Secure Collaborative Key Management System for Mobile Cloud Data Storage

Shakkeera L , Saranya A,Sharmasth Vali Y

Abstract: Mobile Cloud Computing (MCC) is the combination of mobile computing, cloud computing and wireless networks to make mobile thin client devices resource-rich in terms of storage, memory computational power and battery power by remotely executing the wide range of mobile application's data in a pay-per-use cloud computing environment. In MCC, one of the primary concern is the security and privacy of data stored in cloud. The existing techniques are not efficient to manage secret keys during key generation and key distribution processes. The objective of this project work is to develop a secure collaborative key management system (SCKMS) for mobile cloud data storage by implementing by the cryptographic techniques for file encryption and file decryption, key generation, key encryption, key distribution and key decryption processes. In our proposed methodology, DriverHQ public cloud infrastructure is used for accessing the secure file as Storage as a Service (SaaS) mechanism. For generating the secret key, the proposed work implemented with Pseudo Random Number Generator (PRNG) algorithm, it produces the sequence of random numbers for every time. The keys are distributed using general Secret key Sharing Scheme (SSS). The key pattern matching process is implemented to spilt the secret key into three partitions and sent it to client (mobile devices), cloud server and decryption server. The decryption server key and cloud sever key are mapped with client key. The key shares are grouped together using key-lock pair mechanism and it achieves key integrity during untrusted medium communication. The proposed work also eliminates key escrow and key exposure problems. The files are encrypted and decrypted using Rivest-Shamir-Adleman (RSA) algorithm. The RSA algorithm is more vulnerable against the brute force attack, because of using larger key size. Thus, the proposed SCKMS achieves data confidentiality and data integrity in mobile cloud storage data when compared to existing Key Management System (KMS). The work also reduces encryption & decryption computation and storage overhead in client mobile devices, and minimizes the energy consumption of the mobile devices efficiently.

Index Terms: Mobile Cloud Computing; Key Management; Secret Sharing Scheme; Pattern Matching.

I. INTRODUCTION

A. Mobile Cloud Computing

Mobile Cloud Computing [1] is the amalgamation of mobile computing, cloud computing, and wireless networks to carry affluent computational resources to mobile clients, cloud computing providers as well as network operators.

Manuscript published on 30 July 2019.
***** Correspondence Author (s)

Shakkeera, Assistant Professor (Senior Grade) at B. S. Abdur Rahman Crescent Institute of Science & Technology, Chennai, Tamil Nadu, India.
Saranya, Student at B.S.Abdur Rahman Crescent Institute of Science & Technology, Chennai, Tamil Nadu, India.
Sharmasth Vali, Assistant Professor at Dhanalakshmi College of Engineering, Chennai, Tamilnadu, India.

The main goal of the MCC is to provide an execution environment for accessing mobile applications like image processing, gaming, and multimedia applications from thin client devices. It provides profit for cloud service providers and mobile network operators' services. MCC will give an environment for applications, providing an easy way for smaller developers to secure their services and data. MCC provides on-demand services to a shared pool of configurable computing and storage resources. It can be quickly provisioned and unconfined with negligible business effort. Mobile cloud computing is one of future development in mobile technology. MCC services combine the merits of both mobile computing and cloud computing. Hence, provides optimum services for end users.

B. Cloud Computing

Cloud Computing [2] is a new paradigm shift in Information Technology (IT). It enables many services in the form of software, platform, infrastructure, storage, security as services. It can be accessed from anywhere, anytime and anybody with high-speed network connections. It provides dynamically scalable infrastructure for application, data and file storage. With the advent of this technology, the cost of computation, application hosting, content storage and delivery is reduced significantly. The rapidly expanding wireless LAN and cellular technology and satellite services will make the information accessible anywhere and at any time. In the near future, tens of millions of people will carry a portable palmtop or laptop computers.

C. Mobile Computing

Mobile Computing [3] is a technology that performs transmission of information, video, and voice via system or any other wireless devices. The mobile computing contains mobile hardware, mobile software and mobile communication. The mobile communication includes both ad-hoc and infrastructure networks with their communication properties, data format syntax, wireless protocols and defined technologies. The mobile hardware includes thin client mobile devices. The mobile software contains the requirements and characteristics of mobile application like video & audio application, gaming application and image processing applications. The main principles of mobile computing are portability, connectivity, social interactivity and individuality. The essence of mobile computing is to be able to work from any location. The uses of iPads, tablets, smart phones, wearable computer, computers and notebooks have pushed the demand for these devices. These devices are configured to access and store large amounts of vital data.

D. *Key Management*

The process of managing cryptographic keys for secure cryptosystem is known as Key Management System (KMS). It involves the generation, storage, distribution, and replacement. Key management system also records each key access and context.

The cryptographic keys are managed in key management. A cryptographic key is used to get the ciphertext from plaintext. It provides secret key for both encryption and decryption services to achieve data confidentiality and data integrity. To strengthen the cryptosystem, message authentication codes are generated and used in hash functions. In public key cryptography, public key is used for encryption and private key for decryption purpose. In group communication, the keys are managed by the group key management. The group key management uses multicast communication; it sends the messages to the group of recipients with the multicast group address. The secret keys are used to strengthen the security in multicast communication. The popular key management system involves Key Distribution Center (KDC), Public Key Infrastructure (PKI), Certificate Authority (CA). Similarly, key management practices the organization’s tactics to achieve those strategic policy goals. Key management procedures are the documented step-by-step tasks necessary for the secure daily cryptographic operations within an organization. Clearly, it is in the best interest of any organization to establish and promote sound key management policies, practices, and procedures.

E. *Key Distribution*

Key distribution process is an important problem in inter-network communication for accessing the data in the system. The key distribution is performed by the Key Distribution Center (KDC). KDC provides keys to clients or data owners. For each communication, both the parties' request KDC to produce unique key. The secure communication is established between two parties without intervention of third party intruders. Sometimes, the keys are referred as a master key for group communication.

F. *Key Encryption and Key Decryption*

A key is the string of characters that is used to encrypt and decrypt the messages in a symmetric or asymmetric cryptography. The key encryption achieves confidentiality and integrity on data. In the proposed work, key is encrypted and send on to the user. The secret key techniques encipher the files and permit the end user to access the encrypted files. The secret key cryptography depends on length of the key used. This achieves reliability and confidentiality of data. The key space directly depends on length of the secret key. Meanwhile, the exhaustive key search attack is possible when the key is used for longer time. Therefore, the session key is used for secure communication. The session keys are valid only for scrupulous transaction. In other side, the encrypted keys are decrypted using secret key/master key (or) private key of the individual user. The encrypted keys are used for file decryption to achieve the data integrity with the confidentiality of data. This allows confidentiality service using cryptographic techniques.

G. *File Encryption and File Decryption*

File encryption is the process of converting plaintext into the hidden format or ciphertext with the public key. The process provides the security for the application data. File decryption is the reverse process which applies the private keys in public key cryptography. Public keys are accessed by anybody, but the private keys are used only by the individual entity. The private keys are unique to the user. The proposed system is implemented with RSA encryption and decryption algorithm with public and private key pairs. In some applications, message authentication codes and hash functions are used to perform file encryption and decryption. Modern cryptographic algorithms achieve data confidentiality and data integrity without interference of others to access the data.

H. *Cloud Storage*

Cloud storage is the one of the important services in IaaS. The storage spaces are created and managed in both private and public storage structures. The public storage are accessed from anywhere at any time by anybody. Here, the users compensate only for the storage space used. Public cloud storage provides infrastructure as a service in mobile cloud computing environment. Public cloud data are stored in the global data center across multiple regions or countries and the data centers are administered by a third-party service provider. In many conditions with access control [13] policies, the sensitive data are encrypted and decrypted by the end user. This work has discussed security issues concerning to mobile cloud computing. To provide security in mobile cloud computing era is one of the main concerns for cloud service providers. Mobile cloud computing security related issues for the key management & data management.

II. CONTRIBUTIONS

The main contributions of the proposed system is to develop a secure collaborative key management system (SCKMS) for mobile cloud data storage, to achieve the both data and key security, and to solve key escrow and key exposure problems. The main objective of the proposed system is to,

- Achieve data confidently and data integrity.
- Reduces encryption and decryption computation & storage overhead in client mobile devices.
- Minimizes the energy consumption of the mobile devices.

III. LITERATURE REVIEW

Jia, and Cong Wang [4] Enabling Cloud Storage Auditing With Verifiable Outsourcing of Key Updates, the paper proposes the issue for the key updates for distributed storage inspecting with key-presentation flexibility. The work proposes the primary distributed storage reviewing convention with undeniable outsourcing of key updates. In this convention, key updates are outsourced to the Third Party Authority (TPA) and are straightforward for the customer. The TPA just observes the scrambled variant of the customer's mystery key, while the customer can additionally check the legitimacy of the encoded mystery keys while downloading them from the TPA.

