Computer Modeling in Engineering & Sciences
DOI:10.32604/cmes.2020.09361
Article

Tech Science Press

An Effective Non-Commutative Encryption Approach with Optimized Genetic Algorithm for Ensuring Data Protection in Cloud Computing

S. Jerald Nirmal Kumar[1,*], S. Ravimaran[2] and M. M. Gowthul Alam[3]

[1]Anna University, Chennai, 600025, India
[2]M.A.M College of Engineering, Trichy, 621105, India
[3]Sethu Institute of Technology, Virudhunagar, 626115, India
*Corresponding Author: S. Jerald Nirmal Kumar. Email: jeraldcse@gmail.com
Received: 06 December 2019; Accepted: 24 August 2020

Abstract: Nowadays, succeeding safe communication and protection-sensitive data from unauthorized access above public networks are the main worries in cloud servers. Hence, to secure both data and keys ensuring secured data storage and access, our proposed work designs a Novel Quantum Key Distribution (QKD) relying upon a non-commutative encryption framework. It makes use of a Novel Quantum Key Distribution approach, which guarantees high level secured data transmission. Along with this, a shared secret is generated using Diffie Hellman (DH) to certify secured key generation at reduced time complexity. Moreover, a non-commutative approach is used, which effectively allows the users to store and access the encrypted data into the cloud server. Also, to prevent data loss or corruption caused by the insiders in the cloud, Optimized Genetic Algorithm (OGA) is utilized, which effectively recovers the data and retrieve it if the missed data without loss. It is then followed with the decryption process as if requested by the user. Thus our proposed framework ensures authentication and paves way for secure data access, with enhanced performance and reduced complexities experienced with the prior works.

Keywords: Cloud computing; quantum key distribution; Diffie Hellman; non-commutative approach; genetic algorithm; particle swarm optimization

1 Introduction

In our day-to-day life, cloud computing is widely used to carry out huge data storage with varied cloud services. Whereas, the Cloud Service Provider (CSP) enables distributed shared services in which the shared resources are accessible to its users on a pay-as-you-go basis. The cloud services are: Platform as a Service (PaaS), Software as a Service (SaaS), and Infrastructure as a Service (IaaS) [1]. Though it offers tremendous storage to its users, there are several barriers to protect the data/information that is stored in the cloud, some of them are: vendor lock-in, reliability, privacy, pricing, interoperability, and the most significant factor to mention, safety [2]. Security is one of the main issues that decrease the espousal of cloud computing.

Organizations that use the cloud for various service designs deployment models are highly suffered from security threats such as insider attacks, eavesdroppers, etc. The threats are categorized based on the security problems experienced by cloud providers and security issues experienced by their clients [3].

The user most agrees that the main issues in this cloud infrastructure are confidentiality, integrity, and authentication. In several years, many authentication schemes have been introduced. In this way, most researchers agree that cloud infrastructure needs fresh methods to deliver security. Public Key Infrastructure (PKI) deployment is currently a major solution [4]. PKI involving the exchange of keys to an authenticated user in the cloud facilities using certificates via government channel. However, there is some problem with PKI authentication where the public key cryptography provides computational security only because PKI is based on Asymmetric Key Cryptography [5]. Cloud computing offers the next generation of internet-based, extremely scalable distributed computing systems offering as a service computing resource [6]. Quantum Key Distribution (QKD) makes use of quantum mechanics to guarantee secure communication. It is used only to produce and distribute a key. A quantum key can be used with any encryption algorithm to encrypt and decrypt a message. QKD is simple to use. It requires fewer resources to maintain it [7]. Quantum Key Distribution (QKD) has characteristics that are unconditionally safe compared to traditional cryptography. However, the conventional QKD suffers from computational complexity [8,9].

To make the QKD key sharing more quickly, photon detectors and time synchronization techniques are updated. Concerning key distribution speed, it is accounted for that the quickest key sharing rate of QKD in a real fiber condition is 304 kbps and adequate for running a safe TV conference application [10]. Be that as it may, when applications demand higher speed correspondence or when numerous applications run at the same time, a genuine deficiency of keys could happen. QKD key sharing is restricted to point-to-point links and the correspondence separation is likewise constrained to many kilometers in light of the trouble of photon identification [11]. A practical approach for conquering the constraint is to organize QKD to hand-off key information. This systems administration approach empowers every hub on the system to trade keys with an arbitrary node on a similar system regardless of the distance. QKD system field preliminaries and research have been directed around the world, defeating the distance constraint [12,13].

From the protocol point of view, standard secure communication frameworks utilize standard protocols, for example, IPsec [14] and Transport Layer Security (TLS) [15] in much of the time. These protocols perform standard open cryptographic key-based key exchanges, for example, RSA or Diffie–Hellman (DH). To coordinate QKD into these standard protocols, the keys produced by QKD ought to be put away and conveniently provided to these protocols.

From the library usage point of view, standard secure communication applications utilize standard library executions, for example, OpenSSL [16] and APIs. To limit the turn of events or adjustment cost of the applications, and indistinguishable API ought to be given to the applications, even for the situation that a safe specialized technique is incorporated with QKD. It makes it simple to port existing secure communication applications created with the OpenSSL library to QKD prepared applications.

From the functionality viewpoint, the QKD key sharing rate restriction and the speed variety relying upon the optical fiber condition ought to be considered [17]. These may cause a lack of keys, wherein case, the applications need to trust that enough keys will be shared and put away

by QKD, bringing about serious handling delays for applications. Here, we look at the QKD key-based secure data storage and standard secure communication. QKD key-based secure data storage gives profoundly secure communication that can be viewed as data hypothetically secure. In any case, the QKD key-based secure communication may cause communication postpones when beginning secure communication for the situation that there is a deficiency of stored keys. On the other hand, standard secure communication performs key sharing (key exchange) for each safe communication without a moment's delay dependent on DH or RSA that isn't viewed as data hypothetically secure. Nonetheless, the standard secure communication forces no critical confinement on key sharing speed. There are practically no handling delays for the applications beginning secure communication. As clarified above, concerning the data hypothetically secure element and the real-time element of secure communication, there is an exchange off between the QKD key-based secure communication and the standard secure communication.

Thus, while on transmitting, storing and retrieving data from the cloud, the cloud fails to ensure security and privacy. Whenever a user uploads the data, attackers such as malware, Trojans, virus, etc., play a vital role in destroying or altering the contents/information in the data, which is uploaded by the user. If the data is successfully uploaded to the cloud without any interruption, there arises some security issues with the third party, who provides access rights to the user whenever he/she requests for the stored data/information. Moreover, intruders such as insider attackers in the cloud may destroy the data and causes data losses too. Hence the researchers focused on some data recovery techniques such as HSDRT, PCS, ERGOT, Linux Box, Cold and Hot back-up technique, SBBR, but the thing is the implementation cost for such techniques is high and complex. Due to these reasons, the cloud has lost its total security and seems to be unreliable to the user. All these security threats lead the cloud storage as an unreliable one with high time and cost complexity resulting in poor performance.

Hence to handle those issues discussed above in this section, innovation is required in the field of cloud computing to ensure security and privacy for the data stored in the cloud. Thus to tackle the above scenarios, this article seeks to propose a novel Quantum Key Distribution relying upon a non-commutative encryption framework to ensure secured data storage and accessing of the varied cloud resources by fixing all those security threats with reduced computation overheads.

The rest of the paper is organized as follows: A detailed review of prior methodologies is discussed in Section 2; Section 3 describes a brief narration about the proposed methodology and Section 4 analyses the performance of the proposed framework and compares its efficiency with the prior techniques to prove its strength against the security threats. Finally, Section 5 concludes the proposed work, which is followed by references.

2 Related Researches

In this section a detailed review about various non-commutative encryption methodologies is discussed in detail as follows:

Kumar et al. [18] introduced an Elliptic Curve Cryptography (ECC) utilized on the Non-Commutative Cryptographic (NCC) standards. The security and qualities of the original copy were flexible on these two cryptographic suppositions. Theoretically, the proposed approach is working. The organization's concerns for applications are on high demand, designing for accelerating the algorithms, likewise in the tremendous of security, is in gigantic demand.

Misra et al. [19] introduced a three-party key agreement protocol utilizing CSP and BDP that opposes all the security dangers and gives key confirmation. It is secure and effective for the

three-party EHR frameworks. Braid groups are stronger than abelian groups for security perspective and yet, simple to design. This one of a kind component makes them appropriate for the EHR framework. The future extent of this paper is extremely rich as it proposes security answers for the most recent issue. The job of cryptography is to keep intruders shocked constantly to give secure communication. The developed nations have made numerous strides for EHR security however in India it is in the child stage. The idea is that the protocol itself gives authentication and key agreement so the wireless exchange of this sensitive data stays secure with no outsider in any event when the correspondence channel is unsecure.

El-Yahyaoui et al. [20] introduced another practical fully homomorphic encryption scheme. It is a symmetric, noise-free and probabilistic cryptosystem, for which the cipher-text space is a non-commutative ring quaternionic based. It is a productive and practical plan whose security depends on the issue of solving an over-characterized arrangement of quadratic multivariate polynomial equations in a non-commutative ring. This cryptosystem isn't execute to confirm its security.

Gu et al. [21] proposed an NTRU-type public key cryptosystem over the binary field, whose security depends on the computational intractability on the DUSPR issue. The proposed NTRU-type public key cryptosystem over Z2 is generally practical. Although it neglects to prove the feasibility and security of digital signature and authentication through leading NTRU-type public key cryptosystem over the binary field.

Bagheri et al. [22] present an NTRU-like public-key cryptosystem dependent on the quaternion algebras and bivariate polynomials. It diminishes the size of the private keys without violating the framework security and utilize larger messages. Thus, message protection is feasible through larger polynomials expanding message security. Also, message protection is possible through larger polynomials and this permits us to acquire a similar security level as other NTRU-like cryptosystems but utilizing lower dimensions.

Accordingly, during the processes of data transmission, storage and accessing the users are forced to fall under several authentications oriented issues, which highly affect the system reliability and confidentiality. Henceforth, to overwhelm all these issues, a need for novel secured methodology is raised to achieve effective data storage as well as access throughout the cloud environment.

3 Novel Quantum Key Distribution Relying Non-Commutative Encryption Framework

In this section, an authentication structure dependent on quantum identity authentication for cloud computing architecture is proposed. A few authentication protocols for cloud computing have been proposed, however, utilizing the QKD authentication is novel. In light of the inviolable laws of quantum mechanics, the proposed protocol is secure as though an eavesdropper watches the information being transmitted will genuinely change the content of a portion of the bits and is in this manner, detectable. The proposed protocol comprises of four stages, for example, Registration, Login, Authentication and data recovery stage.

A Novel Quantum Key Distribution relying on Non-Commutative Encryption framework is designed in this work to tackle the above-mentioned issues experienced with the prior methodologies. It effectively guarantees a high level secured data transmission; this in turn simultaneously reduces the computation as well as communication complexities. For that, a Novel Quantum Key Distribution approach is utilized. The process flow of our proposed framework is exposed with the aid of Fig. 1.

