Asian Journal of Information Technology 15 (3): 578-592, 2016
ISSN: 1682-3915

Quantum Cryptographic Approach to Decentralized Access Control and Privacy Preserving in Cloud

K. Vidya, V.S. Alamelu, K. Sashi Kumar and L. Sree Chandraa
Department of Information Science and Technology, Anna University,
600025 Chennai, Tamil Nadu, India

Abstract: Security is one of the leading concerns in the distributed system because integration of different components leads to new security issues. Handling the threats and vulnerabilities in the distributed environment is important. The biggest concerns of storing the data in a distributed environment like cloud include reliability, security, privacy preservation of the user. The attribute based encryption techniques enforce strict access permission to the files stored in cloud. There are multiple key distributing authorities because it is not possible for a single authority to distribute keys to all the users of the cloud. The decentralised approach also tackles the problem of single point of failure. Privacy of the users can also be preserved using the attribute based techniques where authentication of the users is done by using the attributes instead of their unique user identities. The existing classical cryptographic system suffers from various attacks where the cloud can tamper the data or an interceptor may sniff the keys. Almost, all the encryption techniques are prone to attacks and hence, an additional layer of security is provided to the classical cryptographic system by using the quantum cryptographic techniques. This makes the entire system resilient towards man-in-the-middle attack, user revocation attack and preserves the data from being tampered by the cloud. The quantum cryptographic techniques increase the time and the heap space occupied during encryption and decryption by an insignificant amount but improves the security vastly by tackling various attacks. The proposed system enhances the security of the existing system while there is an insignificant difference in the performance of the two systems.

Key words: Cryptographic, access control, cloud, reliability, revocation attack

INTRODUCTION

Cloud computing is a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications. There are three cloud computing deployment models. Public clouds are owned and operated by companies that use them to offer rapid access to affordable computing resources to other organisations or individuals. A private cloud is owned and operated by a single company that controls the way virtualised resources and automated services are customised and used by various lines of business and constituent groups. A hybrid cloud uses a private cloud foundation combined with the strategic use of public cloud services. Most of the data stored in cloud is highly sensitive, say, hospital patient record maintenance. Therefore, security and privacy are very important issues in cloud computing (Shaikh and Haider, 2011). The user should authenticate itself before initiating any transaction. It must be ensured that the cloud does not tamper with the data. User privacy is required so that the cloud or other users do not know the identity of the user ensuring user anonymity. Access control in clouds is gaining attention because it is important that only authorized users have access to valid service. There are broadly three types of access control-User-Based Access Control (UBAC), Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC). In UBAC, the access control list contains the list of users who are authorized to access data. This is not feasible in clouds where there are many users. In RBAC users are classified based on their individual roles (Kuhn *et al*., 2010). Data can be accessed by users who have matching roles which are defined by the security of the system. In the ABAC, the users are given attributes and the data has attached access policy. The ABAC is used in this work and the attributes to the users are distributed by the Key Distribution Centre (KDC). To ensure decentralized access control (Ruj *et al*., 2011) multiple KDCs are used. A single KDC is not only a single point of failure but difficult to maintain because of the large number of users that are supported in a cloud

Corresponding Author: K. Vidya, Department of Information Science and Technology, Anna University, 600025 Chennai, Tamil Nadu, India

environment. So, it is emphasized that clouds should take a decentralized approach while distributing secret keys and attributes to users. A trusted party can be used for the purpose of user registration. The users may create, read or write a file to the cloud after registration with the trustee and after receiving secret keys corresponding to their attributes.

Privacy preservation of the user and security of the data stored in the cloud is also achieved using Attribute Based Encryption (ABE) which defines the identity of a user as a set of attributes, e.g., roles and messages can be encrypted with respect to subsets of attributes (key-policy ABE-KP-ABE) or policies defined over a set of attributes (ciphertext-policy ABE-CP-ABE). The key issue is that someone should only be able to decrypt a ciphertext if the person holds a key for "matching attributes" where user keys are always issued by some trusted party. Here, CP-ABE is used where a user's private-key is associated with a set of attributes and a ciphertext specifies an access policy over a defined universe of attributes within the system. A user will be able to decrypt a ciphertext if and only if his attributes satisfy the policy of the respective ciphertext. Many private key cryptosystems have some severe problems in various contexts. The most basic problem is the key distribution. A malevolent third party may be eavesdropping on the key distribution and then use the intercepted key to decrypt some of the message transmission. One of the earliest discoveries in quantum computation and quantum information was that quantum mechanics can be used to do key distribution in such a way that Alice and Bob's security cannot be compromised. This procedure is known as quantum cryptography or Quantum Key Distribution. The basic idea is to exploit the quantum mechanical principle that observation in general disturbs the system being observed. In the classical world the bits either 0 or 1 can be copied. Bits can be observed without changing them. So eavesdropping cannot be detected in classical cryptosystems. A quantum bit (qubit) can be 0 or 1 at the same time. Its state will collapse if it is observed. Quantum cryptography makes use of the subtle properties of quantum mechanics such as the quantum no-cloning theorem and the Heisenberg uncertainty principle (Sharbaf, 2009). According to the Heisenberg's uncertainty principle, measuring a quantum system in general disturbs it and yields incomplete information about its state before the measurement. It cannot be copied as specified by no cloning theorem. Eavesdropping on a quantum communication channel therefore causes an unavoidable disturbance, alerting the legitimate users. This yields a cryptographic system for the distribution of a secret cryptographic key between two parties, initially sharing no secret information. Once this secret key is established, it can be used together with classical cryptographic techniques. The goal of quantum cryptography is to perform tasks that are impossible or intractable with conventional cryptography. Unlike conventional cryptography whose security is often based on unproven computational assumptions, quantum cryptography has an important advantage in that its security is often based on the laws of physics. Applications of quantum cryptography include QKD, quantum bit commitment and quantum coin tossing. These applications have varying degrees of success. Thus problems of access control, authentication and privacy protection should be solved simultaneously. Hence, this research proposes Quantum cryptographic approach for Decentralized Access control and Privacy Preserving in cloud (Q-DAPP) system over the classical Decentralized Access control and Privacy Preserving in cloud (DAPP) system.

Literature review: Cloud computing is gaining more attention these days. Buyya (2013) lists the opportunities, challenges in cloud computing along with the variety of services provided. Since, the data stored in the cloud can be tampered by illegal users, access control is a major issue in cloud. The users of the cloud do not wish their identity to be revealed. Hence privacy preservation is turning out to be an issue as stated in (Shaikh and Haider, 2011). Only authorised users must be given access to valid service. Richardson *et al*. (2010) discussed the three access control techniques. In user-based access control, the permission is granted based on the users. Since, the cloud is dynamic the access control list cannot be established. In role-based access control the roles of the users who can access the files needs to be specified while in attribute-based access control, the attributes that defines the users of the system needs to be specified as a necessary criteria. This ensures that individual identity of the user is not revealed and hence privacy is preserved. The files stored in the cloud need to be secured using appropriate encryption techniques. The keys used for encryption are to be given to the users using a central trusted authority (Wang *et al*., 2012). But, this approach has the disadvantage of single point failure, multiple KDCs are used (Ruj *et al*., 2014). Sushmitha (2014) suggested using multiple keys corresponding to the attributes that identify the user instead of a single key while encrypting using Attribute Based Encryption (Qiao *et al*., 2014). Each KDC has a set of disjoint attributes. When a user requests a KDC for the secret keys of its attributes for decryption, the KDC provides

with the secret keys corresponding to the attributes of that particular user (Kuhn *et al*., 2010). A user will be able to decrypt the cipher text only if he has attributes satisfying the access policy which is attached to the file by the creator of the file (Ruj *et al*., 2014). Also, the user must have attributes matching the claim policy attached to the file in order to write into it.