The work manages key and information security with less time utilization. In the interim, the TPA is semi trusted for key refreshing.

Pratima Popat Gutai et al., [5] Efficient Hierarchical Cloud Storage Data Access Structure with KDC, the paper proposes the specialist based information get to control strategy, that is just the client with the specific expert can get to information put away at the cloud, in view of their power level.

The Hierarchical edge gets to structure (HTAS) is utilized alongside Distributed Key Generation (DKG), known as Hierarchical edge Distributed Key Generation (HTDKG). The issue in the Organization, different levels clients are there in light of their assignment and some edge is doled out for each level. At the season of information get to, just particular levels clients can get the particular measure of information. Along these lines, all information isn't uncovered to any unconcern client. The outsider Key Distribution Center (KDC), which execute the HTDKG convention and assume liability of all key age, dissemination and administration exercises. This framework diminishes the handling time and improves the memory use by using of KDC. The upsides of this paper are to decrease the handling time and all level of the client can get the information in the cloud, however, the limitation is to get the information at record-breaking.

Sikhar Patranabis and Yash Shrivastava [6] discusses on an efficiently implementable version of the basic key-aggregate cryptosystem with low overhead ciphertexts and aggregate keys, using asymmetric bilinear pairings. The construction serves as an efficient solution for several data sharing applications on the cloud, including collaborative data sharing, product license distribution, and data sharing. To decrypt the data using single key. The key size is constant, it efficiently broadcast the data to multiple parities. This key aggregate system is useful for cloud supported data sharing mechanism using elliptic curves. The merits are data confidentiality is achieved and data owner must be able to revoke at any time, but large space is required for data storage.

Yuqi Wang and Kun She [7] A Practical Quantum Public-key Encryption Model, the paper proposes a down to earth quantum open key encryption show. This proposed display makes express stipulations on the age, dissemination, confirmation, and utilization of the mystery keys, along these line frames a discovery activity. In the interim, this proposed demonstrate exemplifies the procedure of encryption and decoding for the clients, and structures a blackbox customer side. This model is for the most part made out of the trusted, secure outsiders Certification Authorities (CA) and Public Key Distribution Center (PKDC), and additionally the customer's unscrambling/encryption gadgets. In this model, the CA and PKDC, which are the skeleton of the entire model, assumes a huge part in their models. The private key age calculation, which can be given by the CA or the customer as its own, is required to rapidly create countless numbers. The fundamental points of interest are each time the new irregular keys are created. In the meantime, the quantity of duplicates of the quantum open key distributed by the PKDC is constrained.

TalariBhanu et al., [8] Encryption And Decryption – Data Security For Cloud Computing utilizing AES Algorithm, the paper proposes a system that will use Advanced Encryption Standard (AES) [12] encryption utilizing USB device. The records may get from the cloud; reports are to be mixed till the USB device is associated with the PC. The proposed structure will perceive the USB that contains the private-key used for the records to be downloaded from the cloud. The exploration work is to give totally secure the records, content documents will be transferred and downloaded whenever, record encryption and decoding are finished. The work doesn't bolster pictures, sound and video records and each time the need of mystery code to get to the document in cloud framework.

Mahesh Kumar K M and Sunitha N R [9] Hybrid Cryptographically Secure Pseudo-Random Bit Generator, the paper proposes the cryptographic key ages, computerized marks and verification conventions. The proposed work contains change to Dual Elliptic Curve Deterministic Random Bit Generator (Dual EC DRBG) and proposes a composite Cryptographically Secure Pseudo-Random Number Generator (CSPRNG) utilizing altered Dual EC DRBG. They looked at the yield against the standard Pseudo-Random Number Generators (PRNGs) like Linear Congruential Generator (LCG), Blum Shub (BBS) and altered Dual EC DRBG. They work fruitful on beating the downsides of Dual EC DRBG and accomplish an expansion in cycle length of the yield grouping created. The primary points of interest are giving greater security for the information and key age and the cycle length is high. Be that as it may, the model moderate in creating pseudo arbitrary bits consequently it isn't reasonable for application which requests more prominent speed.

Nasrollah Pakniat et al., [10] Distributed Key Generation Protocol with Hierarchical Threshold Access Structure, the paper proposes another Distributed Key Generation (DKG) convention for producing a dispersed key among a gathering of members with various levels of specialist. DKG convention with Hierarchical Threshold Access Structure (HTAS) is proposed for discrete-logarithm-based cryptosystems. The approved subsets of clients is characterized by utilizing various leveled edge get to structure The proposed convention can be utilized as a part of disseminated cryptosystems to produce progressive edge signature/encryption plans. The primary favorable circumstances are giving security to each level of client and greater security for the information. Yet, the burden is that more number of exponentiation tasks are required.

Preeti Garg and Vineet Sharma [11] An Efficient and Secure Data Storage in Mobile Cloud Computing through RSA and Hash Function, the paper proposes a component which utilizes the idea of RSA (Rivest-Shamir-Adelman) calculation, hash work alongside a few cryptography apparatuses to give better security to the information put away on the portable cloud. The RSA algorithm uses a variable length encryption contents and a variable size key. A cryptographic hash work takes a message of subjective length and makes a message process of settled length. A hash work creates a short and settled length message process. The proposed conspire utilizes RSA calculation with other encryption, unscrambling procedures to secure the information such that no spillage of information on the cloud could be performed. The encryption is utilized to give security to the information while in transit.

The fundamental benefits are to give privacy and honesty to the information put away in the portable cloud. In any case, it is more costly for the cryptographic activities and exponential tasks, blending activities are required.

Giwon Lee et al., [14] An Efficient Delta Synchronization Algorithm for Mobile Cloud Storage Applications, the paper proposes a proficient delta synchronization (EDS) calculation that totals the refreshed information to lessen the synchronization activity and synchronizes the collected information occasionally to fulfill the consistency. In distributed storage, the applications where the information is shared by different versatile clients, it is basic to give the consistency among portable clients by methods for proper synchronization calculations. Specifically, if the information is every now and again refreshed and the quantity of versatile clients sharing the information is substantial, the synchronization movement can be huge. In addition, the intemperate synchronization movement in versatile systems is more imperative as far as radio asset use and vitality utilization. The paper proposes an ideal arrangement for the total and the periodical synchronization ideas a streamlining issue is defined as a Markov Decision Process (MDP) and an esteem cycle calculation is displayed for figuring the stationary deterministic strategy. The trial comes about show that EDS can pick the ideal activity that strikes a harmony between the diminishment of the synchronization movement and the fulfillment of the consistency**.**

Guanlin Wu, and Junjie Chen [15] Collaborative Storage Algorithm Based on Alternating Direction Method of Multipliers on Mobile Edge Cloud, the proposes an exchanging heading strategy for multipliers-based collective stockpiling calculation called MECCAS (Mobile Edge Cloud Collaborative Storage). The work limits the deferral of errand execution and aggregate expenses for the general task, and in the mean time expand the use of nearby data of hubs and framework dependability, Nodes on versatile edge distributed storage are prepared to do adaptively apportioning assets for capacity to build control utilization viability and diminish the danger of hubs withdrawal. Broad analyses show the prevalence of MECCAS calculation contrasted and other three baselines, i.e., ADM, RDM, and ERASURE. MECCAS can fulfill the portable edge cloud innovation highlights, similar to variable trademark data, frail calculation and dynamic hubs, to the point of limiting deferral of assignments execution, control use viability and the danger of hubs withdrawal, and in the interim guarantee dependability and full coordination of nearby heterogeneous data.

IV. EXISTING SYSTEM

In existing key management system, key authority decrypts the ciphertext using generated secret key without knowledge of data owner. This situation is referred as key escrow problem. Now a days, smart phones are more vulnerable than personal computers, laptops and servers. The privacy [16] of the data storage is more difficult. The private keys are easily exposed to unauthorized users. This leads to key exposure problem in cryptosystem. The drawbacks of the existing system are as follows:

- Security problems and key exposure chances.
- Lack of Key protection during the communication.
- The time required for decryption takes large value.
- Be short of collaborative key management mechanism.

V. SECURE COLLABORATIVE KEY MANAGEMENT SYSTEM (SCKMS)

The proposed system focuses on Secure Collaborative Key Management System aiming to enhance security of key management in mobile cloud data sharing system. The main aim and objectives of the work as follows:

- A novel collaborative key management protocol is presented. With help of that secure key management is guaranteed which is easier to deploy and compare with previous multi-authority schemes.
- Initially, the file is encrypted using key parameter. It is combination of both cloud server and key authority's public keys. Key generation is done by Pseudo Random Number Generation (PRNG) algorithm & provides the security while transmission of both the data and key. PRNG is a mathematical formula to produce sequence of random numbers, so the secret keys are predicted by the intruder.
- Random key is allocated to each entity in group that contains client, cloud server and decryption server.
- PRNG key generation algorithm to achieve data confidentiality, backward and forward secrecy, revocation, avoid key escrow and exposure problems.
- The keys are distributed using general Secret Sharing Scheme (SSS) and key lock pair mechanism to avoid key integrity issues.
- The decryption key is generated using key pattern matching algorithm to avoid impersonation problem. Here hash codes are generated as intermediate keys.
- Key verification process is implemented to improvise the security in secret key management.
- The collaborative key management system can reduces the client decryption overhead and provide the security for the key and data.