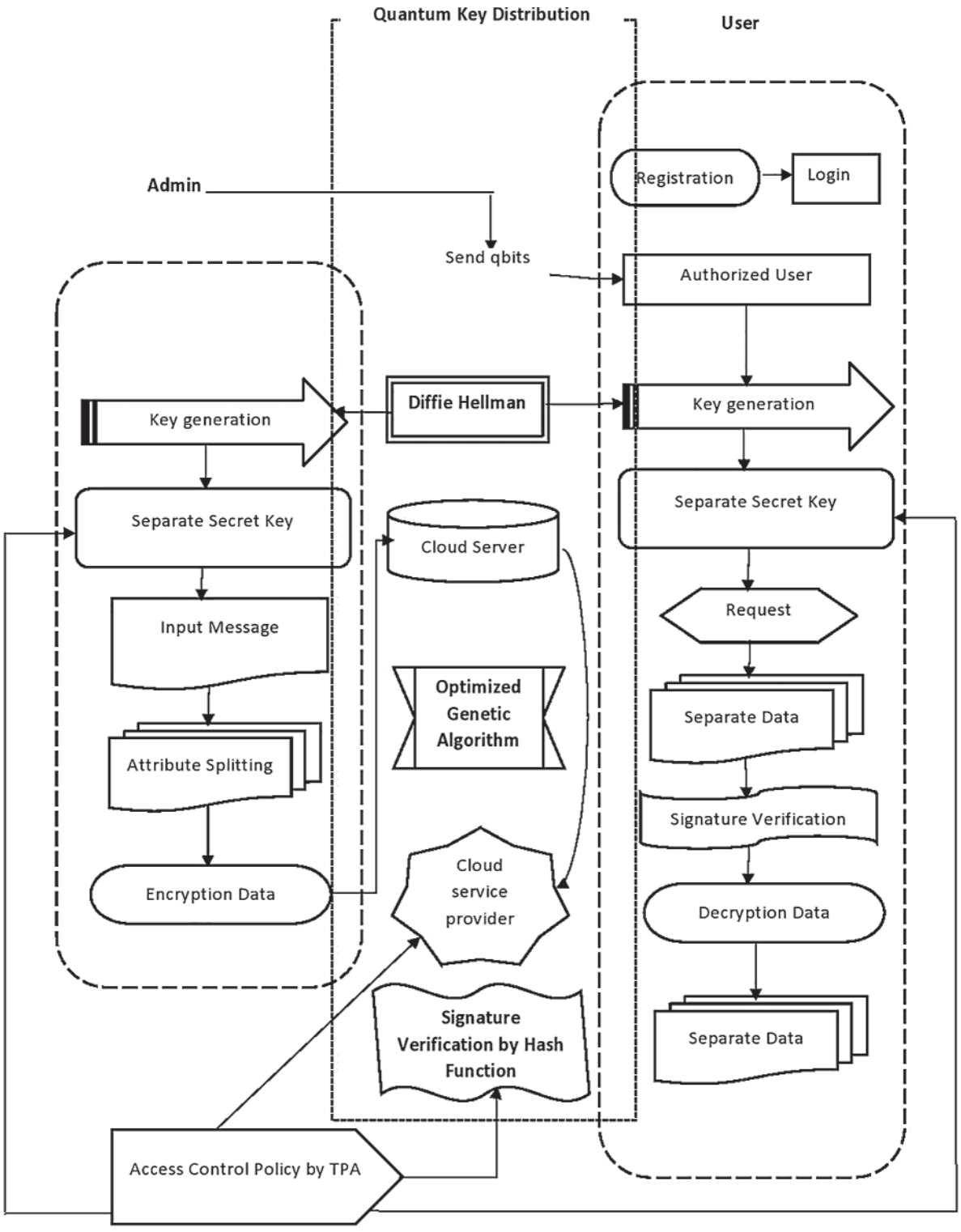

Figure 1: Proposed overall block diagram

3.1 Registration Phase

To register with the Server J_d user R_d sends a unique identification G_d to the cloud server. J_d Creates a unique password Q_d for users R_d after receiving a registration request from the user. Fig. 2 shows the user registration procedure. The steps of the registration procedure are described as follows.

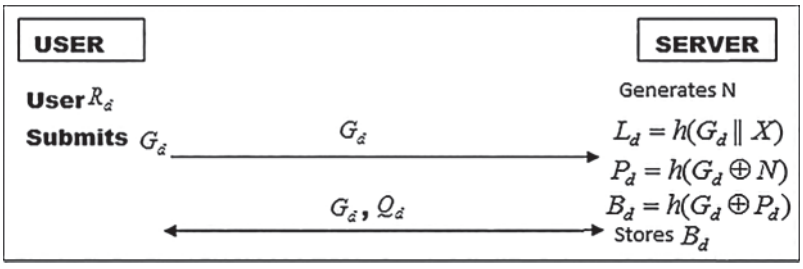

Figure 2: Registration phase

Step 1: User → Server: The user sends his unique identification to the server via a secure channel.

$$R_d \overset{unique\ identification}{\rightarrow} G_d \tag{1}$$

Step 2: After receiving the information, the cloud server J_d computes the parameters of the user as:

$$L_d = h(G_d \| X), P_d = h(G_d \oplus N), B_d = h(G_d \oplus P_d) \tag{2}$$

Step 3: Server → User: The server stores the authentication parameter $\{E_d\}$ and sends a password Q_d to the user via a secure channel.

3.2 Login Phase

The login phase is just the beginning of the experiment. If an unregistered user attempts to sign in, an error occurs. Only authorized user is active G_d and Q_d can log in.

Step 1: The user sends his G_d and Q_d to the server J_d.

Step 2: Server checks G_d and then verifies $h(G_d \oplus P_d)$ is equal to the stored E_d. Once the details are verified, the process then moves on to the verification phase as in Fig. 3. The server rejects the login request if the result does not match the database.

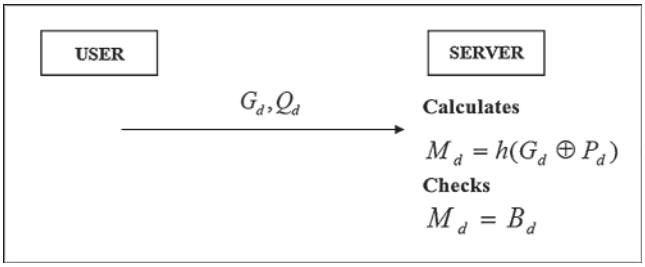

Figure 3: Login phase

3.3 Authentication Phase

After the user successfully logs in to the server, the process moved to the authentication phase. Fig. 5a shows the authentication process. The user is asked to verify that he/she provides all the relevant information and the private user id Q_d generated by sever in the registration section will apply the pre-shared key in the verification section. The cloud server checks the user's validity first by obtaining a DH equivalent to the user id from the server database. When the key matches, the connection is established by this protocol and the user is logged on to the server.

3.4 Key Distribution Using Diffie Hellman with QKD

Current cryptographic network protocols require multiple handshakes between the server and the client to establish specific parameters and session discovery policies. This system allows the client and server to select and adopt various methods and techniques to exchange important information confidentially and securely. Among many other things, one of these agreements involves passing a set of preferred key exchange protocols. These key exchange protocols are used to provide the private keys of remote entities to secretly encrypt their next connection through the interaction of private key algorithms. When transferring, it was agreed to use the first rule-based on both sides, as well as the hash function. One of the most widely used key exchange policies is Diffie–Hellman. Although there are different versions of this protocol, any of them require multiple message exchanges between the two ends. In this way, both ends share some information on the public key reproduction channel (private key).

The key agreement for QKD applies in the same way. When two ends are mentioned, one of them just extract the quantum key and its corresponding key ID into the QKD systems. After that, it transmits that ID (and other important information) over to the open channel and that may be unsafe (public information). This process, similar to the Diffie–Hellman protocol, requires several messages to synchronize keys on both sides of the in and out (text) communications. Therefore, because of these similarities, the integration of the QKD key agreement process in conjunction with the Diffie–Hellman protocol can be obtained directly if the exchange messages are correctly integrated into both processes. Combining these two solutions, Diffie–Hellman messages are expanded and include new parameters, such as quantum key IDs, to protect against other secure sessions.

Fig. 4 shows the exchange of Diffie–Hellman messages, which includes important IDs as a parameter in exchange messages. The workflow is as follows:

- First, the node on the client-side extracts the outgoing communication key from the QKD programs. It can use the standard API or interface [19] or related ones [20].
- Then, in this example, the client sends the key ID to the server.
- The server extracts the IDs and uses it to find the key for its internal channel.
- Similarly, the server extracts its outbound communication key and sends the correct ID to the client in the reply message.
- When the client receives these messages, they use the ID to extract the key of their internal connection.
- Finally, after digesting the generated key using the agreed hash function, the classical key is generated. Both keys are coupled with XOR (module 2) for use together to protect the channel, providing hybrid quantum-classical security.