User revocation attack is where the users who previously possessed attribute keys are no longer valid authenticated users, yet trying to access the resources of the cloud. The files that were previously accessible by a user must be blocked from them once their attributes are revoked (Chen and Ma, 2014). The man-in-the-middle attack is where a third party user tries to intercept the keys being transmitted from the KDC to another user. It is efficiently handled using the quantum key distribution protocol BB84. In the proposed research, an encrypted file is converted to a quantum file before storing in the cloud. This conversion is done using a quantum circuit called Q-BOX which is a reversible circuit designed using the quantum gates such as Feymen and Toffoli gate due to its low cost of computation (Hasan, 2008). The input data is converted to quantum data and stored in the cloud and without the Q-BOX the cloud will not be able to read the file which makes it more secure (Arun and Saravanan, 2013; Maslov *et al*., 2005). All the basic quantum gates used in designing the circuit for the quantum box are discussed by Arun *et al*. (2013). The work discussed by Ruj *et al*. (2014) fails to tackle a few attacks such as the man-in-the-middle attack and user revocation attack. Since, security of the file stored in cloud is of prime importance all possible attacks of the classical cryptographic system should be efficiently handled. But with the advancement in technology, there are many possible attacks to break the security provided. In order to enhance the security provided previously and to tackle attacks like man-in-the-middle, server-colluding-attack, user-revocation attack the following system is proposed. The proposed system uses quantum cryptographic techniques to provide improved security. The use of quantum techniques increases the time and space complexity only by a small margin but provides more security than the classical system. Performance does not deteriorate when quantum cryptography is introduced as the time difference taken between the system with quantum and without quantum seems to be negligible and also the difference in heap space occupied by the system with and without quantum is very minimal. While, performance does not deteriorate, security of the system becomes highly enhanced and system becomes secure against most of the attacks. The objectives of the proposed systems are to authenticate users who store and modify their data in the cloud, to solve man in the middle attacks, to improve security of the data by converting to Quantum data and to provide security to keys during distribution of keys by KDC to Users.

MATERIALS AND METHODS

System architecture

Q-DAPP system architecture: The Q-DAPP system shown in Fig. 1 consists of four stakeholders: trustee, cloud, KDC and user. The cloud in the proposed system is used for storing and retrieving the files. Cloud helps in sharing of data to authorised valid users of the system. It performs the claim policy verification, to ensure that only valid authenticated users of the system with appropriate access rights receives the file from the cloud. The trustee is the only reliable source in the system, generating unique User Identity (UID) for the users of the system. The users are then identified only using the and even secret keys are generated based on the of the user. The KDC performs the job of generating the keys for each user. Each KDC has its own public and secret keys for every attribute that it possesses. It is using these keys

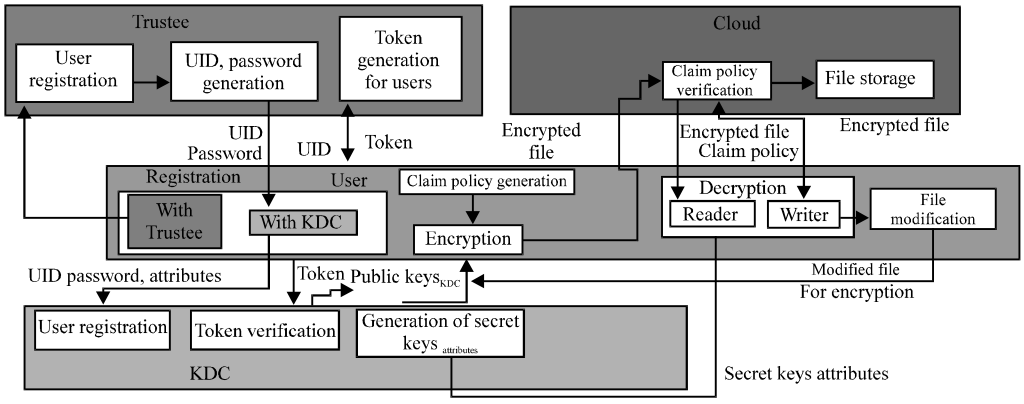

Fig. 1: Architecture of the Q-DAPP system

encryption is performed. The users of the system are allowed to utilize the resources of the cloud based on the attributes they possess. Authenticated users who do not satisfy the constraints of the creator of the file are given a warning message by the cloud.

Trustee: The trustee generates user identity, password and token generation for the users. It issues and password to the users when the user initially enters the system. These user identities are generated for all the users who enter the system. Each user has to register itself with the trustee. Only, when the user submits this token, it can seek keys from the KDC for encryption and decryption.

Cloud: The cloud verifies claim policy and maintains file storage. Whenever, a user wants to modify the data on the cloud it must satisfy the claim policy. The cloud verifies whether that particular user has rights to modify the file using the claim policy. The claim policy verification is done at the cloud server. The quantum file is stored in the cloud so that the cloud cannot tamper or view the contents of the file.

User: There are 3 types of user in the system. Creator is the user who creates and stores the file in the cloud along with the access policy and claim policy. Reader is the user requesting access to the file only for reading purpose and can decrypt the file only when it satisfies the access policy. Writer is the user who wants to modify the file and can only receive the file from the cloud only when it satisfies the claim policy. Each user registers itself with the trustee and receives and password after its registration. The user also encrypts the data, access policy and mapping of attributes using the public key of all the KDCs in the system. The creator generates the claim policy which determines the attributes that the other users must possess to modify the file. The claim policy is sent to the cloud along with the encrypted version of the file. The decryption is done using the secret keys of the attributes of the user. The bi-linear mapping technique is used to map the encryption and decryption keys.

KDC: Since, the system is decentralised, there can be multiple KDCs. Each KDC is responsible for one or more attributes. A user may receive zero or multiple keys for decryption from a particular KDC. The KDC performs user registration and token verification. The user initially submits the and password received from the trustee to the KDC and the KDC stores this information for future verification. While, generating the signing keys, the KDC verifies the token using the public key of the trustee. KDC

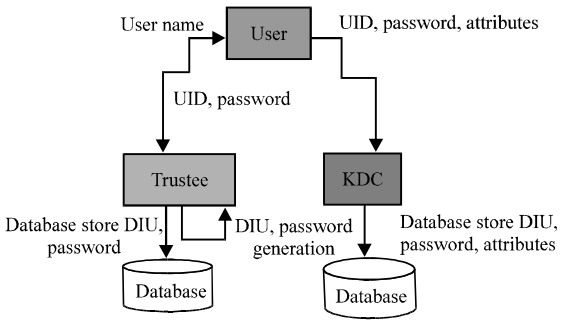

Fig. 2: Initial Registration of the user

is responsible for transmitting the keys to the user for encryption and decryption. The decryption keys are the secret key of the attributes which are transmitted to the users only when the user has previously registered with the KDC and that the KDC is responsible for only that particular attributes that the user possess.