The advantages of the proposed system as follows:

- Secure key management and random key generation are provided to secure data and key, so that security problems are not there.
- Achieves data confidentiality and data Integrity.
- Vulnerable against brute force attack and impersonation problems.
- Reduces the encryption and decryption computation and storage overhead in client mobile devices.

A. System Setup

The proposed key management system is developed using software front tools like HTML and JSP in NetBeans IDE 8.1. For backend design, JDK 1.8.0 and database connectivity with MySQL 5.0.22 are used. The connectivity between application server, database server and user interface is completed with Apache Tomcat 5.0/6.X. DriveHQ storage tool is used for mobile data storage in public cloud infrastructure in secure manner. The complete model is implemented in Windows 8, 64 bit operating system with high-end hardware configurations.

B. Architecture Diagram

The proposed work focuses on Secure Collaborative Key Management System (SCKMS) for mobile cloud data storage. The Fig. 1 shows the proposed architecture diagram between mobile client (data user) and cloud server along with key authority.

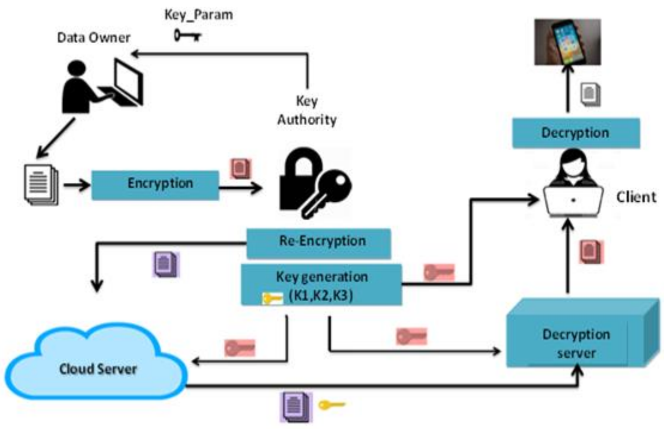


Fig.1. Architecture Diagram of SCKMS

The data owner outsources the data files to cloud server via key authority. Initially the files are encrypted using Key_Param . The owner re-encrypts the already encrypted files again using Key_Param. Encryption and re-encryption achieves data integrity in proposed system. The key authority generates the secret keys and distributes the keys using general secret sharing scheme. Finally the cloud server sends the secret key to client (mobile devices) for key verification process. Finally, secret key is used to decrypt the uploaded file in cloud storage. By using the secret key, authenticate the valid users in public cloud environment.

C. Block Diagram

The Fig. 2 depicts the block diagram of proposed system.

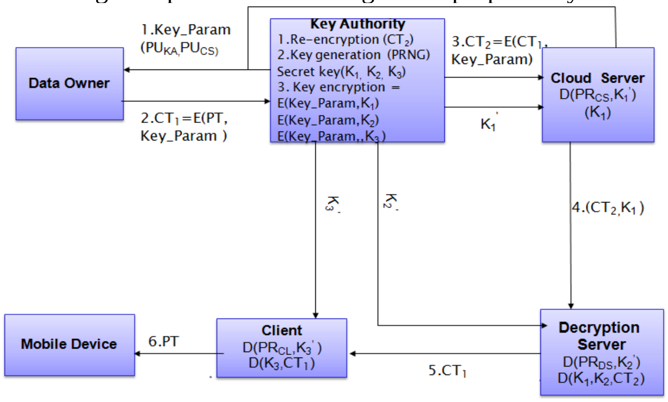


Fig. 2. Block Diagram of SCKMS

D. Operational Workflow

The Fig. 3 depicts the operational workflow between data owner, key authority, cloud server, decryption server and client in proposed key management system.

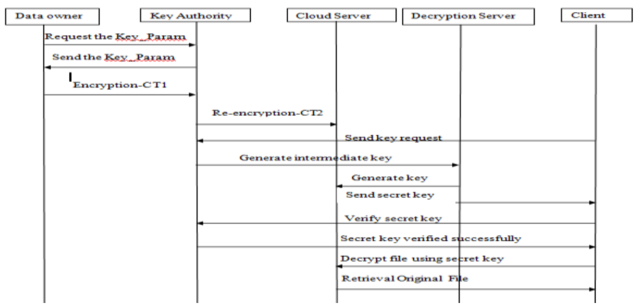


Fig. 3. Operational Workflow

E. Module Design

i) Cloud Setup using DriverHQ

The DriveHQ is offering file storage in public cloud. The files are uploaded files from personal computer to a cloud folder. It can be easily accessed, managed, share or publish the files from anywhere at any time. The DriveHQ is having following services like integrated group file sharing, cloud file backup, accessibility of DriveHQ file manager, it automatically back up files on client devices. The mobile client can access the cloud via Mobile App from anywhere and anytime. The DriveHQ cloud is the low-cost enterprise cloud storage and they offering free trial storage space upto 2 GB.

ii) File Encryption in Key Authority

Input : Key_Param (PU_{KA},PU_{CS}),Plaintext (PT)
Output : Ciphertext (CT_1,CT_2)
Key_Param : (PU_{KA},PU_{CS})

For giving security for the data in cloud we need to store the data in encrypted format. Encryption and Re-encryption process are done. Encryption is the process of converting information or data into a code/ciphertext by key authority. Re-encryption is the process of encrypting a already encrypted message or information in such a way that only authorized parties can access the messages in the input files. Encryption is done by using Key_Param. Key_Param is the combination of Key authority public key and Cloud server public key. It is the most effective way to achieve data security. The Fig. 4 depicts file encryption in key authority.

PT,CT_1
Key Authority
CT_1,CT_2

Fig. 4. File Encryption in Key Authority

iii) Key Generation using Pseudo Random Number Generator (PRNG) Algorithm

Input : SEED KEY
Output : Secret keys (K_1, K_2, and K_3)

PRNG algorithm is generating a sequence of random numbers. Seed key is used to initialize a random number to generate the dynamic secret keys. Each time new random key is generated while running the application.

Output of the each round is given to input of the next round. The Fig. 5 depicts key generation using PRNG algorithm.

SEED KEY
Key generation (PRNG)
K_1, K_2, K_3

Fig. 5. Key Generation using Pseudo Random Number Generator (PRNG) Algorithm

iv) Key Encryption and Key Distribution

Input : K_1, K_2, and K_3

Output : K_1', K_2', K_3'

Key authority is encrypting the secret key with Key_Param (PU_{KA}, PU_{CS}) and distributing the encrypted secret keys to client (K_3'), decryption server (K_2'), cloud server (K_1'). String key pattern matching algorithm is used for key encryption process. In pattern matching check the sequences of data with some pattern among raw data. It is used to identify the matching pattern with another token sequence. It achieves security for the key during the key transmission. The Fig. 6 shows key encryption and key distribution processes by the key authority.

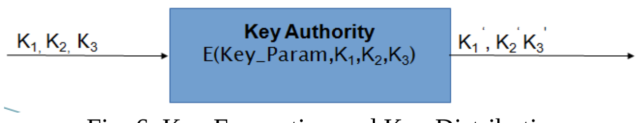


Fig. 6. Key Encryption and Key Distribution

v) Key Decryption and File Decryption

Input : K_1', K_2', K_3 and K_1 , K_2, K_3' and CT_1,CT_2

Output : K_1, K_2, K_3 & CT_1, PT

Keys K_1', K_2', K_3' are decrypted by cloud server, decryption server & client private key to get the keys K_1 ,K_2, K_3 while the key decryption process, client key, decryption server intermediate key and cloud server intermediate key are matched with string pattern matching process after the authentication process is completed. Finally, the keys are sent to client mail for the conformation. Using the cloud server key K_1, and decryption server key K_2, client key K_3, along with ciphertext CT_1 & CT_2 to decrypt to get the original plaintext PT. The Fig. 7 & Fig. 8 shows key decryption and file decryption processes in client, cloud server and decryption server respectively.

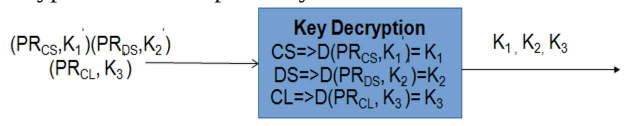


Fig. 7. Key Decryption

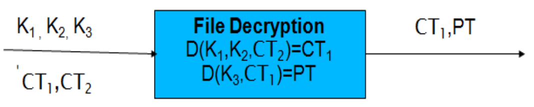


Fig. 8. File Decryption

F. Proposed Methodology

The secure collaborative key management system is to increase security for the data and key for the mobile cloud data sharing. For providing security to the data/file, the system uses RSA algorithm for encryption & decryption, PRNG algorithm for key generation. In PRNG, each time the new key is generated, so the attacker could not get the key and relevant data or file. PRNG algorithm achieves data confidentiality, backward and forward secrecy, revocation and also avoids key escrow and exposure problems. Key escrow is nothing but without permission of user to access the data & key exposure is the try to access data without all the keys.