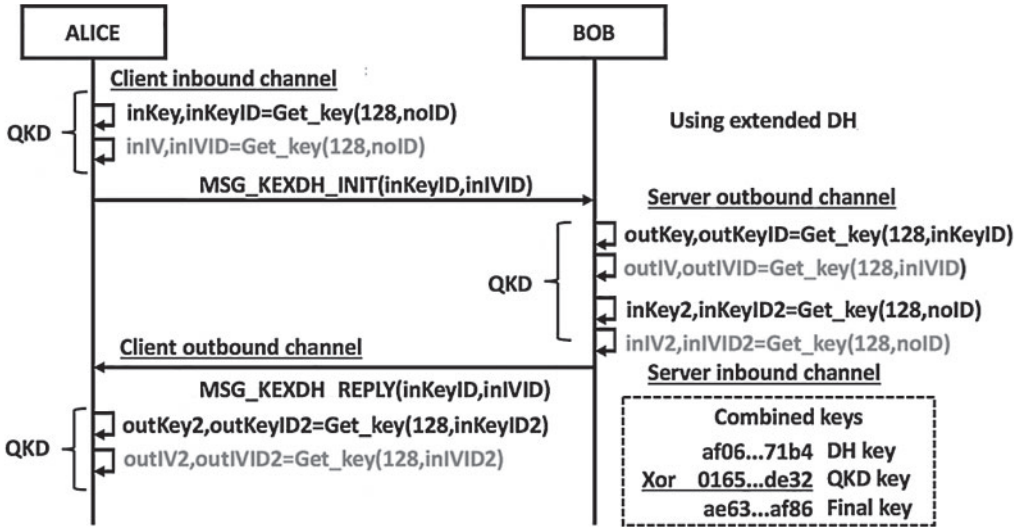

Figure 4: Diffie–Hellman and QKD key exchange protocol integration

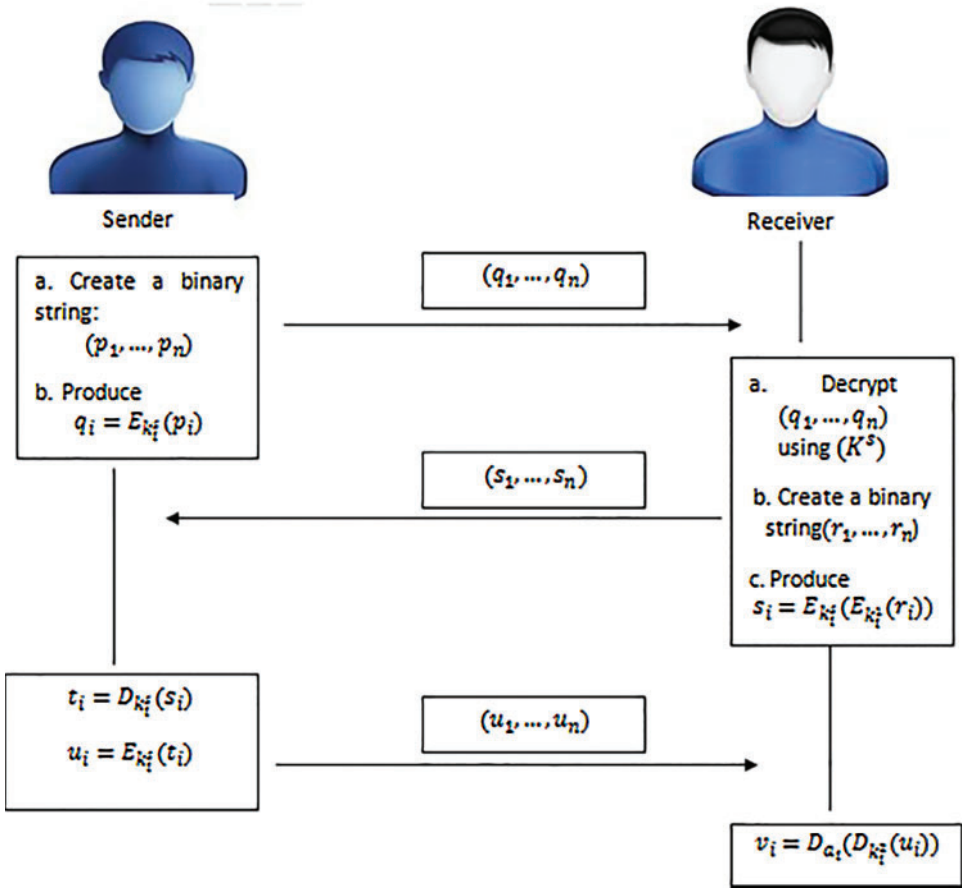

Figure 5a: User authentication procedure

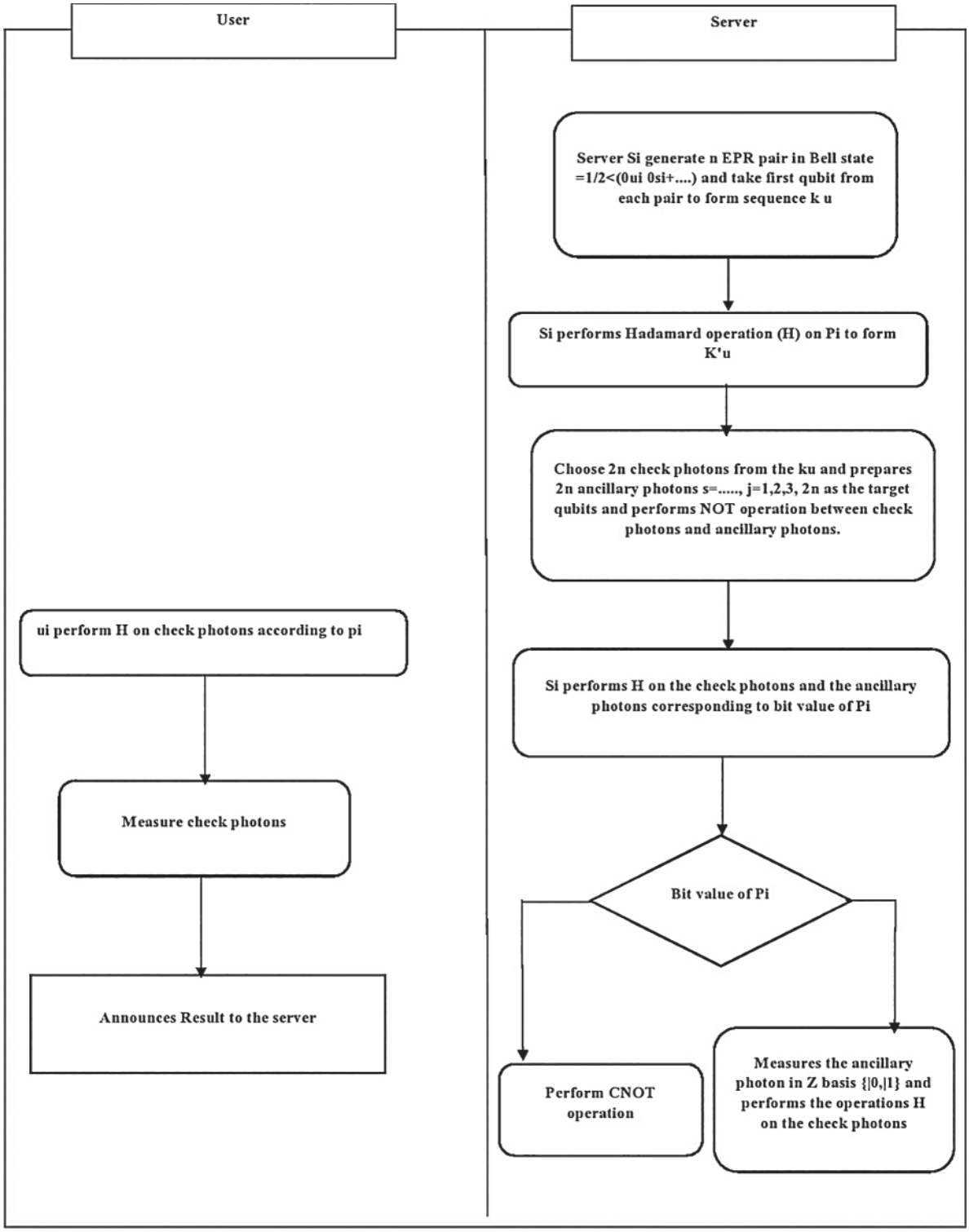

Figure 5b: Key generation after mutual authentication

3.5 Authentication

Furthermore, to tackle the man-in-the-middle attack faced by the conventional quantum cryptography, an authentication scheme has been designed in this work with the aid of the four possible X-gates $\left\{x(0), x\left(\frac{\pi}{2}\right), x(\pi)\ and\ x\left(\frac{3\pi}{2}\right)\right\}$. Authentication is a vital task to secure communication between users. With this scheme, the private keys owned by both the sender $\left(k^1\right)$ as well as the receiver $\left(k^2\right)$ along with the secret shared key $\left(k^3\right)$ provide mutual authentication for the cloud users. To achieve mutual authentication, the scheme undergoes four stages, which are described with the aid of Fig. 5.

At the first stage, the sender creates a binary string (plain text), which is then encrypted by the shared secret key and is sent to the receiver. There at Stage 2, the receiver decrypts the ciphertext and holds the result by its side. It then generates another binary string, says (r_i) and with the aid of shared secret as well as the private key, the string is encrypted and forward to the sender. In Stage 3, the encrypted ciphertext is decrypted by the sender using the shared key. Finally, the ciphertext is decrypted at the receiver side by the private key and the string acquired from Stage 2, $\left(D_{a_i}\right)$ to make clear that if the decrypted plain text (V_i) is the same as that of the text created at Stage 2. Thus, the man-in-the-middle attack experienced by the prior methodology is overwhelmed here with this scheme; whereas, the authentication succeeds only if the plaintexts are alike, i.e., $(V_i = r_i)$.

After achieving the encryption process, the data which is uploaded by the admin is stored in the cloud server. Cloud Service provider is an entity that provides data storage service and computational resources dynamically to the data owner and users. But due to the insider attackers in the cloud, the data get lost which tends to the heavier loss for the data owner and the users. So there will be a need for security in the cloud while storing the data and also if any data could be lost by some attackers, it has to be recovered. Henceforth the proposed technique is adopted which is explained below.

Fig. 5b shows the key generation phase after mutual authentication. After authentication, both the user and the server agree on the session key. The server takes a random number and agrees with the session key. The subsequent communication takes place via text message with the session key generated.

3.6 Repairing of Data by Optimized Genetic Algorithm

Due to probable data exploitation by interior outbreaks i.e., insider attacker adds in clambered data into the encrypted data or even to the encrypted index, the search result can be a fault result. So there is a need for a security mechanism to verify and retrieve the desirable file. The Optimized Genetic Algorithm (OGA) is an advanced file hold-up concept that uses an operative ultra-widely distributed data transmission mechanism and high-speed encryption technology. The backup sequence and Recovery sequence are used in the proposed system. In the Backup sequence, it accepts the data to be backed-up and in Recovery Sequence, when some mischance happens the Cloud Server (constituents of the Optimized Genetic Algorithm) begins the recovery arrangement. Though there are some restrictions such as security issues. To secure these issues, the OGA method performed in Fig. 6.

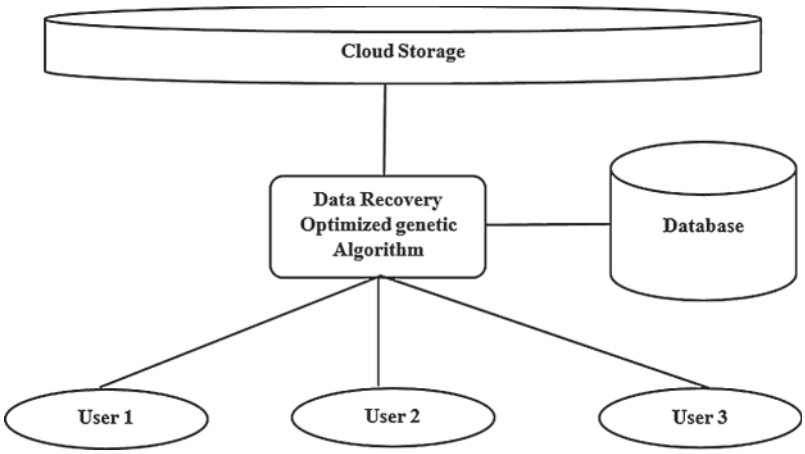

Figure 6: The secure data streaming with optimized genetic algorithm

3.7 Optimized Genetic Algorithm Method

Apply backup GA operation to the top 2N users and create another 2N user.

Step 1 (selection): From the user, select the 2N best user (i.e., non-attackers) according to fitness.