Data flow in the Q-DAPP system: Whenever, a user enters the system, the user must register itself with both the trustee and the KDC as shown in Fig. 2. To register with the trustee, the user initially sends its name to the trustee. The trustee generates both and password for the user. The and password are both unique in the system. Similarly to register with the KDC, the user sends its and password to the KDC. Since, the KDC trusts the trustee, it assumes that the user is valid and on reception of the, password and the attributes from the user, the KDC stores the password and only those attributes that the particular KDC is responsible for. Figure 3 shows the detailed data flow in the DAPP system. Here, the user's role is split into reader, writer and creator. The data flow is explained in a sequential manner:

- Step 1: The user who wishes to create a file, sends its to the trustee requesting for the token
- Step 2: The trustee checks whether the user has previously registered itself and then generates the token which includes the sign of the trustee. The trustee signs the UID using its private key
- Step 3: The creator then forwards this token to all the KDCs in the system
- Step 4: The KDC verifies the sign on the token using the public key of the trustee
- Step 5: All the KDCs send their public key for encryption for the attributes that they are responsible for
- Step 6: The creator encrypts both the message and the access tree using ABE technique

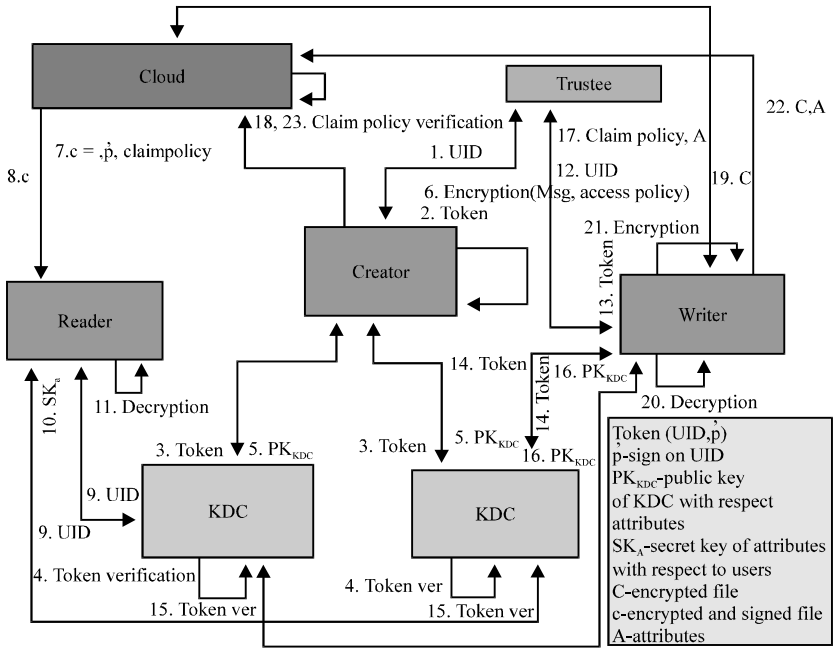

Fig. 3: Data flow of the Q-DAPP system

- Step 7: Now, the creator sends the claim policy along with the encrypted message to be stored in the cloud
- Step 8: When another user wants to read the file, it requests the cloud for the cipher text
- Step 9: The user then forwards the to the KDC, requesting for the secret keys of attributes
- Step 10: If the KDCs are responsible for that particular attribute, it sends the secret keys to the users that enable them to decrypt the file that they received from the cloud. If the KDCs are not responsible for the attributes, then they don't send any response back to the user
- Step 11: The decryption is done at the user end using the ABE
- Step 12: If a user wants to modify the content of the file, the user sends its to the trustee requesting for the token
- Step 13: The trustee generates the token for the writer similar to that of the creator. The token is signed using the trustee's private key
- Step 14: The writer forwards the token to all the KDCs
- Step 15: The KDC verifies the token using the trustee's public key
- Step 16: The KDC sends its own public key for encryption and secret key of attributes for decryption
- Step 17: The writer sends its own attribute to the cloud for the claim policy verification to take place
- Step 18: The cloud verifies the claim policy that was sent by the creator previously
- Step 19: If the claim policy is satisfied by the writer then it is assumed to be an authenticated user who can modify the file and hence the cloud sends the cipher text to the user
- Step 20: The user now decrypts the file using ABE
- Step 21: After modification of the existing file, it performs all the tasks that was performed initially by the creator. The writer encrypts the file using ABE
- Step 22: The encrypted file is then sent back to the cloud
- Step 23: Claim policy is verified again and stored in the cloud

Figure 3 explains the interaction between various components of the system in sequential steps. The user who wants to store a file in the cloud, authenticates itself using the token to all the KDCs. The KDC then sends the keys required for encryption. The creator formulates the access and claim policy for the file and encrypts the file and the corresponding access policy using the keys received from the KDC. The encrypted version of the file and the corresponding claim policy are sent to the cloud for storage. Whenever, a user makes a request for reading the file, the file is sent to the user by the cloud. But, only if the user has the corresponding attributes, the user can decrypt the file. If the user do not have the required attributes, then user cannot decrypt the file. The user has the original encrypted version of the file from the cloud, but cannot decrypt it. The user who is willing to modify

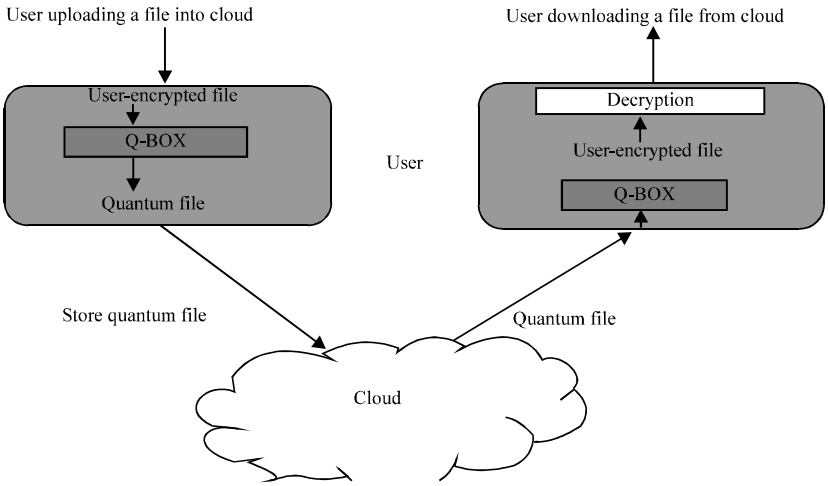

Fig. 4: The Q-BOX integration in QDAPP system

the file, sends a token to all the KDCs to authenticate itself and receives both the encryption and decryption keys. The user then sends its attributes to the cloud, so that the cloud can perform the claim policy verification. On successful verification, the cloud sends the file to the user and the user decrypts the file, modifies it and then performs the required encryption before placing them on the cloud. The cloud performs claim policy verification once again before updating the original version of file.

Q-BOX integration in Q-DAPP System: Figure 4 shows the integration of the Q-BOX to the classical system which provides an additional layer of security to the keys and the file. All the users in the system are assumed to have the same quantum circuit. The creator after encrypting and signing the file passes it through the quantum circuit to generate a meaningless quantum data so that the cloud cannot read the contents of the file and then stores it in the cloud. All the users in the system, receives this quantum file from the cloud and then converts it back to the encrypted file by using the reversible quantum circuit which then decrypts the encrypted file using appropriate keys. The quantum keys is distributed using BB84 protocol in this research. The KDC transmits the keys for encryption and decryption to the users. During the transmission of keys, the keys can be intercepted by a third party user. To prevent the invalid users from sniffing the keys, BB84 protocol is used during the transmission of keys. If the key to be transmitted uses BB84 protocol, then the interceptor cannot retrieve the keys and hence, man-in-the-middle attack can be tackled easily. The TSG and HNG gates are used which are proved to be reversible. The representation of TSG and HNG gates are shown in Fig. 5a, b.