G. Pseudo Random Number Generator (PRNG)

PRNG algorithm that uses mathematical formulas to produce sequences of random numbers. It generates a sequence of random numbers. The PRNG algorithm combines bit manipulation and bitwise logical operation to produce the random key as secret key. The output of key is given to the input of the second round, so the intruder could not easily captures the secret key.

i) PRNG Algorithm

The algorithm 1 shows steps in Pseudo Random Number Generator algorithm.

Algorithm 1. PRNG Algorithm

Step 1: Initially, the input key values are replaced with binary values as 0 and 1. If the key consists of n bytes, then the key length should be 4*n.

Step 2: Bitwise XOR operation is calculated on two successive bytes in block, i.e. (1st XOR 2nd) replaces the 1st byte, (2nd XOR 3rd) replaces 2nd byte, etc., The process continues until the last byte is XOR'ed with first byte or (nth XOR 1st) replaces the nth byte.

Step 3: The consecutive bytes are transformed with each other. Suppose, if 'n' is odd number, then the last byte is unchanged.

Step 4: Finally, the resulting bit value of the previous step is separated into halve. Left and right halves, each with 4*n bits length. The final bit sequence of 8*n can be considered as first random key.

Step 5: The first and last bytes are left unaltered and intermediate bytes are exchanged.

Step 6: The produced key in step 5 will be fed as an input to step 2. This process will generate next random key for different communication.

Step 7: The step 2 to step 5 can be repeated in order to generate more number of random keys.

The Fig. 9 shows the process of PRNG algorithm.

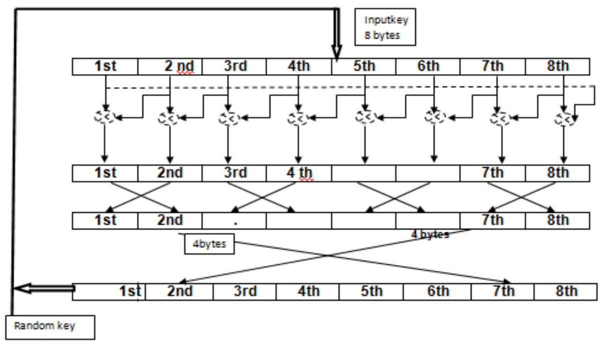


Fig. 9. File Decryption

The following example illustrates the implementation of PRNG algorithm for 32 bit key sequence. Let us assume that,

Step 1: Consider binary representation of the seed characters as:

1100 1010 0011 1001 1111 0100 0110 1010

Step 2: Produces:

0110 1001 1010 0110 1011 0010 1100 0110, then

Step 3: Produces:

1001 0110 0110 1010 0010 1011 0110 1100

Step 4: And finally, produces:

0010 1011 0110 1100 1001 0110 0110 1010

This is the generated pseudorandom number. For each round, the output of each stage is given to the input of the next round of processes.

H. File Encryption using RSA

RSA is an algorithm used for the file encryption and decryption. It is an asymmetric cryptographic algorithm. It provides the data confidentiality, integrity. RSA keys size is large, typically 512 bits, 1024, 2048, and 4096-bits long. For high-security applications, to keep data confidential, RSA recommends a key size larger than 2048 bits. For encrypting the file, the public key is used along with plaintext to get the ciphertext. RSA key length of 1024 bits is sufficient for many medium-security applications. In RSA, the key size is large so the intruder could not catch the secret key easily. Algorithm 1 shows steps in RSA encryption process.

Algorithm 2. RSA Encryption Process

```
Step 1: Choose n: n = p * q, start with two prime numbers, p and q.
Step 2: Calculate F(n): F(n) = (p-1)(q-1)
Step 3: Choose e (public key for data communication)
Step 4: d & n must be relatively prime (i.e., gcd (d,n) = 1)
Step 5: Perform Encryption:   $C = P^e \bmod n$
```

I. **File Decryption using RSA**

In the decryption process, respective private keys are used to get the original plaintext from ciphertext. Here the private key is the combination of client, cloud server and decryption server. Algorithm 3 shows steps in RSA decryption process.

Algorithm 3. RSA Decryption Process

```
Step 1: Choose n: n=p*q , Start with two prime numbers, p and q.
Step 2: Calculate F(n):    F(n): = (p-1)(q-1)
Step 3: Choose d:
Step 4: d & n must be relatively prime (i.e., gcd(d,n) = 1)
Step 5: Decryption:  $P = C^d \bmod n$
```

In above algorithm, the step 5 shows the decryption of ciphertext (C) using private key (d)

J. Secret Sharing Scheme (SSS)

Secret sharing scheme distributes a secret key/data among a set of participants, each participant having a share of the secret. Each participant share of the secret cannot construct the original secret key. The secret is accessed when specific conditions are fulfilled. The secret can only be reconstructed by grouping all participants' shares of the secret. For examples, the bank offering the locker services to the customer when they are having joint account. For the locker service in bank, they are generating three pin number, one for the bank manager and two for the customer 1 and customer 2. The secret is opened by three entities only when exact conditions are fulfilled. The locker will be opened, when the three members are grouped together, without loss of anyone's pin number.

K. Key Encryption

The secret key is generated randomly using PRNG algorithm. For each communication to access the file data, the secret key is produced without any repetitions. The secret key (K) is encrypted using key pattern matching algorithm.

i) Key Pattern Matching Algorithm

Pattern matching is used for identifying the matching pattern or substituting the matching pattern with another token sequence. String key pattern matching is the important scenario in pattern matching, either it may be in the form of tree structures. The key pattern lies the index position from 0 – 25. Using the pattern, the index partitions are identified. The patterns are changed dynamically to generate the index position. Finally the index positions are combined to get the secret key from three entities such as client, decryption server and cloud server.

L. Key Decryption

The key decryption is done by reverse of string key pattern matching algorithm. In decryption process, combination of the client, cloud server and decryption server's private keys are used to decrypt the encrypted key. It achieves the authentication of secret key. Secret key used to decrypt the encrypted file / data in proposed system.

Algorithm 4 explains key pattern matching algorithm for both key encryption and decryption processes.

Algorithm 4. Key Encryption and Key Decryption using Key Pattern Matching Algorithm

```
Step 1: BEGIN
Step 2: Initialize the input variables
            lt = Length of the Text
            lp = Length of the Pattern
            px = Prefix-function of pattern (p)
            c = Number of characters matched
Step 3: Get the input text elements
Step 4: Initialize the variable c=0, the starting index of the match
Step 5: In Key encryption,
        Evaluate the first letter of the pattern with first letter of the text
            If equivalent is not found then
                Replace with the value of px[c] to c
            If equivalent is found then
                Add the value of c by 1
Step 6: Confirm whether all the pattern letters are matched with the text characters
            If all the pattern letters are not matched with the text characters then
                Repeat the seek process
                Go to step 7.
            If all the pattern letters are matched with the text characters then
                Print the number of transfers taken by the pattern
                Print the starting index position of text element whenever the
                match is found
Step 7: Search for the next match
Step 8: Combine the key index values ($K_1'$, $K_2'$, $K_3'$) to get
        Key K
Step 9: Assume alphabets / special symbols for integers 0-25
Step 10: Get the corresponding letters for key index values
            Return $K_1'$, $K_2'$, $K_3'$
            Return Encrypted Secret Key K
Step 11: In Key decryption,
  Retrieve the corresponding letters using reverse key pattern matching algorithm Input key splitted into three equal parts (8 bit sequence) Match the key pattern for every splitted key Retrieve 4 bit sequence from the spitted Use initial pattern to match sequence
Step 12: Return decrypted key $K_1$, $K_2$, $K_3$
```

Step 13: END

The following example illustrates the implementation of key pattern matching algorithm with 24 bit key sequence.