Step 2 (cross over): Using the 2N best users, apply, compare files crossover to update the best 2N users by the following equations.

$$a_i = uniform\,(0,1)\,a_i + (1 - uniform\,(0,1))\,a_{i+1} \quad i = 1, 2, \ldots 2n-1 \tag{3}$$

$$a_i = uniform\,(0,1)\,a_i + (1 - uniform\,(0,1)) \quad i = 2N \tag{4}$$

Step 3: Mutation. Apply mutation with a 20% mutation probability to the best 2N updated chromosomes according to the equation

$$a_k = a_k + rand \times N\,(0,1) \tag{5}$$

Step 4: PSO method. Apply PSO operator (velocity and position updates) for updating the 2N users with the worst fitness.

$$b_{id}^{new} = c \times b_{id}^{new} + h_1 \times rand \times \left(u_{id} - a_{id}^{old}\right) + h_2 \times rand \times \left(u_{id} - a_{id}^{old}\right) \tag{6}$$

Updates. The particles velocities and positions are updated by the following equations

$$a_{id}^{new} = a_{id}^{old} + b_{id}^{new} \tag{7}$$

where h_1 and h_2 are the velocity coefficients, *rand* be the random number in [0, 1] and $c = [0.5 + rand/2.0]$ PSO illustrates that the velocity for each user's particles is updated according to its previous velocity b_{id} the best location in the neighborhood about the particle $u_{id.}$ be the particle velocity. In each dimension is clamped to a max velocity a_{max} and maximum velocity a_{max} is set to a certain fraction of the range of the search space in each dimension. The above final equation shows how each particle position (a_{id}) is updated during the search in the solution space.

Thus with our proposed methodology, the security issues in cloud data storage are successfully overwhelmed with effective quantum key distribution and authentication schemes, which

in turn simultaneously enhances the performance of the proposed system that is discussed in Section 4.

4 Results and Discussion

The proposed framework is implemented using the QKD devices provided by Quantum in the cloud. The experimental setup is completed with the aid of the local Aneka cloud. As the Cloud is a dynamic environment accumulating diverse computing resources, the superior of mounting the intact structure on top of an effective runtime environment. Aneka affords the facility of providing diverse arrangements for articulating distributed applications by posing diverse software development prototypes. Aneka is a form of the PaaS service model and can be deployed in both public as well as private clouds. Here in our work, Aneka is used as a private cloud infrastructure.

4.1 Configuring Private Cloud

The complete process of implementing a private cloud using Aneka is as follows, Aneka Cloud has two main components, Master Container and Worker Containers. In most cases only one container and cloud requirement determines the number of container operations. Worker Containers are registered with Master Container. A large container acts as a gateway to the cloud. The role of a large container is to take a request from consumers, break it down into smaller tasks, and then classify them into Customers depending on the resources available to them.

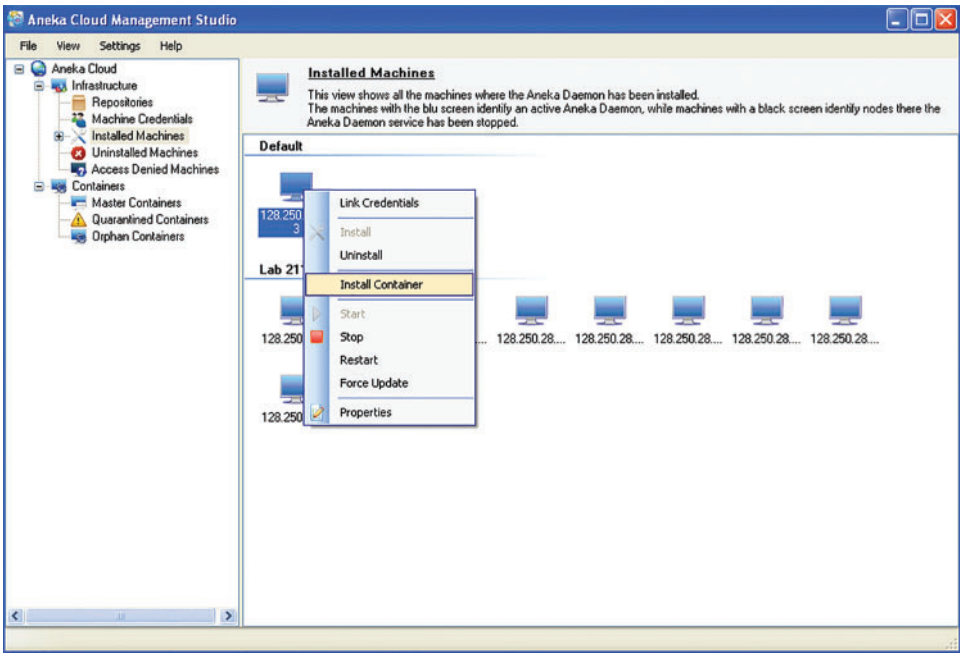

Figure 7: Selecting the node for master container

4.2 Proposed Simulation Results

Aneka is established on the .net platform with the support of Windows OS and all that thing imposts it unique commencing a technology statistic of vision as conflicting to the extensively

accessible. Here user access interface is an Aneka management studio that provides compute and storage services for our research work.

4.2.1 Aneka Cloud Configuration

Initially, the cloud using Aneka is configured by installing the Master Container, in the selected machine as shown in Fig. 7. To create a cloud, it is a difficult task to decide on a system that will work as a Master Container. One must choose a system that can process requests from the expected number of customers, and control the work done by a large number of worker containers.

After the node selection of the master container, the node configuration is performed as in Fig. 8. Because the master node is the primary entry point in the Aneka Cloud. It handles the programming engine and keeps track of all the applications and operations running in the cloud. Without a master node, it is not possible to set up the Aneka cloud.

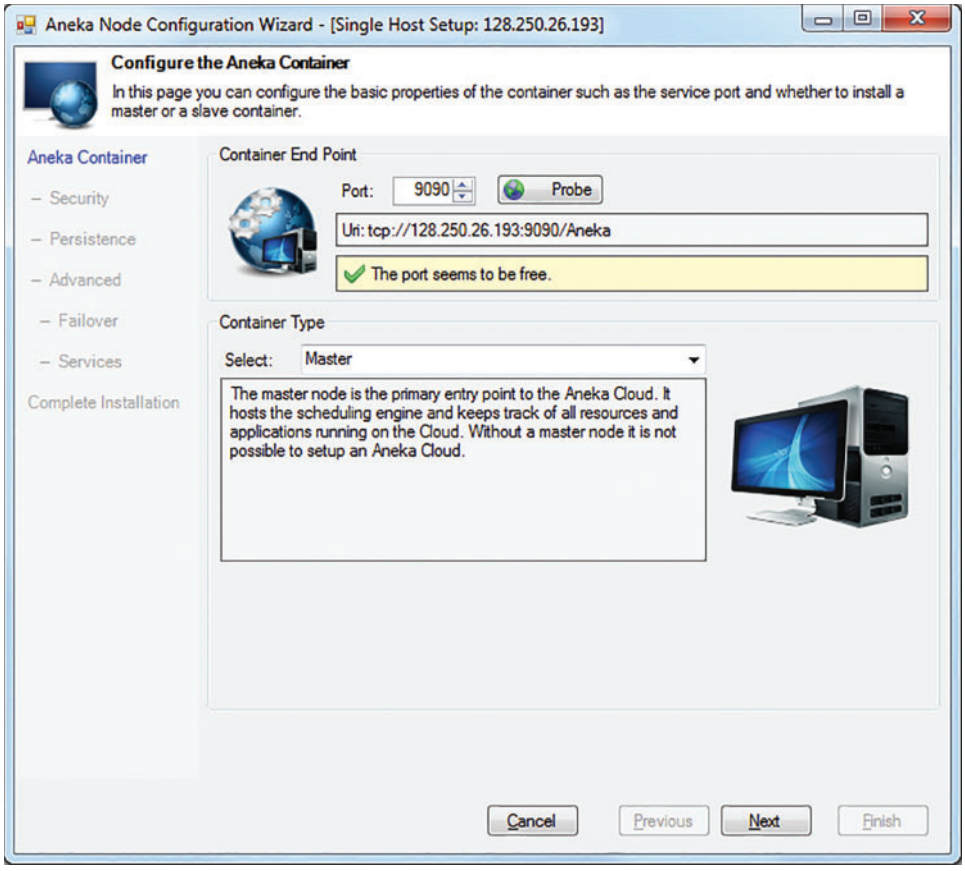

Figure 8: Installation wizard for installing the master

Now the basic installation and configuration of Aneka cloud ends. Then we experimentally verify our proposed framework by following the steps such as Registration, Login, Authentication and data recovery stage. Which are figured out and discussed in the below sections.

4.2.2 Registration Phase

This step is used to provide cloud security. The user sends their unique identifier to the server through a secure channel. After retrieving the information, the cloud server adds user parameters. After that the server stores the authentication parameter and sends the password to the user through a secure channel. Fig. 9 shows the user registration through the Aneka management studio.

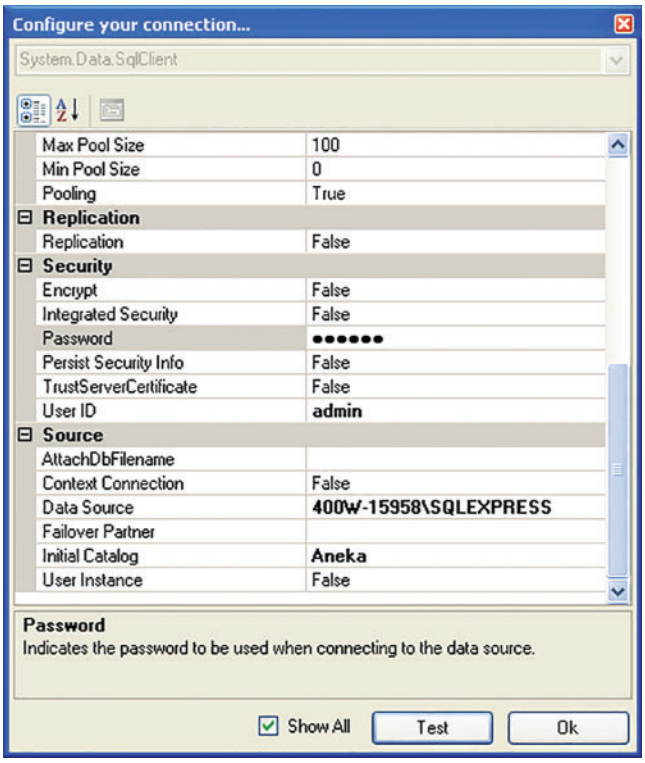

Figure 9: User registration

4.2.3 Login and Authentication with DH and QKD

This step is used to configure cloud security. All users have to authenticate using a valid username and correct password to use any service provided in the cloud as in Fig. 10. Which is experimentally verified with the help of DH and QKD algorithms. It also ensures that the master and the worker confirm each other and all information exchanged between them is encrypted. Ensures data security.

Secret Key is the security key used to ensure cloud security. This is the key that is shared among all authentication containers and encrypted data transmitted between. For a secure cloud to work, it is imperative to generate a secret key, and use it for all security purposes. For secure communication, this button is used by all employees. It is stored in a Key Store shared with some friendly names.

