhatt, "Enhancing multi-tenancy security in the cloud computing using hybrid ECC-based data encryption approach", *IET Communications,* Vol. 14, No. 18, pp. 3212-3222, 2020.

[14] Y. Yao, Z. Zhai, J. Liu, and Z. Li, "Lattice-based key-aggregate (searchable) encryption in cloud storage", *IEEE Access*, Vol. 7, pp. 164544-164555, 2019.

[15] M. Shabbir, F. Ahmad, A. Shabbir, and S. A. Alanazi, "Cognitively managed multi-level authentication for security using Fuzzy Logic based Quantum Key Distribution", *Journal of King Saud University-Computer and Information Sciences,* Vol. 34, No. 4, pp. 1468-1485, 2022.

[16] H. Zhu, C. Wang, and X. Wang, "Quantum fully homomorphic encryption scheme for cloud privacy data based on quantum circuit", *International Journal of Theoretical Physics*, Vol. 60, No. 8, pp. 2961-2975, 2021.

[17] F. Wang, J. Wang, and S. Shi, "Efficient Data Sharing With Privacy Preservation Over Lattices for Secure Cloud Storage", *IEEE Systems Journal*, 2021.

[18] M. Sumathi, and S. Sangeetha, "A group-key-based sensitive attribute protection in cloud storage using modified random Fibonacci cryptography", *Complex & Intelligent Systems*, Vol. 7, No. 4, pp. 1733-1747, 2021.

[19] I. Pedone, A. Atzeni, D. Canavese, and A. Lioy, "Toward a complete software stack to integrate quantum key distribution in a cloud environment", *IEEE Access*, Vol. 9, pp. 115270-115291, 2021.

[20] K Sundar, S. Sasikumar, and C. Jayakumar, "Enhanced cloud security model using QKDP (ECSM-QKDP) for advanced data security over cloud", *Quantum Information Processing*, Vol. 21, No. 3, pp. 1-17, 2022.