Key Encryption

Input: TXODNkdTX3oNkdbTXkdvVODN

In Client,

Pat [ ] = ”TX”

Output: Pattern found at index 0
Pattern found at index 7
Pattern found at index 15
K_1'= (0, 7, 15) // Intermediate key

In Decryption Server,

Pat[ ] = ”DN”

Output: Pattern found at index 3
Pattern found at index 9
Pattern found at index 22
K_2'= (3, 9, 22) // Intermediate key

In Cloud Server,

Pat [ ] = ”kd”

Output: Pattern found at index 5
Pattern found at index 12
Pattern found at index 17
K_3' = (5, 12, 17)

$\mathbf{K = K_1' + K_2' + K_3'}$

Output: K = (3, 9, 22, 5, 12, 17, 0, 7, 15)

By assumption, 0 - 25 → Alphabets / Special symbols

TABLE I. CONVERSION OF INTEGERS TO SPECIAL CHARACTERS

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
!	@	#	%	^	&	*	(	)	_	+	=	-	\	]	[	}	{	?	/	.	>	,	<	|

K_1' = %_,

K_2' = &-{

K_3' = !([

K = TXODNkdTXDNkdbTXkdvVODNa //Encrypted Secret Key

Key Decryption

K_1 =TXOD // Original Key 1

K_2 =DNkd // Original Key 2

K_3 =kdvV // Original Key 3

In above example, consider input key size is 24 bits. The string key pattern matching algorithm is used for matching the key. Input keys are splited into three keys like K_1', K_2' and K_3'. Here we are assuming 0-25 index position values from TABLE I. These values are used to match the index position to get the encrypted secret keys K_1', K_2, and K_3'. In Key decryption, input key spitted into three equal part, from that spitted key, we match every key by their key pattern, to get the 4 bit continuous sequence, it’s for all the keys K_1, K_2, K_3 .By this process, we can get the 12 bit keys out off 24 bit keys and the reverse of the process to get the original key.

VI. EXPERIMENTAL RESULTS AND DISCUSSIONS

A. Experimental Setup

The proposed key management system is developed using software front tools such as HTML and JSP in NetBeans IDE 8.1. For backend design, JDK 1.8.0 and database connectivity with MySQL 5.0.22 are used. The connectivity between application server, database server and user interface are connected with Apache Tomcat 5.0/6.X server. DriveHQ cloud storage tool is used for mobile data storage in public cloud infrastructure. DriveHQ file manager makes remote storage for data files. The file manager transfer, access, share, synchronize, collaborate and publish the files remotely. The files are dragged and dropped to DriverHQ remote cloud storage. Using DriverHQ file manager, drag and drop files & folders. The system uses FTP and WebDAV connectivity using web browser. The free space availability for each user login is 1GB. Nearly, 200MB-800MB files are uploaded into cloud storage space. The complete proposed system is implemented in Windows 8 64 bit operating system with high-end hardware configurations.

The proposed work mainly focuses on accessing the data from client mobile devices. To access the file / data in mobile devices from cloud storage, initially the connectivity is established. The proposed system user interface URL is accessed from mobile devices (192.168.43.209:8084/MCC). So, data user can easily upload / store their data in public cloud storage by implementing security mechanisms. The file data are encrypted before the data are stored in cloud storage. To access the data within the mobile devices, the secret key is needed to decrypt the encrypted file. Thus the proposed system is implemented with file encryption, key generation, key encryption, key distribution, and key decryption.

B. Performance Metrics

The performance metrics measures and analysis the implementation of the key management protocol. The following are the performance metrics to evaluate the proposed system with existing techniques.

Integrity - The integrity of data in cloud and mobile devices.

Confidentiality - Authorized users to access the cloud data without affecting the originality of data.

Storage Overhead - The overhead on data stored in cloud (block-level and object-level storage) and mobile devices. The storage overhead is measured with number of files stored in different storage spaces in DriverHQ cloud.

Computation Overhead - The computation overhead depends on data storage, encryption and decryption processes, transmission and reception data rate in cloud storage.

Energy Consumption - Analyzes the energy consumption of mobile devices (thin clients) using computation overhead and storage overhead.

C. Experimental Results

Access the secure collaborative key management system for mobile cloud data storage in http://localhost:8084/MCC. The Fig. 10 shows data owner login page information to upload the data files to DriverHQ cloud storage.

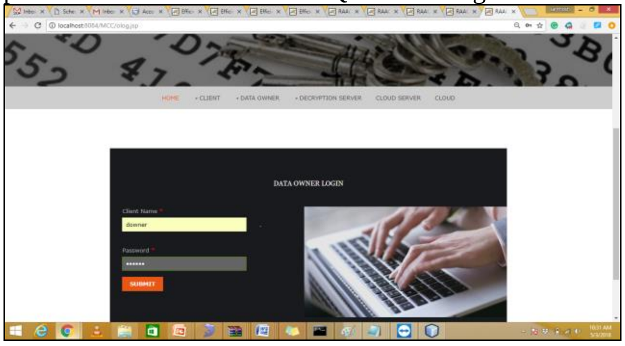


Fig. 10. Data Owner Login Page

The Fig. 11 shows file upload menu from data owner into DriverHQ cloud storage. Once the files are uploaded, the data owner can view the files.

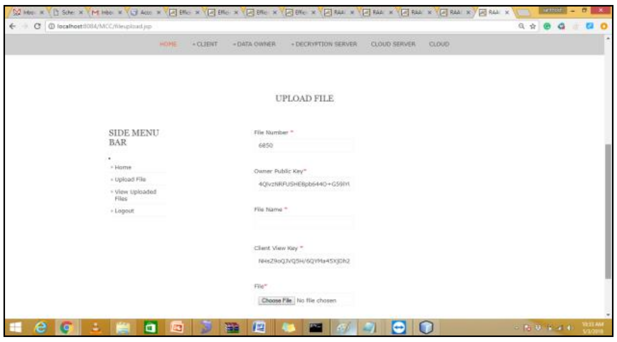
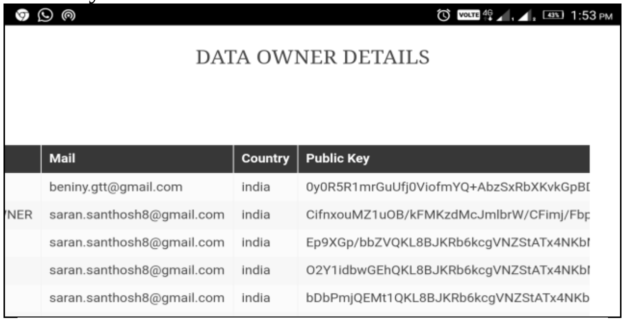


Fig. 11. File Upload

The Fig. 12 shows existing data owner details with the following information: Date of access, public key of the data owner to encrypt the file. In our proposed system, initially the file is encrypted using Key_Param. Key_Param is the combination of key authority's public key and cloud server's public key. The owner of the data can see and access their public key from mobile devices itself.



DATA OWNER DETAILS

	Mail	Country	Public Key
	beniny.gtt@gmail.com	india	0yOR5R1mrGuUfj0ViofmYQ+AbzSxRbXKvkGpBI
NER	saran.santhosh8@gmail.com	india	CifnxouMZ1uOB/kFMKzdMcJmlbrW/CFimj/Fbp
	saran.santhosh8@gmail.com	india	Ep9XGp/bbZVQKL8BJKRb6kcgVNZStATx4NKbI
	saran.santhosh8@gmail.com	india	O2Y1idbwGEhQKL8BJKRb6kcgVNZStATx4NKbI
	saran.santhosh8@gmail.com	india	bDbPmjQEMt1QKL8BJKRb6kcgVNZStATx4NKb

Fig. 12. Accessing Data Owner Public Key Value from mobile device

The Fig. 13 depicts DriverHQ storage space once the files are uploaded into public cloud storage. All the files are stored in encrypted format only. It can achieve data integrity on data owner's information. Only the authorized user can only access the information from cloud storage by collaborating secret key shares.

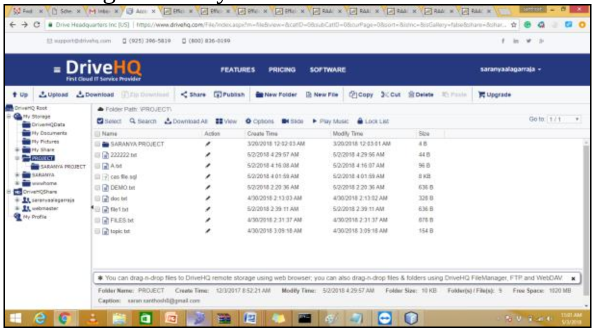


Fig. 13. DriverHQ Storage

The Fig. 14 shows encrypted files are in DriverHQ file folder.