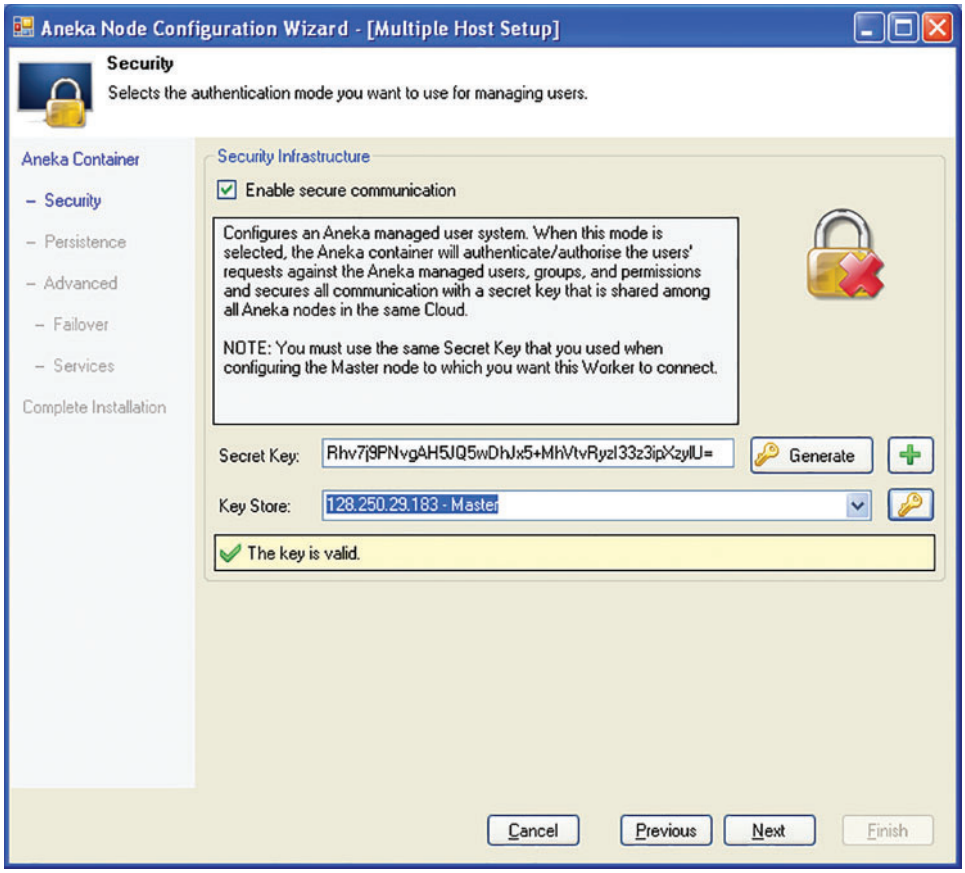

Figure 10: Authentication for user management

4.2.4 Data Recovery Stage with OGA

This process is to configure a large container. There is one big container, and this works like a cloud interface. If this system fails, the cloud stops its function, regardless of the worker's performance at any level. This problem can be solved or minimized in Aneka's cloud with the feature of providing a backup container using failover configuration through OGA. Following the failover configuration step, the cloud can find the primary master and backup master. In the event, the primary master fails, the backup master controls the cloud so that its performance does not stop, as failure reduces the availability of the cloud. To enable failover configuration, the cloud must have a very large feature on its network. The failover configuration with OGA is shown in Fig. 11.

Master container and worker container can be secure by restricting the services configuration as shown in Fig. 12.

Fig. 13 demonstrates the functioning archetypal of the proposed framework with Aneka public cloud. Facilities such as persistence and security are transversal to the intact mound of amenities that are compared by the Container. The contemporary publication of Aneka affords a storing enactment established on the File Transfer Protocol (FTP) service. Further storing amenities be able to incorporate into the arrangement through provide a particular enactment of a data channel. The consumer or sender stores the information by utilizing novel Quantum

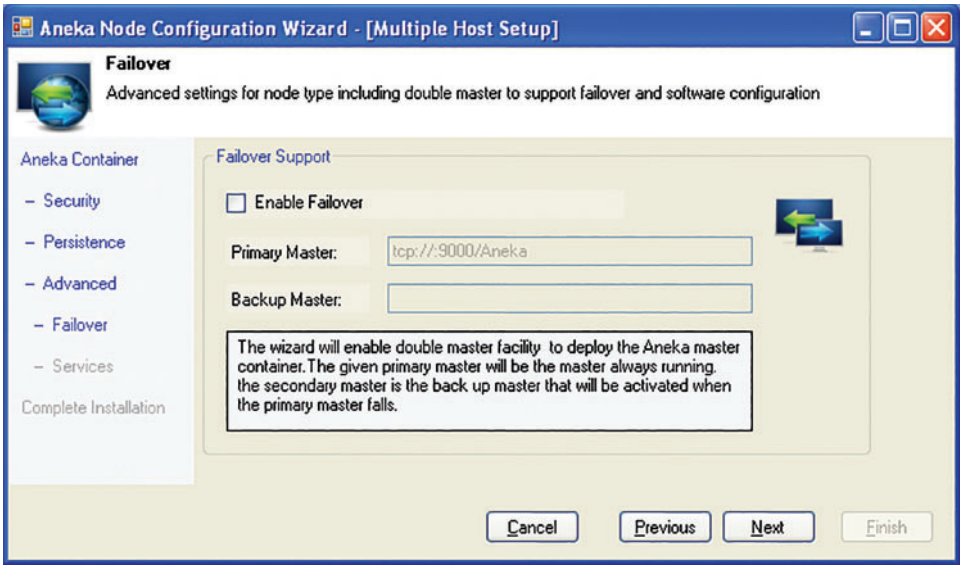

Figure 11: Failover configuration with OGA

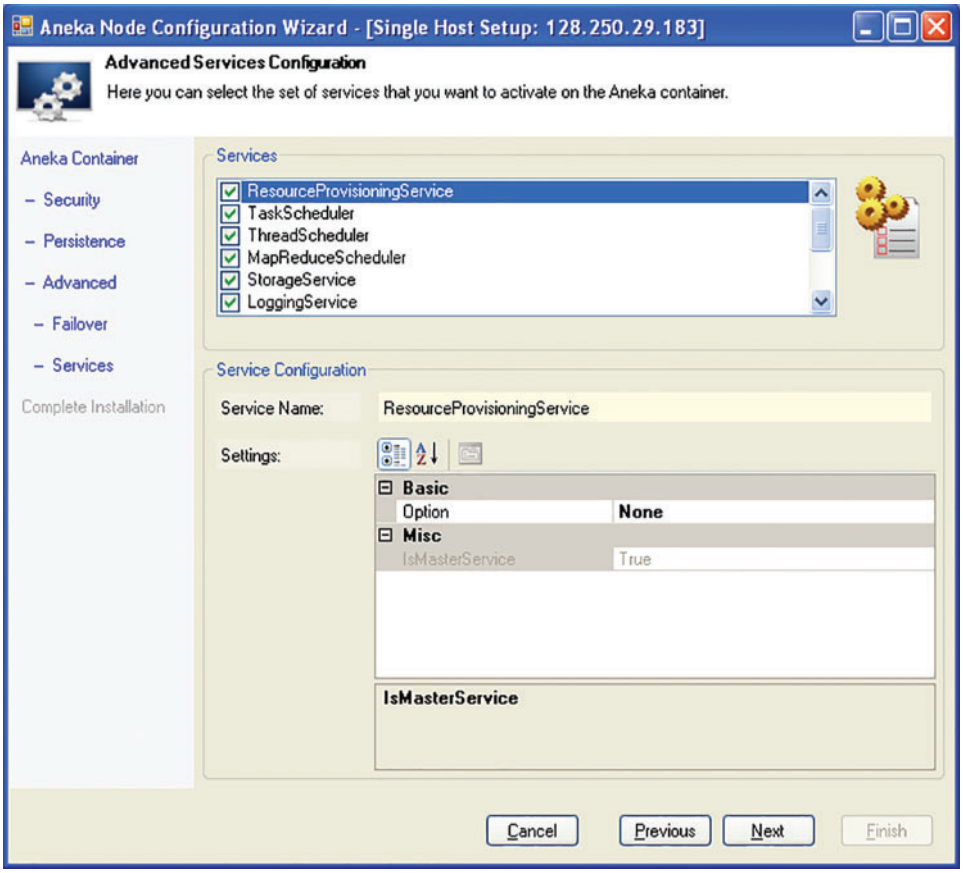

Figure 12: Service configuration for security of resources

Key Distribution relying upon non-commutative encryption and Diffie Hellman (DH) for key distribution. The encoded information was securely stored in the Aneka cloud through admin for secret authentication. Both the sender and receiver can communicate with each other through network communication by sharing their public and private keys. The available resources were utilized Optimized Genetic Algorithm (OGA) for backup and recovery sequence. Aneka delivers an infrastructure able to support a diverse set of storage amenities. It is also elastic for the reason that it is probable to upturn on demand the number of nodes that are part of the Aneka Cloud affording to the user needs. The incorporation of virtual resources into the Aneka Cloud does not familiarize unambiguous challenges: once the virtual resource is attained by Aneka it is solitary essential to have an administrative interpretation and network access to it and organize the Container on it as it takes place for any other physical node. Aneka has effectively contributed to resolving the scalability and security concern tackled to upturn the performing of the requests that influence the Cloud for computation desires.