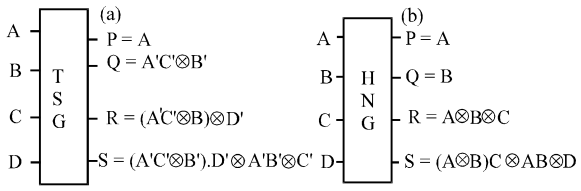

Fig. 5: a) The TSG gate and b) HNG gate

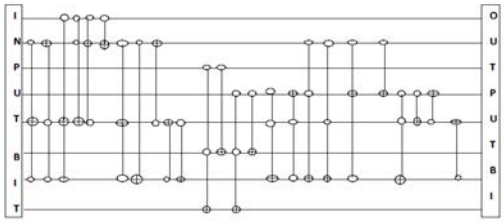

Fig. 6: Quantum Circuit-QBOX

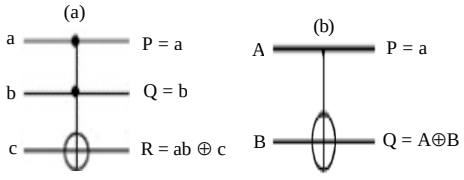

Fig. 7: a) Toffoli gate and b) Feyman gate

Q-OX quantum circuit using gates: Figure 6 shows the design of the quantum circuit called Q-Box. The Q-Box is a reversible circuit constructed using the Toffoli gate and Feyman gate. The input is split into blocks of 8 bits and fed into the Q-BOX whose output is 8 qubits. The representation of Toffoli and Feyman gates are shown in Fig. 7a, b. The Toffoli gate is a reversible quantum gate which takes three bits input and gives three qubits

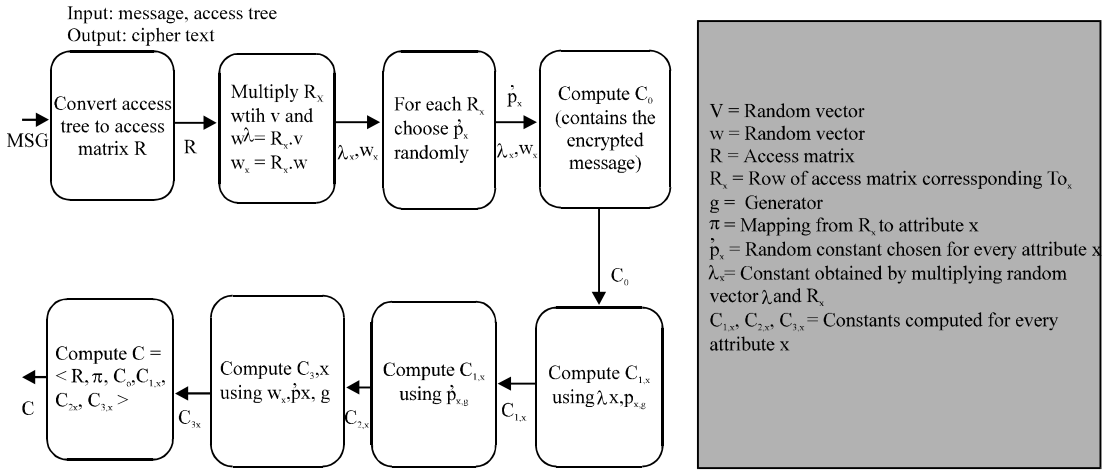

Fig. 8: Attribute Based Encryption (ABE)

output. It toggles the third bit based on the truth value of first two bits. The Feyman gate is a reversible quantum gate which takes two bits input and gives two qubits output. It toggles the second bit based on the truth value of first bit.

Algorithms: In order to ensure access control and privacy preservation in cloud, encryption schemes are based on the attributes that the user possess. The algorithm discussed in 4.1.1 is applicable for decentralized cryptographic approach.

Attribute based encryption: In ABE, user has a set of attributes in addition to its unique identity. Multi-authority ABE protocol is used here. Figure 8 depicts the algorithm for encryption of the file by the user. The steps of the algorithm are:

- Step 1: The access tree is a tree in which attributes form the leaves of the tree and the intermediate nodes and the root are made of AND and OR gates. If the parent node consists of OR gate then the child uses an unchanged vector. If the parent node is AND gate then the left child uses the same vector as the parent with 1 appended to the right and the right child becomes (0, -1)
- Step 2: Multiply each row of access matrix corresponding to attribute x with two random vectors v and w. Pick a random seeds. The first value of vector v is s. The first value of the vector w is 0
- Step 3: Another random value TMx is chosen for each attribute
- Step 4: Compute $C_0 = MSG\ e(g, g)^s$
- Step 5: Compute $C_{1,X} = e(g, g)^{\lambda(x)}\ e(g, g)^{\alpha_{\pi(x)}\lambda(x)}$ for all the attributes
- Step 6: Compute $C_{2,X} = g^{\lambda(x)}$ for all the attributes
- Step 7: Compute $C_{3,X} = g^{y(\pi(x))\lambda(x)}\ g^{w(x)}$ for all the attributes
- Step 8: Compute C = <R, π, C_0, $C_{1,x}$, $C_{2,x}$, $C_{3,x}$> where R is the access matrix, π is the mapping from R_x to attributes, now C is sent to the cloud for it to be stored

The input to the encryption function is the access tree and the file to be encrypted. The access tree is a binary tree comprising of attributes and threshold gates. The attributes form the leaves of the binary tree and threshold gates are present as intermediate nodes. The tree is converted to access matrix based on the rules specified in step 1. Then, two random vectors are formed. The seed chosen s forms the first element of vector v and 0 is the first element of vector w. The rest of the elements of the vector are randomly chosen from the elements that form the cyclic group; the g is the generator of the cyclic group; e(g, g) is the bilinear map. The πx refers to the attribute position in the access tree; $\alpha_{\pi(x)}$ and $y_{\pi(x)}$ are the secret keys corresponding to attribute x. There exists constants λ_x and w_x for every attribute x. A constant TMx is chosen for each attribute x and C_0, $C_{1,X}$, $C_{2,X}$ and $C_{3,X}$ is calculated for each attribute. C_0 comprises of the message in encrypted form. The other three constants are computed for every attribute.