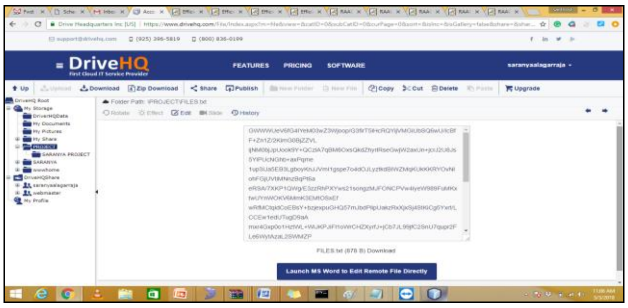
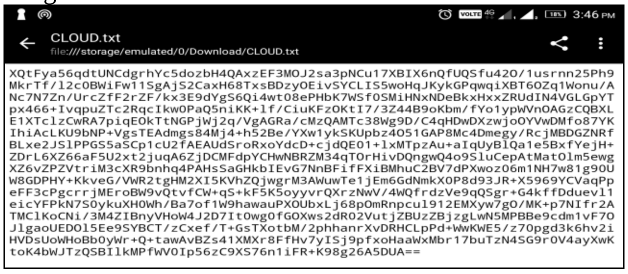

DriveHQ
First Cloud IT Service Provider
FEATURES
PRICING
SOFTWARE
Download
Zip Download
Share
Publish
Copy
Cut
Delete
Upgrade
Folder Path: \PROJECT\FILES.txt
Launch MS Word to Edit Remote File Directly

Fig. 14. Encrypted File in DriverHQ Cloud Storage

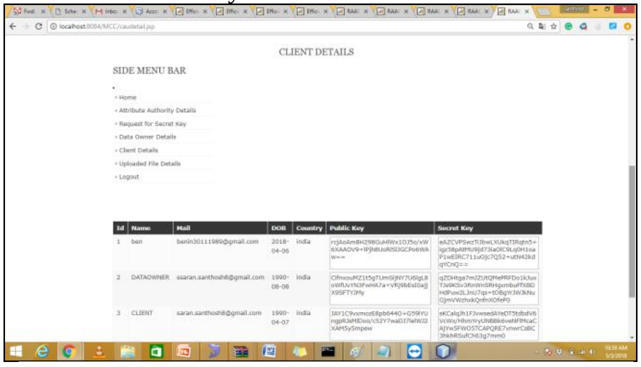
The Fig. 15 shows encrypted file in DriverHQ cloud storage via mobile.



```
XQtFya56qdtUNCdgrhYc5dozbH4QAxzEF3MOJ2sa3pNCu17XBIX6nQfUQSfu42O/1usrnn25Ph9
MkrTf/l2cOBWiFw11SgAjS2CaxH68TxsBDzyOEivSYCLIS5woHqJKykGPqwqiXBT6OZq1Wonu/A
Nc7N7Zn/UrcZfF2rZF/kx3E9dYgS6Qi4wt08ePHbK7WSf0SMiHNxNDeBkxHxxZRUdIN4VGLGpYT
px466+IvqpuZTc2RqcIkwOPaQ5niKK+lf/CiuKFzOKtI7/3Z44B9oKbm/fYo1ypWVnOAGzCQBXL
E1XTclzCwRA7piqEOkTtNGPjWj2q/VgAGRa/cMzQAMTc38Wg9D/C4qHDwDXzwjoOYVwDMfo87YK
IhiAcLKU9bNP+VgsTEAdmgs84Mj4+h52Be/YXw1ykSKUpbz4O51GAP8Mc4Dmegy/RcjMBDGZNRf
BLxe2JSlPPGS5aSCp1cU2fAEAUdSroRxoYdcD+cjdQE01+lxMTpzAu+aIqUyBlQa1e5BxfYejH+
ZDrL6XZ66aF5U2xt2juqA6ZjDCMFdpYCHwNBRZM34qTOrHivDQngwQ4o9SluCepAtMatOlm5ewg
XZ6vZPZVtriM3cXR9bnhq4PAHsSaGHkbIEvG7NnBFifFXiBMhuC2BV7dPXwoz06m1NH7w81g90U
W8GDPHY+KkveG/VWR2tgHM2XI5KVhZQjwgrM3AWuwTe1jEm6GdNmkXOP8d93JR+X5969YCVaqPp
eFF3cPgcrrjMEroBW9vQtvfCW+qS+kF5K5oyyvrQXrzNwV/4WQfrdzVe9qQSgr+G4kffDduevl1
eicYFPkN7SOykuXHOWh/Ba7of1W9hawauPXOUbxLj68pOmRnpcul912EMXyw7gO/MK+p7NIfr2A
TMClKoCNi/3M4ZIBnyVHoW4J2D7It0wg0fGOXws2dRO2VutjZBUzZBjzgLwN5MPBBe9cdm1vF7O
JlgaoUEDOl5Ee9SYBCT/zCxef/T+GsTXotbM/2phhanrXvDRHCLpPd+WwKWE5/z70pgd3k6hv2i
HVDsUoWHoBbOyWr+Q+tawAvBZs41XMXr8FfHv7yISj9pfxoHaaWxMbr17buTzN4SG9rOV4ayXwK
toK4bWJTzQSBIlkMPfWVOIp56zC9XS76n1iFR+K98g26A5DUA==
```

Fig. 15. Encrypted File in DriverHQ Cloud Storage

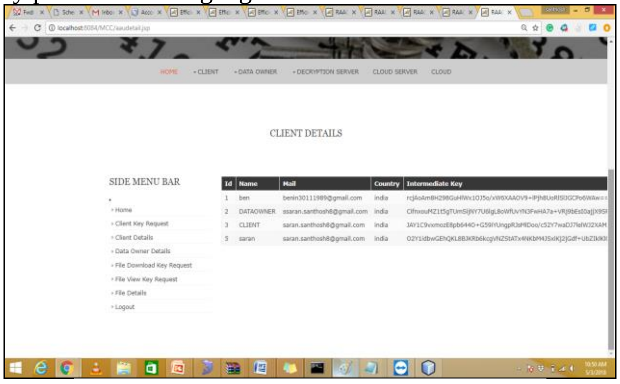
The Fig. 16 shows Client (data user) details with Public Key & Secret key pairs. The secret key is requested by the client while requesting to access the file from cloud storage. Thus, the client's key request is sent to key authority. Once the keys are generated, key verification process is initiated to validate the secret key.



Id	Name	Mail	DOB	Country	Public Key	Secret Key
1	ben	benin30111989@gmail.com	2018-04-06	india	rcjAoAmBH298GuHlWx1OJ5o/xW6XAAOV9+lPjh8UoRlSlJGCPo6WAw==	eAZCVPSwzTlJbwLXUkqTIRqtn5+igz58pAlMU9jd73laOlC9LqOH1oaP1wEIRC711uOjc7Q52+utN42kdqYCnQ==
2	DATAOWNER	ssaran.santhosh8@gmail.com	1990-08-08	india	CifnxouMZ1t5gTUmSIjNY7U6lgL8oWfUvYN3FwHA7a+VRj9bEsI0ajjX9SFTYJMy	qZDHtga7mJZUtQMePRFDo1kJsxTJs9KSv3RnWn5RHgxmbuFfXBDHdPuw2LJnU7qs+tOBgYrJWJkNuGjmVWzhxkQnfnXOfeP0
3	CLIENT	saran.santhosh8@gmail.com	1990-04-07	india	JAY1C9vxmozE8pb644O+G59lYUngpRJsMlDoo/c52Y7waDJ7lelWJ2XAM5y5mpew	sKCalqJh1FJvwsedAYeDT5tdbdV6VcWx/HhmYryUNBBk6veNFlMcaCAJYw5FWOSTCAPQRE7vnwrCsBC3hkhRSufCh63g7mm0

Fig. 16. Client details with Public Key & Secret key pairs

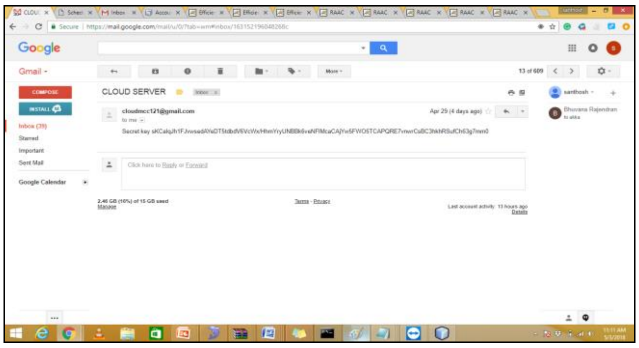
The Fig. 17 shows client details with Intermediate key. The intermediate keys are generated to ensure the data confidenality. The keys are sent to decryption server to collaborate and to get the final encrypted secret key using key pattern matching algorithm.



Id	Name	Mail	Country	Intermediate Key
1	ben	benin30111989@gmail.com	india	rcjAoAmBH298GuHlWx1OJ5o/xW6XAAOV9+lPjh8UoRlSlJGCPo6WAw==
2	DATAOWNER	ssaran.santhosh8@gmail.com	india	CifnxouMZ1t5gTUmSIjNY7U6lgL8oWfUvYN3FwHA7a+VRj9bEsI0ajjX9S
3	CLIENT	saran.santhosh8@gmail.com	india	JAY1C9vxmozE8pb644O+G59lYUngpRJsMlDoo/c52Y7waDJ7lelWJ2XAM
5	saran	saran.santhosh8@gmail.com	india	O2Y1idbwGEhQKL8BJKRb6kcgVNZStATx4NKbM4J5xlKJ2JGdf+UbZIklKl

Fig. 17. Client details with Intermediate key

The Fig. 18 shows that the secret keys are generated and sent to client's mail. Only the authorized data owner can only get the secret key. Thus, the proposed system achieves both data confidentliaty and data integrity for mobile data information in public cloud storage. The generated secret keys are again verified by key verification process before the original file (plaintext) is accessed from cloud storage.