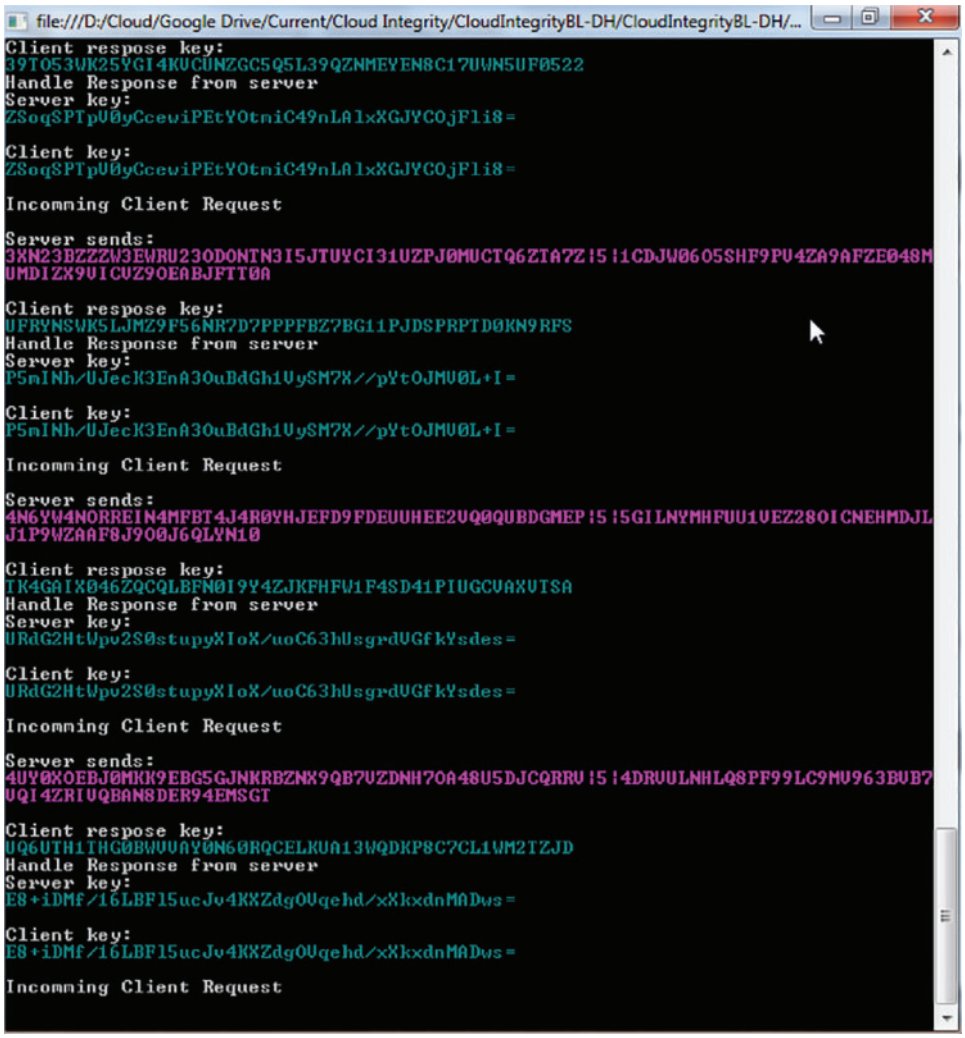

Figure 13: Working of the proposed framework with Aneka public cloud

4.3 Performance Analysis for the Proposed Framework

To show the efficiency of the proposed method, the following factors are evaluated.

4.3.1 Encryption Time

In encryption algorithm conversion of the plain text into the ciphertext takes place within a particular duration which is said to be encryption time. The encryption duration of the data after encrypting it by the projected framework with an optimized Genetic Algorithm technique is related to the encryption procedures.

$$\text{Encryption time} = \frac{C_T}{\mathrm{R_T}} \tag{8}$$

where Encryption time is the, C_T is the computation time of encryption, $\mathrm{R_T}$ is the response time of encryption. It is the time difference between submissions of a request until the response begins to be received.

4.3.2 Decryption Time

In decryption algorithm conversion of the ciphertext into the plain text takes place within a particular duration which is said to be decryption time. The decryption duration of the data after decrypting it by the projected framework is related to the decryption procedures.

$$\text{Decryption time} = \frac{d_T}{\mathrm{s_T}} \tag{9}$$

where Decryption duration, d_T is the computation time of decryption, $\mathrm{s_T}$ is the response time of decryption.

From the Tab. 1, it is known that the encryption time for a proposed framework for different file size File-1, File-2, File-3 and File-4 with the size of 21, 46, 71 and 84 Kb is 293, 302, 312 and 321 ms. The graphical encryption time for the proposed method is given below.

Table 1: Encryption time for the proposed framework

File	File size (kb)	Encryption time (ms)
File-1	21	293
File-2	46	302
File-3	71	312
File-4	84	321

The above Fig. 14 shows the encryption time used in various files in the proposed methodology. Encryption duration is assessed based on the division of computation time to response time. The encrypted text is assessed by the ciphertext policy quality based encryption. As encryption increases the file size also increases.

From Tab. 2, it is known that the decryption time for a proposed framework for different file size File-1, File-2, File-3 and File-4 with the size of 21, 46, 71 and 84 Kb is 289, 295, 312 and 324 ms. The decryption time for the proposed method is given graphically below in Fig. 15.

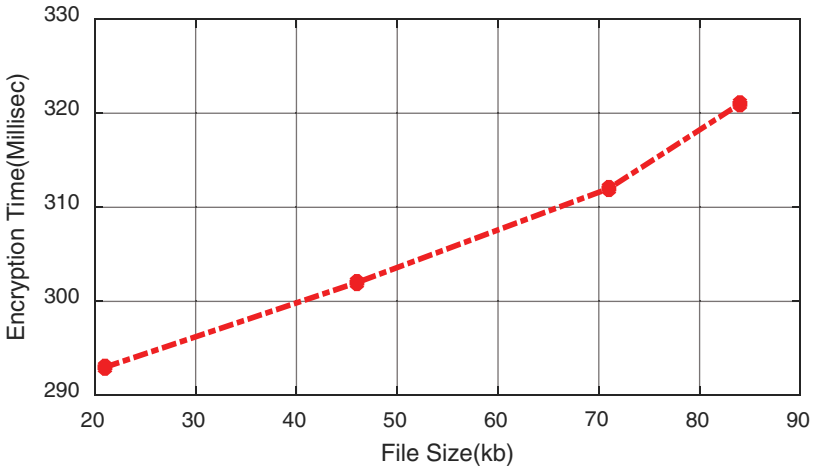

Figure 14: Encryption time

Table 2: Decryption time of proposed framework

File	File size (kb)	Decryption time (ms)
File-1	21	289
File-2	46	295
File-3	71	312
File-4	84	324

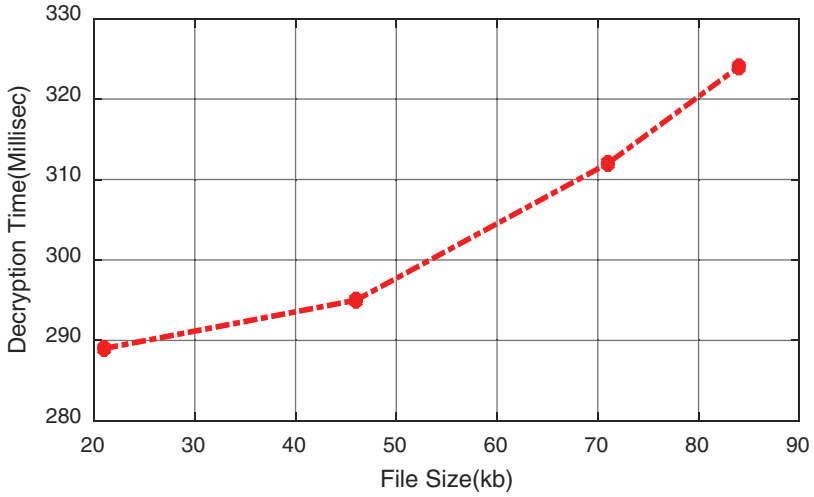

Figure 15: Decryption time of proposed framework

The above Fig. 15 shows the decryption time used in various files in the proposed methodology. Decryption duration is assessed based on the conversion of cipher to plain text in the decryption algorithm. No variation is found in decryption time with file size.

Fig. 16 represents the error probability that occurs by the influence of eavesdropper in the transmission channel.

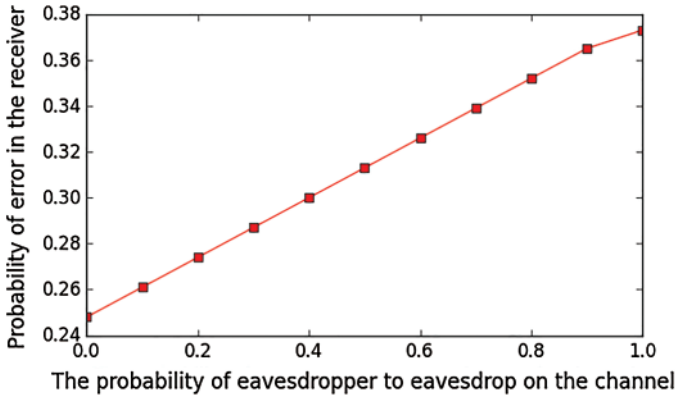

Figure 16: Rate of error observed over eavesdropping

Fig. 17 which represents how the channel overcomes the eavesdropping issue in the channel with different probabilities. Here, the purple line, green line, and the red line represents the different possible probability of the attackers. By this the probability of the error can be detected effectively whereas the eavesdropper has been detected effectively as the number of transmitted bits is rising; thus the proposed work effectively tackles the security issues during various cloud services and strengthens the data storage and its access in a highly secured manner.

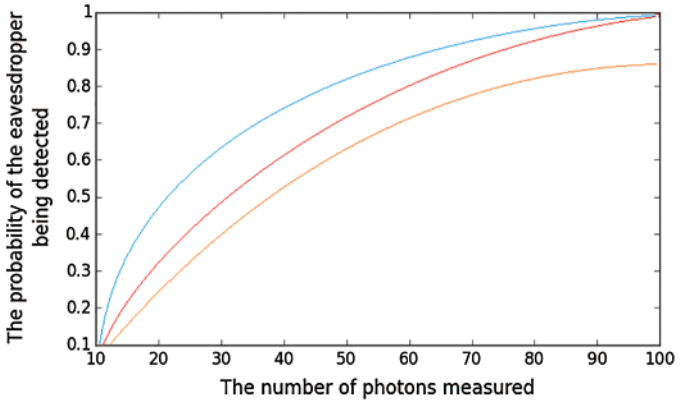

Figure 17: Probability of eavesdropping perceive the channel

As shown in Fig. 18, the initial bit error rate has been slightly reduced with the proposed framework. Moreover, the secure key rate S_k is given by,

$$S_k = S_{kr}\{1 - (1 + g(x) f(x))\} \tag{10}$$

where, S_{kr} be the shifted key rate, $g(x)$ and $f(x)$ be the binary entropy functions respectively. During the error rate estimation, the statistical error has been highly reduced by setting a certain interval of time for each transmission distance. Tab. 3 demonstrates the obtained result with the

aid of the above equation. Here, secure key rates are calculated using the sifted key rates, error rates and double click fraction that we obtained experimentally.

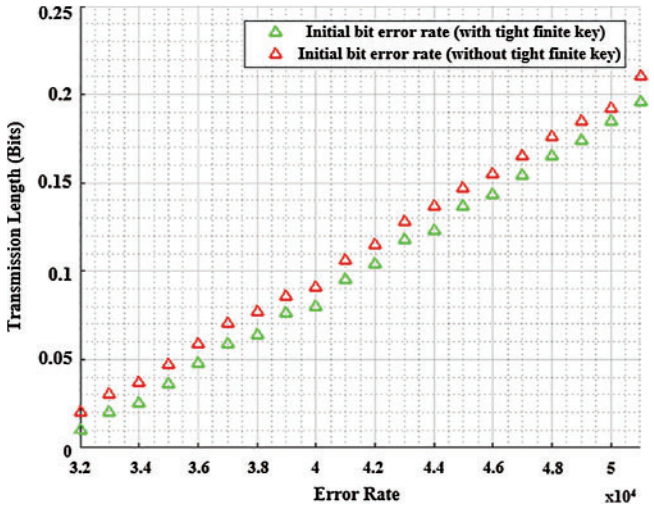

Figure 18: Proposed initial bit error rate

Table 3: Mode analysis of network

Distance (loss [in dB])	0	17	40	85
Effective measurement time [s]	245	230	210	340
Average single count rate [kcps]	50	28	23	7
Total sifted key [bits]	13230	15300	8400	9200
Sifted key rate [bit/s]	460	340	230	56
Error rate	0.012	0.043	0.056	0.052
Secure key rate [bit/s]	230	203	134	43
Run time (M cycles)	406	753	2200	1502
Average latency (ms)	603	1792	17452	700

Here, Tab. 3 describes the mode analysis of the network in our proposed work. For 0 dB distance loss, the obtained error rate is 0.012, sifted key rate is 460 bits/s, secure key rate is 230 bits/s, the total sifted key rate is 13230 bits and so on. Thus, from this table, it is clear that the proposed framework acts as an effective barrier against various attacks and provides secured data storage and its access by handling the security threats in the cloud service provider.