Decryption (ABE) algorithm: The receiver receives attributes and secret keys from the attribute authority and it is able to decrypt the information if it has matching attributes. Figure 9 depicts the steps that are to be followed during decryption of the file by the user. The Steps of the algorithm are:

- Step 1: Both the access matrix R and the mapping π is obtained from the cipher text

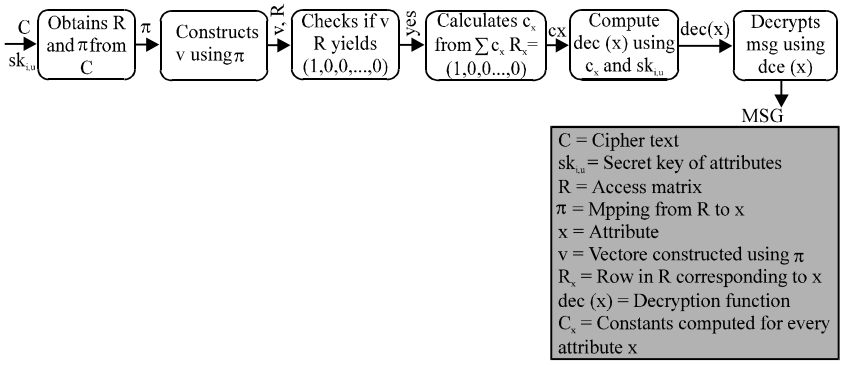

Fig. 9: Decryption (ABE): input: cipher text, secret key of attributes; output: message

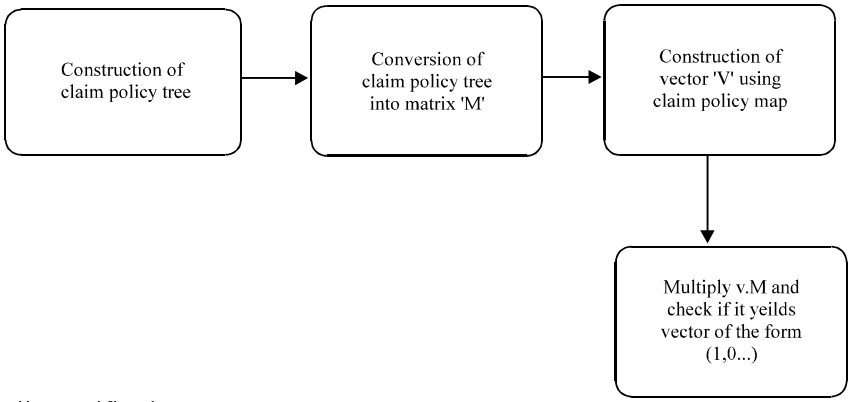

Fig. 10: Claim policy verification

- Step 2: The user constructs vector v based on the mapping of attributes (π)
- Step 3: If the multiplication of the vector v and access matrix R results in (1,0,..,0) decryption proceeds, else decryption is impossible
- Step 4: $\Sigma c_x R_x = (1,0,...,0)$ since R_x is also known, the only unknown in the above equation is c_x which can be computed easily
- Step 5: Calculate dec (x) for each attribute x in the subset using the formula $dec(x) = C_{1,x} e(H(u),C_{3,x}) / e(sk_{\pi(x),u}, C_{2,x})$
- Step 6: MSG is obtained from C_0 as follows MSG = C_0/π dec(x)

The input to the decryption function is nothing but the cipher text. The access matrix R and the mapping π is derived from the cipher text. Once it is done, the user constructs a vector v using the map π. Now on multiplication of R and v if a vector of the form (1,0,0,...) is obtained then decryption is possible, else a warning message is displayed to the user stating that he is not eligible to decrypt the file. The decryption constant, dec(x) is computed for each attribute x using the constants $C_{1,X}$, $C_{2,X}$ and $C_{3,X}$ and the corresponding secret key of the attribute pertaining to the user $sk_{\pi(x),u}$ where π_x gives the position of the attribute x in map and u is the of the user wishing to decrypt the file.

Claim policy verification algorithm: Figure 10 depicts the steps performed by the cloud when a write request is received. Cloud performs the claim policy verification using the policy specified by the creator of the file. On successful verification, the file is transferred to the user who had previously made the write request.

Step 1: The creator of the file specifies the claim policy determining the writers of the file. Based on the claim policy a binary tree is constructed.

Step 2: The tree is then converted to a matrix. The claim policy tree is a tree in which attributes form the leaves of the tree and the intermediate nodes and the root are made of AND and OR gates. If the parent node consists of OR gate then the child uses an unchanged vector. If the parent node is AND gate, then the left child uses the same vector as the parent with 1 appended to the right and the right child becomes (0, -1).

Step 3: Using the claim policy, tree claim policy map is defined. Now, the cloud creates a vector V on receiving the attributes from the user using the claim policy map.

Step 4: The cloud performs the verification by multiplying v and M and checking whether, it yields a vector of the form (1,0,...).

The cloud performs claim policy verification on receiving the write request from the user. To perform this verification, the cloud constructs a tree based on the claim policy specified by the creator of the file. It then constructs the vector using the claim policy map when it receives the attributes from the writer. It then performs multiplication of the claim policy matrix and the vector that was constructed previously and checks if the multiplication yields a vector of the form (1,0...). If the condition is satisfied then the file is transferred to the requested user.

Implementation

Platform and scenario setup: This study discusses the various modules that are implemented and the tools that are used for implementing them. The output that is obtained with respect to the scenario is discussed. The platforms and tools used are as follows. The operating system that is being used is Ubuntu. For implementing the cloud, Amazons 3 is used. The user module comprises of four packages:

- Connectivity
- UI
- Crypto
- Q-Box

In the connectivity package, the http connection is handled and the UI package consists of the code for designing the UI screens visible at the user end. The crypto package comprises of the code for encrypting and decrypting the file and the Q-Box. After encryption the file is applied to the quantum box and it is stored in the cloud. The encryption and decryption module is present at the user end where the encryption is based on bilinear cryptography. The jpbc library that is present is used for implementing bilinear pairing in java. This library contains support for cyclic groups and supersonic curves. All the users of the proposed system are assumed to have this encryption-decryption algorithms along with the Q-Box.

The creator of the file is supposed to enter the access policy and claim policy corresponding to the file. The access and claim policy of the file is sent to the cloud for verification at the later stage. KDC performs the job of key generation. Each KDC is responsible for disjoint set of attributes. KDC then generates the secret key for each user and stores it in a file. There are two files kdcp and kdcs for each KDC which contains the public and secret keys of the attributes respectively, for the attributes that each KDC is responsible for. A file is generated for each user on its registration with the KDC. The KDCs of the proposed system resides in different machines and the users are assumed to know the location of all the KDCs in the system. The KDC applies the BB84 protocol to the keys before transmitting them to the user. The cloud module is responsible for claim policy verification. The file stored in the cloud is quantum encrypted and hence the cloud cannot tamper the file. Even, if by any chance the cloud gets to know the keys, retrieval of file is impossible because the cloud does not have the Q-Box. The time at which each file was uploaded or modified is also stored along with the file name.

Trustee contains a user database designed using MySql. The database table consists of three fields User Id (UID), user name (uname) and password for each user of the system. The trustee generates password for each user randomly during the initial registration of the user. The trustee signs the and sends it as a token to the user. The user with this token is considered to be a valid user as the trustee is assumed to be a trusted entity in the system. The database is stored at the trustee's site. The user registers itself with the trustee when it enters the system. The user id is generated by the trustee and must be unique within the system. After the initial registration of the user, whenever a user logs in the password entered by the user is checked against the password field of the appropriate user in the database table.