CLOUD SERVER

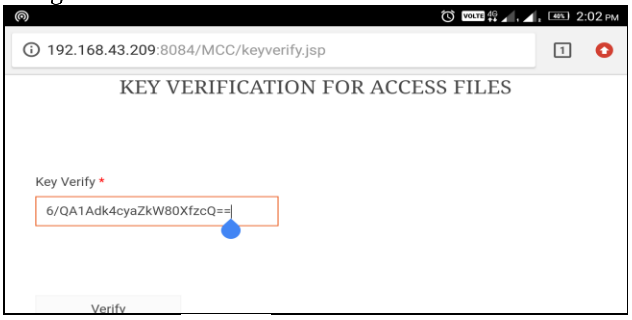
cloudmcc121@gmail.com
to me
Apr 29 (4 days ago)

Secret key sKCalqJh1FJvwsedAYeDT5tdbdV6VcWx/HhmYryUNBBk6vaNFIMcaCAjYw5FWOSTCAPQRE7vnwrCsBC3hkhRSufCh53g7mm0

Click here to Reply or Forward

Fig. 18. Secret Key in Client Mail

The secret key is generated to access the decrypted file stored in cloud storage. In proposed system, the generated secret keys are verified by using key verification process. The Fig. 19 shows key verification process to access files through mobile devices.


192.168.43.209:8084/MCC/keyverify.jsp

KEY VERIFICATION FOR ACCESS FILES

Key Verify *

6/QA1Adk4cyaZkW80XfzcQ==

Verify

Fig. 19. Key Verification

D. Results and Analysis

RSA: RSA is an asymmetric or public key cryptographic algorithm. The algorithm is used for file encryption and decryption. It provides the confidentiality, integrity, authenticity and non-reputability of electronic communications and data storage. RSA keys are typically 512 bits, 1024 or 2048 bits, 4096 bits long. It can create faster, smaller and more efficient cryptographic keys to secure data transmission over untrusted communication.

PRNG: PRNG algorithm is used for key generation. The algorithm generates the random keys for each communication. The random keys are used to secure the data in open space.

DriverHQ: DriverHQ is the public cloud storage service for repository of data. It's an Infrastructure as a Service (IaaS) for storage access. The files are uploaded into cloud storage and can easily access, manage, share or publish files from anywhere and at any time.

The proposed Secure Collaborative Key Management System (SCKMS) is compared with the existing Key Management System (KMS) in terms of data integrity, data confidentiality, storage overhead, computation overhead and energy consumption of mobile devices. Moreover, the performance of the proposed system is evaluated over varying number of files, number of users, storage spaces.

i) Data Integrity Vs Number of Files

The Fig. 20 shows the data integrity of the proposed by varying the numbers of files are accessed from mobile devices to cloud and vice versa. The graph clearly shows that data integrity is achieved when numbers of files are increased. Initially, the system considers 50 files in the form of text documents. In SCKMS, the data integrity is achieved by performing file encryption and file decryption using RSA public key cryptography algorithm, key encryption and key decryption using secret sharing scheme with key pattern matching algorithm. Meanwhile, the secret keys are generated using PRNG algorithm. It generates sequence of random keys as secret keys for each data file access. The key pattern matching algorithm authenticates key requests of data user, decryption server, and cloud server. This eliminates the intruder to access the files in cloud and mobile devices.

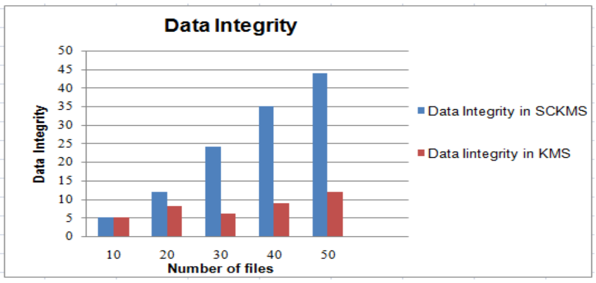


Fig. 20. Data Integrity Vs Number of files

ii) Data Confidentiality Vs Number of Files

The Fig. 21 shows the data confidentiality of proposed and existing systems when numbers of files are accessed from mobile devices to cloud and cloud to mobile devices. The authorized users can access the data at anytime from anywhere. Initially, the system considers 50 files for data access. In SCKMS, data owner, key authority and cloud server are having the authorized access. The security access is not stored in any storage spaces. Every time while login from mobile devices, the system has to authenticate the valid user. In addition, the authorized users are uploading and downloading the data files to and from public cloud storage without intervention of third party authorities. Meanwhile, the data confidentiality intern achieves data integrity in cloud.

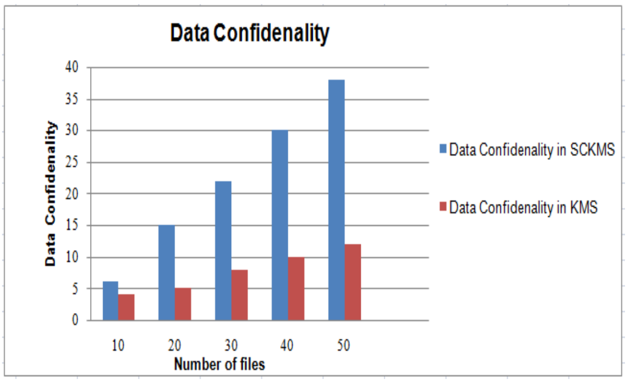


Fig. 21. Data Confidentiality Vs Number of files

iii) Data Confidentiality Vs Number of Users

The Fig. 22 shows the data confidentiality of both SCKMS and KMS when numbers of users are varying in mobile cloud. Initially, the system considers 50 users to access the data from DriverHQ cloud storage. Only the authorized users are accessing the data from cloud storage by using secret key generated by key authority using PRNG algorithm. In collaborative key access, the client key request is forwarded to decryption server and cloud server and final secret key is sent to client's mail.

Once the key is generated, the key verification process is done for confirmation. This achieves data confidenality in client, decryption server and cloud server.

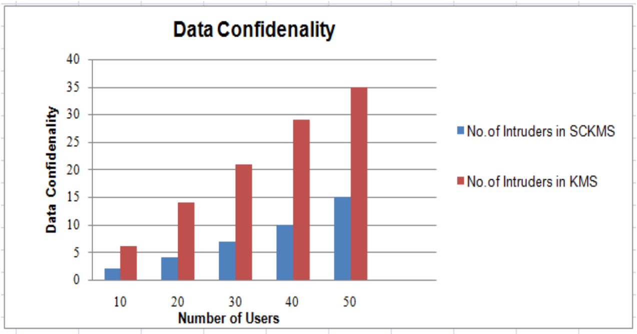


Fig. 22. Data Confidentiality Vs Number of Users

iv) Computation Overhead Vs Number of Files

The computation overhead depends on data storage, encryption and decryption processes, transmission and reception data rate in cloud storage. The numbers of files are varying from 10-50 files. The memory spaces are considered from 100MB-1000MB. The Fig. 23 depicts computation overhead by accessing various numbers of files. Compare to existing key management system, the proposed system minimizes computation overhead by offloading files from mobile devices to cloud storage. The cloud server can access, manipulate and store data without utilizing mobile storage space.

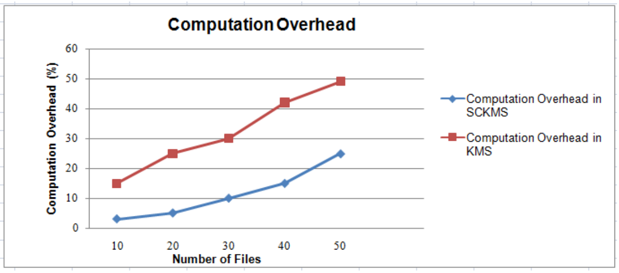


Fig. 23. Computation Overhead Vs Number of Files

v) Storage Overhead Vs Number of Files

The overhead on data stored in cloud (block-level and object-level storage) and mobile devices. The storage overhead depends on memory spaces in cloud infrastructure. The memory spaces vary from 100MB-1000MB. Initially, 50 text documents are considered for storage. The Fig. 24 depicts storage overhead by accessing and manipulating number of files in mobile cloud. The resource and storage intensive files and applications are stored in cloud storage, not in mobile devices (offloading). The offloaded applications are accessed in mobile devices. This eliminates storage overhead in proposed secure collaborative key management system.