Thus, it is clear from the above analysis that the quantum cryptography provides unconditional security and the sniffing detection properties to achieve secured data transmission and access control.

4.4 Comparison Result

The encryption and decryption duration for the novel Quantum Key Distribution relying on Non-Commutative Encryption Framework is compared with the various existing methods discussed in Section 2. The comparative results are shown in the following Tab. 4.

Table 4: Comparison of proposed and existing technique with different parameter

References	Encryption time (ms)	Decryption time (ms)	Time complexity (ms)	Run time (ms)	Average latency (ms)
Mishra et al. [9]	389	367	556	567	876
Dinesha et al. [23]	384	360	544	546	786
Li et al. [24]	375	356	531	460	770
Zukarnain et al. [5]	356	350	506	480	700
Proposed	307	305	504	406	603

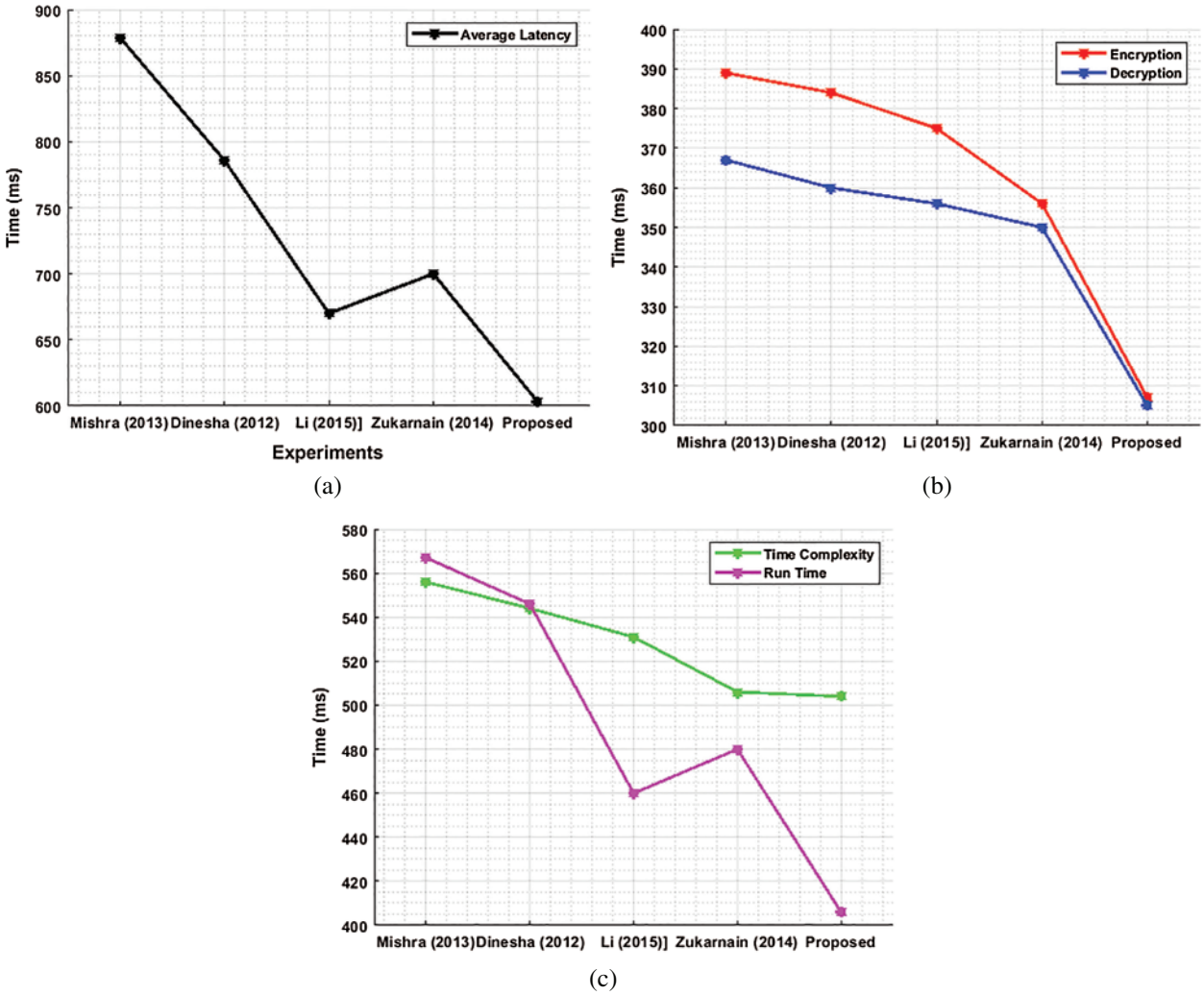

Figure 19: Proposed comparison with prior methodologies. (a) Average latency. (b) Encryption and decryption time. (c) Time complexity and run time

Tab. 4 describes the time complexity is the computational complexity, which denotes the amount of time it takes to run an algorithm. By comparing the proposed framework with those prior methodologies, the proposed framework works best in providing secured cloud data storage. The prior methodologies discussed in Mishra et al. [9], Dinesha et al. [23], Li et al. [24] and Zukarnain et al. [5] achieve encryption within 389 (ms), 384 (ms), 375 (ms) and 356 (ms) respectively, whereas the proposed work achieves encryption within 307 (ms). Thus the time taken to encrypt the data is comparatively low when compared with the existing methodologies.

In the above Fig. 19 describes (a) Average Latency, (b) Encryption and Decryption Time (c) Time complexity and Run Time moreover, the time complexity is also highly reduced to 504 (ms), where the prior methodologies exhibit 556 (ms), 544 (ms), 531 (ms), and 506 (ms) respectively for the methodologies discussed in Mishra et al. [9], Dinesha et al. [23], Li et al. [24] and Zukarnain et al. [5]. It shows that with the proposed work the time complexity is also highly reduced.

Fig. 20 describes that there is a 100 percent failure rate with public cloud implementations. The success rate for achieving effective main distribution is 98 percent. Another observation with our experiments is that the generation of keys with QKD devices given by the "Quantum in Cloud" platform is 100 percent efficient. Thus, from the aforementioned results and comparison results, it is clear that the proposed framework exhibits better efficacy in providing data security with less computational time and cost.

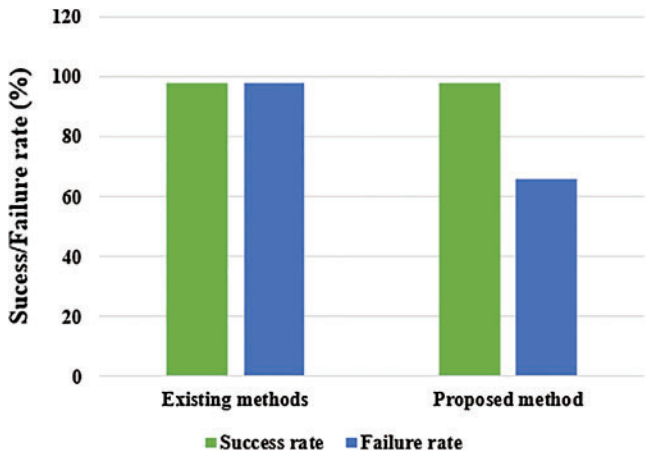

Figure 20: Comparison in terms of success/failure rate and key generation efficiency

Tab. 5 shows the throughput comparison values of some existing methodologies with the proposed technique. Which is figured out in below Fig. 21. The experiment is conducted for different encryption techniques with different file sizes. From these, we analyze that our proposed methodology provides a maximum average throughput of 0.3449, which is higher than other existing methodologies.

Tab. 6 shows the Avalanche effect values with different file sizes for some existing methodologies with the proposed technique. Which is figured out in below Fig. 22. This shows that our proposed methodology achieves an efficient avalanche value when there is a 1-bit change in its key value. Which ensures the security of our proposed technique in the Aneka cloud environment.

Table 5: Throughput comparison of proposed and existing methodology with different file sizes

File size	100	200	300	400	500	600	Average
DESC	0.6971	0.3537	0.2074	0.1272	0.0924	0.0666	0.2574
AESC	0.7136	0.35028	0.2027	0.1191	0.0881	0.0410	0.2525
Blowfish	0.7338	0.4208	0.2894	0.1411	0.1177	0.0738	0.3051
Proposed	0.8215	0.5320	0.3032	0.1613	0.1563	0.0954	0.3449

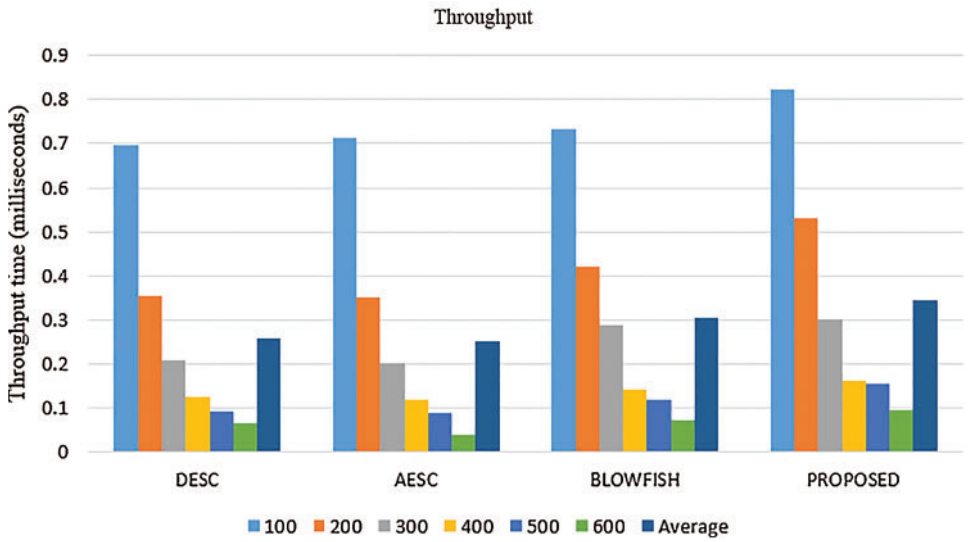

Figure 21: Throughput comparison with different file sizes

Table 6: Avalanche effect with different file sizes when 1-bit change in key

File size	100	200	300	400	500	600
DESC	46.26	46.28	46.37	46.76	46.22	46.22
AESC	46.41	46.18	46.5	46.56	46.19	53.85
Blowfish	62.12	62.15	62.13	62.13	62.1	62.12
Proposed	64.16	64.12	64.25	64.27	64.11	64.59

Tab. 7 demonstrates the Avalanche effect values when there is a bit of change occur in plain text. Values for the different methodologies with different file sizes are drawn graphically in below Fig. 23. From these, we analyze that our proposed methodology provides different values when there is a little bit of change in plain text. All these comparisons show the effectiveness and quickness of our proposed technique, which is sufficiently higher than the other existing algorithms and ensures the security of our proposed technique in the Aneka cloud environment.