The system was developed for an organisation scenario where one of the two KDCs are responsible for the roles (CEO, dept manager, supervisor, labour), the other KDC holds the secret keys of the departments (production, sales and marketing, personnel, finance and accounting). There are six users in the system and it is assumed that there are six files uploaded in the cloud. The creator of the file defines the access policy and claim policy. The users satisfying the access policy can read the file and those satisfying the claim policy can modify the files. Based on the policy, binary tree is generated and access policy and claim policy maps are obtained from the tree. The cloud uses the claim policy map when it receives the attributes from the writers and then it forms a vector and performs claim policy verification. On the other hand, the access policy map is received by the reader along with the cipher text. The reader uses this map to form the corresponding vector and perform decryption. The attributes chosen for KDC1 are: CEO, department

Table 1: The access permission of each user of file

File name	Readers (users)	Writers (users)
1	1, 2, 6	1
2	2, 4, 5, 6	2, 6
3	1, 5, 6	5
4	2, 5, 6	5
5	3, 5, 6	5
6	4, 5, 6	5

manager, supervisor and labours. The attributes chosen for KDC2 are: production, sales and marketing, personnel finance and accounting. The scenario setup in the Q-DAPP system is follows; the users of the proposed system are assumed as:

- User1: Production department (e), department manager (b)
- User2: Sales and marketing department (f), department manager (b)
- User3: Personnel department (g), department manager (b)
- User4: Finance department (h), department manager (b)
- User5: Finance department (h), CEO (a)
- User6: Sales and marketing department (f), CEO (a)

The creators of file are file1: user1, file2: user2, file3: user1, file4: user2, file5: user3, file6: user4. The access permission for each user of the file is assumed as follows (Table 1). The access policies for the files are:

- File1: (Production department and (ceo or department manager) or supervisor)) or sales and marketing department
- File2: (Sales and marketing department) and (ceo or department manager) or supervisor) or finance department
- File3: (Production department and (department manager or supervisor)) or CEO
- File4: (Sales and marketing department and (department manager or supervisor)) or CEO
- File5: (Personnel department and (department manager or supervisor)) or CEO
- File6: (Finance and accounting department and (department manager or supervisor)) or CEO

The access policy maps are:

- File1: (Production department, ceo, department manager, supervisor, sales and marketing department)
- File2: (Sales and marketing department, ceo, department manager, supervisor, finance department)
- File3: (Department manager, supervisor, production department, CEO)
- File4: (Department manager, supervisor, sales and marketing department, CEO)
- File5: (Department manager, supervisor, personnel department, CEO)
- File6: (Department manager, supervisor, finance and accounting department, CEO)

The claim policies for the files are :

- File1: ((CEO or department manager) and production department)
- File2: ((CEO or department manager) and sales and marketing department)
- File3: ((Personnel department or finance and accounting department) and CEO)
- File4: ((Personnel department or finance and accounting department) and CEO)
- File5: ((Personnel department or finance and accounting department) and CEO)
- File6: ((Personnel department or finance and accounting department) and CEO)

The claim policy maps are:

- File1: (CEO, department manager, production department)
- File2: (CEO, department manager, sales and marketing department)
- File3: (Personnel department, finance and accounting department, CEO)
- File4: (Personnel department, finance and accounting department, CEO)
- File5: (Personnel department, finance and accounting department, CEO)
- File6: (Personnel department, finance and accounting department, CEO)

Handling the attacks

Man-in-the-middle attack: Man-in-the-middle attack is the case where a user of the system sniffs the keys of another user in transit. When, the keys are transferred from the KDC to the appropriate user another user sniffs the keys and uses these keys to gain access to the files. The BB84 protocol is then implemented at the KDC and the user end. This creates a quantum secure channel to protect the keys in transmission. If there is an interceptor then both the parties in communication will know about the eavesdropper due to the presence of quantum secure channel.

User revocation attack: User revocation is where the user of the system are revoked of all the attributes he possess. Once the attributes are revoked the user must be denied access to the files even if he possesses the appropriate attribute keys. This issue is handled by using the token signed by the trustee. The database table at the trustee is updated every time a user of the system is revoked of the attributes. Whenever a user performs any file actions, the trustee checks whether the user is still a valid user and generates a token by signing the user id with its private key. The cloud sends the latest version of the file only on receiving the token form the trustee. Therefore, if the user's attributes are revoked they cannot procure the file from the cloud.

Server colluding attack: Server colluding attack is the case where individual users who are not allowed to access the file combine their attributes and tries to gain access to the file. The KDC generates secret keys for every attribute pertaining to the user, i.e.) if a user possess two attributes then he/she has two secret keys, one for each attribute from the respective KDCs. The generation of the secret key uses user id as one of its parameter. Now, if two users collude, the secret keys corresponding to the attributes will have different user ids. Thus, two user ids is an indication that the users have colluded and hence there is a warning message.

RESULTS AND DISCUSSION

Performance evaluation: The performance of the system is measured in terms of time and space complexity. Figure 11-16 compares the time taken by the modules of the system with and without Q-Box. Figure 17-21 shows the heap space occupied by different objects at the client end with and without Q-Box.

Encryption time comparison with and without Q-Box: Figure 11 shows the comparison of the encryption time taken for a file with and without the presence of the quantum box. The difference in time is negligible since the variation is only a few milli-sec. This will help in providing a justification for the use of quantum cryptography in future. The difference in time for various file sizes ranges from 0.312-3.288 sec. The maximum difference occurs for the file with the large size. The average difference in time is 0.3273 sec which is >1 sec. The graph also suggests that the difference proportionately increases as the file size increases. Since, the quantum technique provides improved security and average deviation is only 1 sec, it is efficient to use quantum approach in cryptography. Figure 12 shows the percentage increase in the time taken