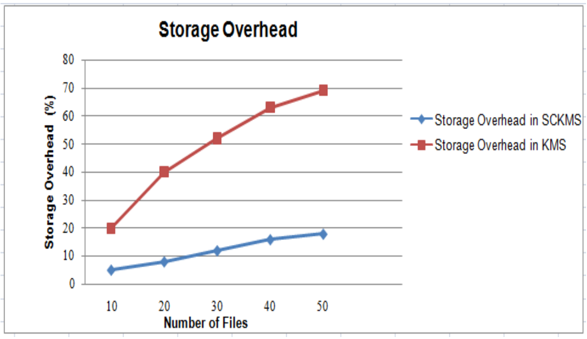


Fig. 24. Storage Overhead Vs Number of Files

vi) Energy Consumption Vs Number of Files

The energy consumption in mobile devices is analyzed based on computation overhead and storage overhead. The Fig. 25 shows energy consumption of mobile devices when compared to existing techniques. The existing key management system is not effectively utilized and accessed the storage spaces. The compute-intensive applications are stored in thin client device itself. Many of the storage spaces are wasted based on disk scheduling and memory allocation techniques. This will increase the computation and storage overhead, intern increases energy consumption in mobile devices. The proposed system implements offloading concepts with data integrity and data confidentiality in mobile devices.

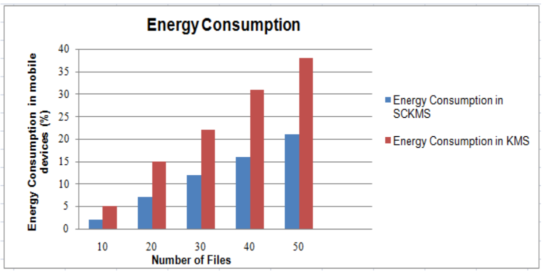


Fig. 25. Energy Consumption Vs Number of Files

VII. CONCLUSION

Mobile Cloud computing is cost-effective, on demand network access, resource for sharing the data, more convenient from anywhere and anytime. Therefore, enforcing the protection against the personal, confidential and sensitive data stored in the cloud is extremely crucial. The simultaneous participation of a large number of users requires fine-grained access control for data sharing in mobile cloud environment. In MCC, one of the primary concern is the security and privacy of data stored in cloud. The existing techniques fail to manage key and data. The proposed mechanism perfectly addresses not only key escrow problem but also a worse problem called key exposure. The objective of this work is to develop an secure collaborative key management system for mobile cloud data storage by implementing file encryption, key generation, key distribution (key encryption and key decryption) and file decryption methods. For the data storage DriverHQ public cloud infrastructure is used. It can be easily accessed, managed, share or publish the files from anywhere at any time.

RSA algorithm is used for files encryption and decryption. The secret keys are generated by using PRNG algorithm that produces sequences of random numbers. For the key encryption and decryption processes, key pattern matching algorithm is implemented. In key decryption, intermediate keys of client, cloud server and decryption server are matched by pattern matching process. The algorithm achieves the security for the file data and key by eliminating key escrow & key exposure problems. Finally, data integrity and data confidentiality in mobile cloud storage is achieved and the system also minimizes computation and storage overheads in client mobile devices, and minimizes the energy consumption of mobile devices. The future work focuses on reducing ciphertext size in encryption, minimizing encryption and decryption cost and Feasibility to have multiple file formats.

REFERENCES

1. Hlatshwayo M, and Tranos Zuva, "Mobile Public Cloud Computing, Merits and Open Issues", International Conference on Advances in Computing and Communication Engineering (ICACCE), 2016.
2. Santosh Kumar, and Gouda R H, "Cloud Computing – Research Issues, Challenges, Architecture, Platforms and Applications: A Survey ", International Journal of Future Computer and Communication, Vol. 1, No. 4, December 2012.
3. Masoud Nosrati, Ronak Karimi H, "Mobile Computing: Principles, Devices and Operating Systems", World Applied Programming, Vol .2, No. 7, 2014.
4. Jia, and Cong Wang, "Enabling Cloud Storage Auditing With Verifiable Outsourcing of Key Updates", IEEE Transactions on Information Forensics and Security, Vol. 11, No. 6, 2016.
5. Pratima Popat Gutai, Rutuja S Kothe, and Jahveri S B, "Efficient Hierarchical Cloud Storage Data Access Structure with KDC", IEEE International Conference in Electronics, Communication and Computer Technology (ICAECCT), 2016.
6. Sikhar Patranabis, Yash Shrivastava, "Provably Secure Key-Aggregate Cryptosystems with Broadcast Aggregate Keys for Online Data Sharing on the Cloud", Vol. 66, No. 5, 2017.
7. Yuqi Wang, and Kun She, "A Practical Quantum Public-key Encryption Model", Third International Conference on Information Management (ICIM), 2017.
8. TalariBhanu Teja, Vootla Hemalatha, and Priyanka K, "Encryption And Decryption – Data Security For Cloud Computing – Using AES Algorithm", SSRG International Journal of Computer Trends and Technology (IJCTT) - Special Issue, April 2017.
9. Mahesh Kumar K M, and Sunitha N R, "Hybrid Cryptographically Secure Pseudo-Random Bit Generator", Second International Conference on Contemporary Computing and Informatics (IC3I), 2016.
10. Nasrollah Pakniat, Mahnaz Noroozi, and Ziba Eslami, "Distributed Key Generation Protocol with Hierarchical Threshold Access Structure", IET Information Security, Vol. 9, No. 4, 2015
11. Preeti Garg, and Vineet Sharma, "An Efficient and Secure Data Storage in Mobile Cloud Computing through RSA and Hash Function", International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014
12. Prasoon Raghav, Rahul Kumar, and Rajat Parashar, "Securing Data in Cloud Using AES Algorithm", International Journal of Engineering Science and Computing, Vol. 6, No. 4, 2016
13. Zheng Yan, Xueyun Li, Mingjun Wang, and Athanasios V Vasilakos, Senior Member, Vasilakos, "Flexible Data Access Control Based on Trust and Reputation in Cloud Computing", IEEE Transactions on Cloud Computing, Vol. 5, No. 3, 2017
14. Giwon Lee, Haneul Ko, and Sangheon Pack, "An Efficient Delta Synchronization Algorithm for Mobile Cloud Storage Applications", IEEE Transactions on Services Computing, Vol. 10, No 3, 2017.
15. Guanlin Wu, Junjie Chen, Weidong Bao, Xiaomin Zhu, Wenhua Xiao, Ji Wang, and Ling Liu, "Collaborative Storage Algorithm Based on Alternating Direction Method of Multipliers on Mobile Edge Cloud", IEEE International Conference on Edge Computing (EDGE), 2017
16. Debiao He, Neeraj Kumar, Mohammad Khurram Khan, Lina Wang, and Jian Shen, "Efficient Privacy-Aware Authentication Scheme for Mobile Cloud Computing Services", IEEE System Journal, Vol. 12, No. 2, 2018.
17. K. Vijayakumar and V. Govindaraj, "An Efficient Communication Technique for Extrication and Cloning of packets on cloud", International Journal of Applied Engineering Research, ISSN 0973-4562 Vol. 10 No.66 May 2015

AUTHORS PROFILE

Shakkeera is working as Assistant Professor (Senior Grade) at B. S. Abdur Rahman Crescent Institute of Science & Technology, Chennai, Tamil Nadu, India since 2006. She has received B.Tech. in Information Technology from Crescent Engineering College affiliated to Anna University, Tamilnadu, India in 2005, M.E in Computer Science and Engineering from B. S. Abdur Rahman Crecent Engineering College affiliated to Anna University, Tamilnadu, India in 2010. She has a teaching experience of 13 years. She has published more than 25 research publications in refereed International/National Journals and International/National Conferences. Her areas of specializations are Cloud Computing, Mobile Cloud Computing, Internet of Things, Network Security, Mobile Ad-Hoc Networks and Web Services.

Saranya is a student at B.S.Abdur Rahman Crescent Institute of Science & Technology, Chennai, Tamil Nadu, India. She received B.Tech Degree in Information Technology from Hindusthan College of Engineering and Technology, Coimbatore, in the year 2012. She is currently pursuing his M.Tech IT degree in B.S.Abdur Rahman Crescent Institute of Science & Technology, Vandalur. Her areas of interest are Cloud Computing, Network Security.

Sharmasth Vali is working as Assistant Professor at Dhanalakshmi College of Engineering, Chennai, Tamilnadu, India since 2013. He has received his B.Tech. in Computer Science and Engineering from Shadan College of Engineering and Technology, JNTU Hyderabad in 2007, M.E. in Computer Science and Engineering from B.S.Abdur Rahman Crescent Engineering College, Anna University in 2009. He is presently doing Ph.D in Information Technology from B. S. Abdur Rahman Crescent Institute of Science & Technology, Chennai, Tamil Nadu, India. He has a teaching experience of 9 years. He has published more than 10 research publications in refereed International/National Journals and International/National Conferences. His areas of specializations are Network Security, Intrusion Detection and Prevention Systems, cloud computing and Mobile Ad-Hoc Networks.