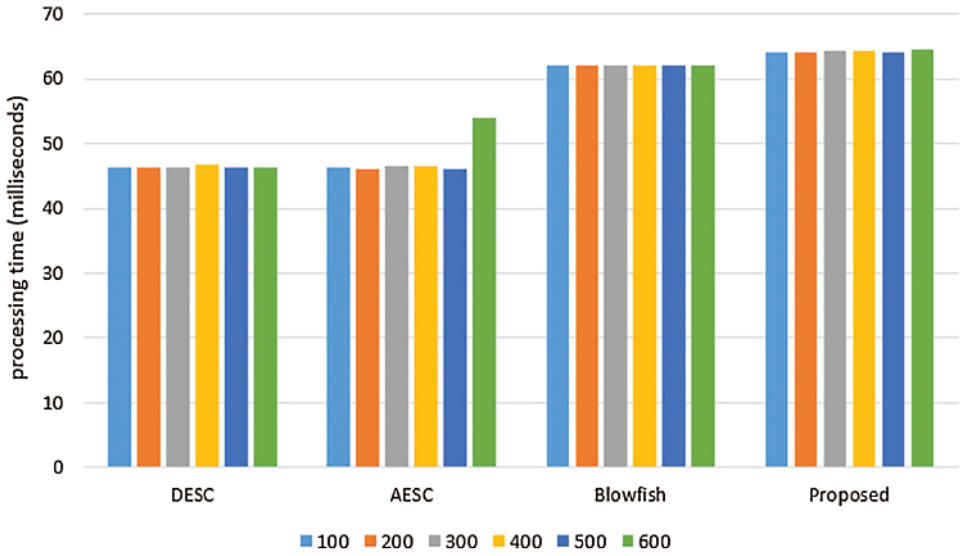

Figure 22: Avalanche effect comparison with different file sizes when 1-bit change in key

Table 7: Avalanche effect with different file sizes when 1-bit change in plaintext

File size	100	200	300	400	500	600
DESC	50.14	50.15	50.13	46.67	50.12	50.15
AESC	47.19	48.15	48.05	48.1	48.15	48.2
Blowfish	46.16	46.21	46.62	46.06	49.9	57.01
Proposed	56.94	57.15	56.13	56.67	57.14	58.17

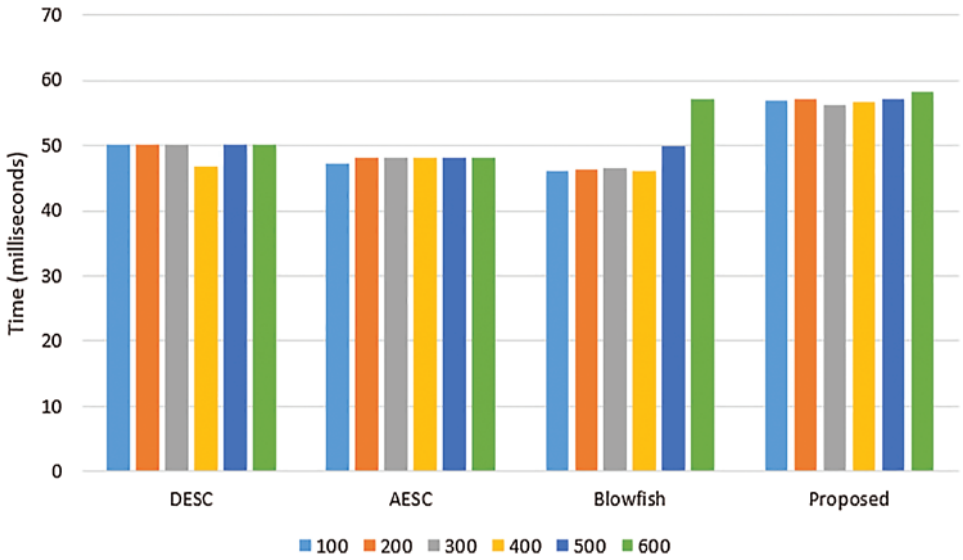

Figure 23: Avalanche effect with different file sizes when 1-bit change in plaintext

5 Conclusion

In this work, a novel Quantum Key Distribution relying on a non-commutative encryption framework is proposed. In which the files are uploaded by the admin, whereas authentication for an authorized user by novel DH technique prevents the influence of man-in-the-middle attack as well as the eavesdropper in the transmission channel. Moreover, for secure data retrieval, OGA has utilized in our proposed work. Thus with this proposed work, the security threats are highly arrested, which ensures a secured data storage and transmission with reduced computational time and cost. To show the efficacy of the proposed work, the framework is compared with the prior methodologies.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Dinh, H. T., Lee, C., Niyato, D., Wang, P. (2013). A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless Communications and Mobile Computing, 13(18),* 1587–1611. DOI 10.1002/wcm.1203.
2. Nedelcu, B., Stefanet, M. E., Tamasescu, I. F., Tintoiu, S. E., Vezeanu, A. (2015). Cloud computing and its challenges and benefits in the bank system. *Database Systems Journal, 6(1),* 44–58.
3. Suthar, F., Khanna, S., Patel, J. (2019). A survey on cloud security issues. *International Journal of Computational Science and Engineering, 7,* 120–123.
4. Basin, D., Cremers, C., Kim, T. H. J., Perrig, A., Sasse, R. et al. (2016). Design, analysis, and implementation of ARPKI: an attack-resilient public-key infrastructure. *IEEE Transactions on Dependable and Secure Computing, 15(3),* 393–408. DOI 10.1109/TDSC.2016.2601610.
5. Zukarnain, Z. A., Khalid, R. (2014). Quantum key distribution approach for cloud authentication: enhance tight finite key. *International Conference on Computer Science and Information Systems*, pp. 28–33. Dubai (UAE).
6. Almorsy, M., Grundy, J., Müller, I. (2016). An analysis of the cloud computing security problem. arXiv preprint arXiv: 1609.01107.
7. Broadbent, A., Schaffner, C. (2016). Quantum cryptography beyond quantum key distribution. *Designs, Codes and Cryptography, 78(1),* 351–382. DOI 10.1007/s10623-015-0157-4.
8. Bennett, C. H., Brassard, G. (2020). Quantum cryptography: public key distribution and coin tossing. *International Conference on Computers, Systems & Signal Processing*, Vol. 560(Part 1), pp. 7–11. Bangalore, India, arXiv preprint arXiv: 2003.06557.
9. Mishra, D., Kumar, V., Mukhopadhyay, S. (2013). A pairing-free identity based authentication framework for cloud computing. *International Conference on Network and System Security*, pp. 721–727. Berlin, Heidelberg: Springer.
10. Sasaki, M., Fujiwara, M., Ishizuka, H., Klaus, W., Wakui, K. et al. (2011). Field test of quantum key distribution in the Tokyo QKD network. *Optics Express, 19(11),* 10387–10409. DOI 10.1364/OE.19.010387.
11. Tysowski, P. K., Ling, X., Lütkenhaus, N., Mosca, M. (2018). The engineering of a scalable multi-site communications system utilizing quantum key distribution (QKD). *Quantum Science and Technology, 3(2),* 024001.
12. Tanizawa, Y., Takahashi, R., Sato, H., Dixon, A. R. (2017). An approach to integrate quantum key distribution technology into standard secure communication applications. *Ninth International Conference on Ubiquitous and Future Networks*, pp. 880–886. Milan, Italy: IEEE.

13. Polnik, M., Mazzarella, L., Di Carlo, M., Oi, D. K. L., Riccardi, A. et al. (2020). Scheduling of space to ground quantum key distribution. *EPJ Quantum Technology, 7(1),* 802. DOI 10.1140/epjqt/s40507-020-0079-6.
14. Aboba, B., Simon, D., Arkko, J., Eronen, P., Levkowetz, H. (2005). Extensible authentication protocol (EAP) key management framework. *Work in Progress*.
15. de Fuentes, J. M., Hernandez-Encinas, L., Ribagorda, A. (2018). Security protocols for networks and internet: a global vision. In: Daimi, K., (Eds.), *Computer and network security essentials*. pp. 135–151. Cham: Springer.
16. Viega, J., Messier, M., Chandra, P. (2002). *Network security with openSSL: cryptography for secure communications*. O'Reilly Media, Inc.
17. Peev, M., Pacher, C., Alléaume, R., Barreiro, C., Bouda, J. et al. (2009). The SECOQC quantum key distribution network in Vienna. *New Journal of Physics, 11(7),* 075001. DOI 10.1088/1367-2630/11/7/075001.
18. Kumar, G., Saini, H. (2020). An ECC with probable secure and efficient approach on noncommutative cryptography. In: Jain, L., Tsihrintzis, G., Balas, V., Sharma, D., (Eds.), *Data Communication and Networks*. pp. 1–11. Singapore: Springer.
19. Misra, M. K., Chaturvedi, A., Tripathi, S. P., Shukla, V. (2019). A unique key sharing protocol among three users using non-commutative group for electronic health record system. *Journal of Discrete Mathematical Sciences and Cryptography, 22(8),* 1435–1451. DOI 10.1080/09720529.2019.1692450.
20. El-Yahyaoui, A., El Kettani, M. D. E. C. (2017). A verifiable fully homomorphic encryption scheme to secure big data in cloud computing. *International Conference on Wireless Networks and Mobile Communications*, pp. 1–5. Rabat, Morocco: IEEE.
21. Gu, Y., Xie, X., Gu, C. (2019). A new NTRU-type public-key cryptosystem over the binary field. *Computers, Materials & Continua, 60(1),* 305–316.
22. Bagheri, K., Sadeghi, M. R., Panario, D. (2018). A non-commutative cryptosystem based on quaternion algebras. *Designs, Codes and Cryptography, 86(10),* 2345–2377. DOI 10.1007/s10623-017-0451-4.
23. Dinesha, H. A., Agrawal, V. K. (2012). Multi-level authentication technique for accessing cloud services. *International Conference on Computing, Communication and Applications*, pp. 1–14. Dindigul, Tamilnadu, India: IEEE.
24. Li, X., Li, W., Shi, D. (2015). Enterprise private cloud file encryption system based on tripartite secret key protocol. *International Industrial Informatics and Computer Engineering Conference*, pp. 166–169. Atlantis Press.