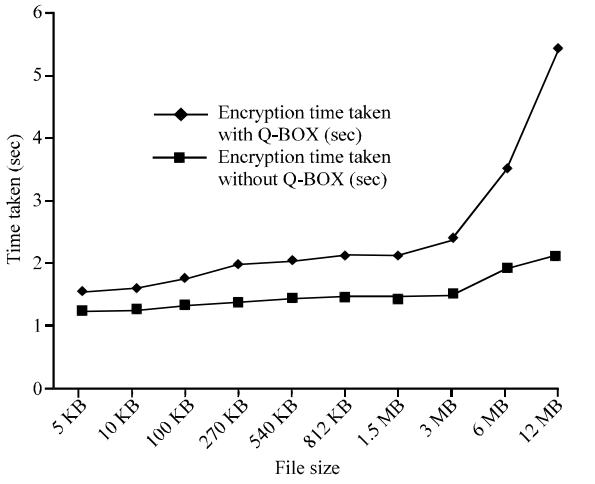

Fig. 11: Encryption time comparison with and without Q-Box

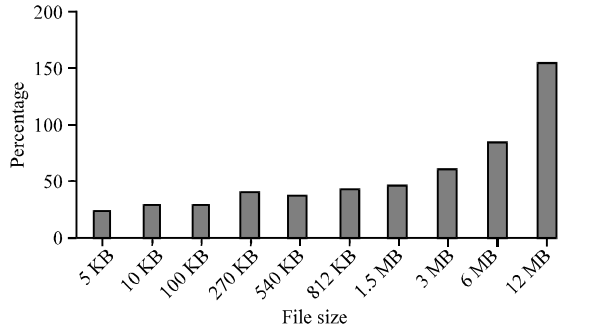

Fig. 12: Percentage increase in time taken during encryption

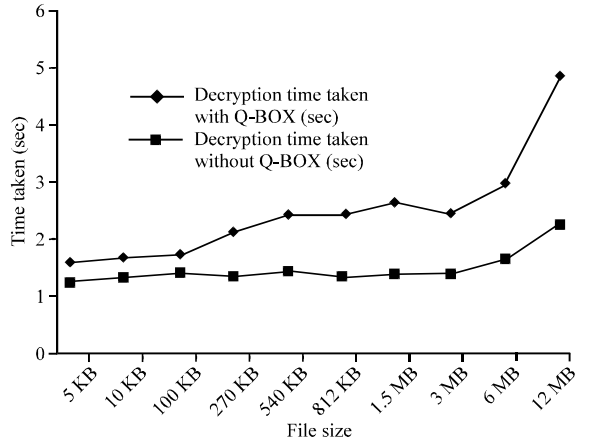

Fig. 13: Decryption time comparison with and without Q-Box

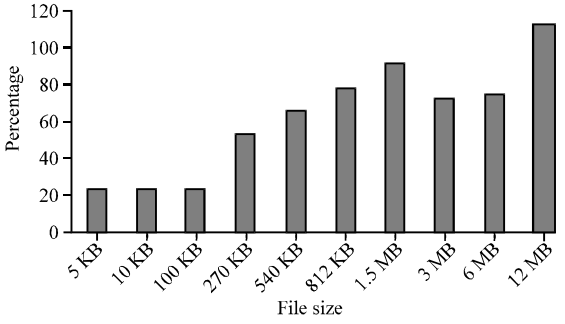

Fig. 14: Percentage increase in the time taken during decryption

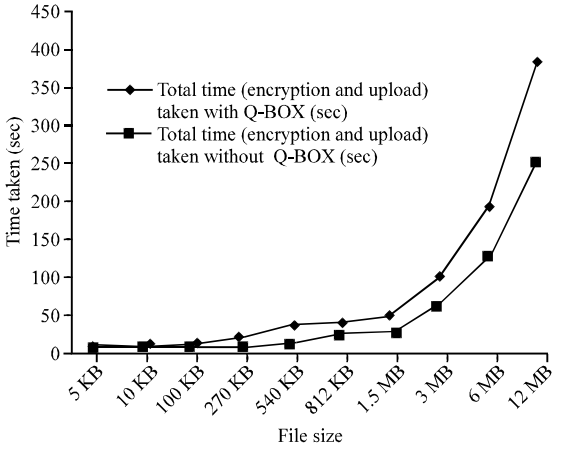

Fig. 15: Total time (encryption and upload time) with and without Q-Box

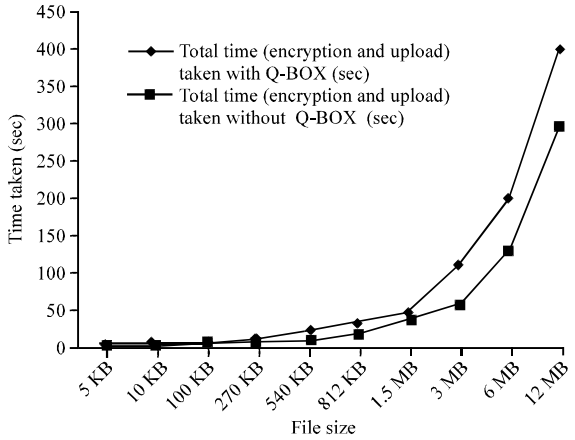

Fig. 16: Total time (decryption and download time) with and without Q-Box

with and without quantum box during the encryption for different file sizes. It suggests that there is not much variation in the time taken and the impact is quite

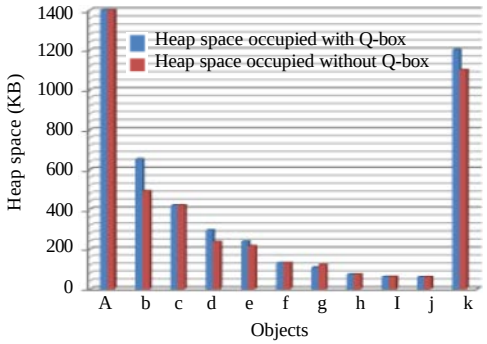

Fig. 17: Heap space comparison with and without Q-Box

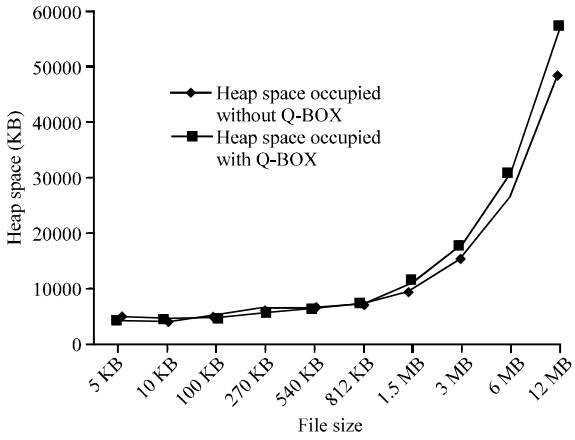

Fig. 18: Heap space occupied during encryption with and without Q-Box

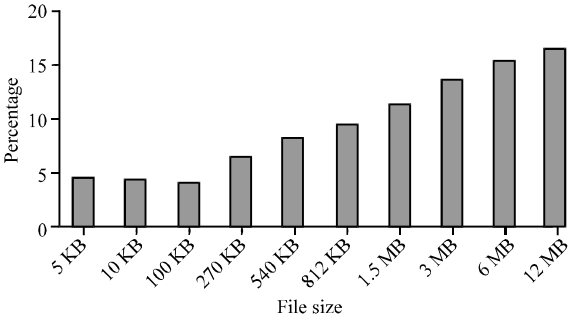

Fig. 19: Percentage increase in heap space during encryption

prevalent only when quantum box is applied to a larger file. The average percentage increase during encryption is 55.36% while the maximum being 154.46%.

Decryption time comparison with and without Q-Box: Figure 13 shows the insignificant difference in time with and without the quantum box during decryption. There is only a minor difference in time, i.e.) a few milli-sec. The

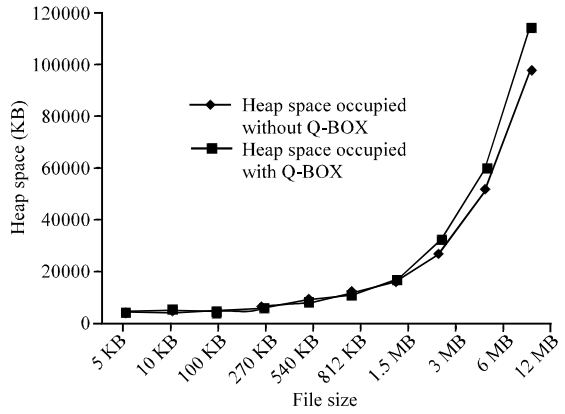

Fig. 20: Heap space occupied during decryption with and without Q-Box

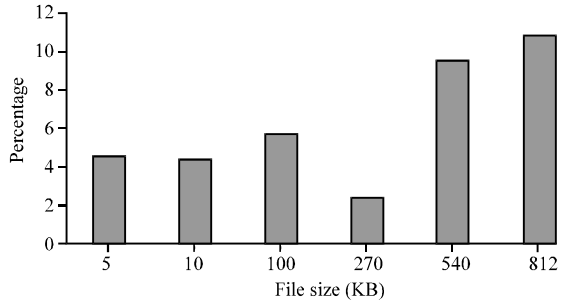

Fig. 21: Percentage increase in heap space during decryption

maximum difference between the two curves is 2.58 sec and it occurs for a file size of 12 MB. The deviation between the two curves ranges from 0.3-2.58 sec. The average variation time during decryption is 0.993 sec.

Figure 14 shows the percentage increase in the time taken with and without quantum box during the decryption for different file sizes. It suggests that there is not much variation in the time taken and the impact is quite prevalent only when quantum box is applied to a larger file. The average percentage increase during decryption is 61.44% while the maximum being 111.68%.

Total time (encryption and upload time) with and without Q-Box: Figure 15 shows that the time taken for encrypting and uploading file to the cloud is proportionate to the file size and there is only a minor difference in time taken whether the file is quantum encrypted or not.

Total time (decryption and download time) with and without Q-Box: Figure 16 shows the time taken for decryption and downloading a file from the cloud by the user. The file may be quantum or non quantum and the time difference between them is negligible.

Heap space comparison with and without Q-Box: Figure 17 shows the comparison of heap space occupied by different objects with and without quantum. It can be seen that there is only a minor difference in the heap space when the quantum box is used. The variation is only a few KBs. The maximum difference in heap space for individual object is only 161 KB. The overall increase in heap space due to the use of quantum box for a file is 333.6 KB. With the rapid growth in technology, variations in KB is negligible and insignificant.

Heap space occupied during encryption with and without Q-Box: Figure 18 shows the comparison of the heap space during encryption for files of different sizes with and without the presence of the quantum box. The difference is negligible since the variation is only a few bytes. In the current scenario where memory is easily available, memory deviation of few bytes does not matter. This will help in providing a justification for the use of quantum cryptography in future. The difference in terms of bytes for various file sizes ranges from 205-7577.8 bytes. The maximum difference occurs for the file with the large size. The average deviation being 1685.57 bytes serves a valid justification for the use of quantum techniques. Figure 19 shows the percentage increase in the heap space occupied with and without quantum box during the encryption for different file sizes. It suggests that there is not much variation in the time taken and the impact is quite prevalent only when quantum box is applied to a larger file. The average percentage increase during encryption is 8.84% while the maximum being 15.42%.

Heap space occupied during decryption with and without Q-Box: Figure 20 shows the insignificant difference in heap space with and without the quantum box during decryption. There is only a minor difference in the heap space being used. The maximum difference between the two curves is 14958 bytes and it occurs for a file size of 12MB. The deviation between the two curves ranges from 218-14958 bytes. The average variation in the heap space occupied during decryption for various file sizes is 3149.9 bytes. Figure 21 shows the percentage increase in the heap space occupied with and without quantum box during the decryption for different file sizes. The average percentage increase during decryption is 9.57% while the maximum being 16.98%.

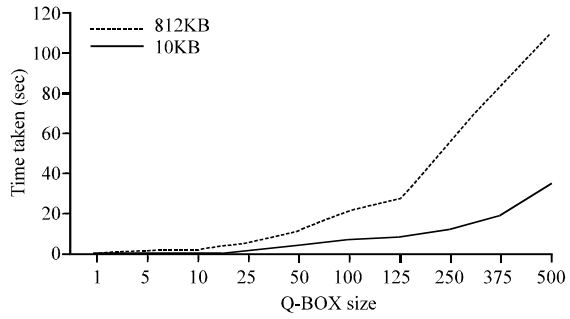

Fig. 22: Time taken for conversion of encrypted file to quantum file for different sizes of Q-Box

Time taken for the conversion of classical encrypted file to quantum encrypted file for different sizes of Q-Box: Figure 22 shows the time taken for applying the quantum box of different sizes to a previously encrypted file. The comparison of the time taken for two files of sizes 812 KB and 10 KB is shown in the graph. The x-axis shows the number of times the quantum box is called in a loop over the encrypted file. The graph indicates that the time taken exponentially increases as the quantum box size increases for both the files. There is a variation in the time taken for the same quantum box size. This variation ranges from 0.342-73.24 sec. On an average, there is a deviation of 22.404 sec. This indicates that the quantum box size must be carefully chosen as the file size increases.

CONCLUSION

In the proposed research, the privacy of the user is preserved and access permission to the file is granted using attribute based techniques. The attribute based cryptography ensures anonymity and thus helps in preserving the privacy. The attribute based encryption allows only users possessing a certain attributes to access the file thus enforcing strict access permissions. The KDCs are decentralized to avoid single point of failure and reduces the overhead incurred in case of single KDC. The BB84 protocol is used for distributing the keys which provide an additional layer of security to the keys during transmission. The encrypted file is converted into a quantum file and then stored in cloud ensuring that the cloud cannot tamper the data. The proposed system is resilient towards three major attacks; man-in-the-middle attack, user revocation attack and sever colluding attack.

Therefore, the proposed system is better in terms of security than the classical cryptographic systems without much compromise in the performance. The proposed decentralized access control technique with anonymous authentication provides user revocation and also user's privacy preservation. The proposed research also prevent server colluding attacks. The cloud does not know the identity of the user who stores information but only verifies the user's credentials. Key distribution is done in a decentralized way. The cloud learns the access structure used by the owner and the attributes of the users. Future reseaches may be done to hide the access structure from the cloud by scrambling the matrix. Another challenging problem would be to hide the user attributes from the cloud.

REFERENCES

Arun, M. and S. Saravanan, 2013. Reversible arithmetic logic gate for quantum computation. Int. J. Intell. Eng. Syst., 6: 1-9.

Buyya, R., 2013. Introduction to the IEEE transactions on cloud computing. IEEE. Trans. Cloud Comput., 1: 3-21.

Chen, J. and H. Ma, 2014. Efficient decentralized attribute-based access control for cloud storage with user revocation. Proceedings of the 2014 IEEE International Conference on Communications (ICC), June 10-14, 2014, IEEE, Sydney, NSW., pp: 3782-3787.

Hasan, M.M., 2008. Low-cost realization of toffoli gate for the low-cost synthesis of quantum ternary logic functions. Proceedings of the 11th International Conference on Computer and Information Technology ICCIT 2008, December 24-27, 2008, IEEE, Khulna, Bangladesh, pp: 259-263.

Kuhn, D.R., E.J. Coyne and T.R. Weil, 2010. Adding attributes to role-based access control. IEEE. Comput., 43: 79-81.

Maslov, D., C.Young, D.M. Miller and G.W. Dueck, 2005. Quantum circuit simplification using templates. Proceedings of the Conferences Design, Automation and Test in Europe 2005, March 7-11, 2005, IEEE, USA., ISBN: 0-7695-2288-2, pp: 1208-1213.

Qiao, Z., S. Liang, S. Davis and H. Jiang, 2014. Survey of attribute based encryption. Proceedings of the 2014 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), June 30-July 2, 2014, IEEE, Las Vegas, NV., pp: 1-6.

Ruj, S., A. Nayak and I. Stojmenovic, 2011. Dacc: Distributed access control in clouds. Proceedings of the 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), November 16-18, 2011, IEEE, Changsha, China, pp: 91-98.

Ruj, S., M. Stojmenovic and A. Nayak, 2014. Decentralized access control with anonymous authentication of data stored in clouds. Parallel Distrib. Syst. IEEE. Trans., 25: 384-394.

Shaikh, F.B. and S. Haider, 2011. Security threats in cloud computing. Proceedings of the International Conference on Internet Technology and Secured Transactions, December 11-14, 2011, Abu Dhabi, pp: 214-219.

Sharbaf, M.S., 2009. Quantum cryptography: A new generation of information technology security system. Proceedings of the Sixth International Conference on Information Technology: New Generations ITNG'09, April 27-29, 2009, IEEE, Las Vegas, NV., pp: 1644-1648.

Wang, C., Q. Wang, K. Ren, N. Cao and W. Lou, 2012. Toward secure and dependable storage services in cloud computing. IEEE Trans. Serv. Comput., 5: 220-232.