J Grid Computing
DOI 10.1007/s10723-015-9352-9

# A Cloud-Manager-Based Re-Encryption Scheme for Mobile Users in Cloud Environment: a Hybrid Approach

**Abdul Nasir Khan · M. L. Mat Kiah · Mazhar Ali · Shahaboddin Shamshirband · Atta ur Rehman Khan**

Received: 16 February 2015 / Accepted: 23 September 2015

**Abstract** Cloud computing is an emerging computing paradigm that offers on-demand, flexible, and elastic computational and storage services for the end-users. The small and medium-sized business organization having limited budget can enjoy the scalable services of the cloud. However, the migration of the organizational data on the cloud raises security and privacy issues. To keep the data confidential, the data should be encrypted using such cryptography method that provides fine-grained and efficient access for uploaded data without affecting the scalability of the system. In mobile cloud computing environment, the selected scheme should be computationally secure and must have capability for offloading computational intensive security operations on the cloud in a trusted mode due to the resource constraint mobile devices. The existing manager-based re-encryption and cloud-based re-encryption schemes are computationally secured and capable to offload the computationally intensive data access operations on the trusted entity/cloud. Despite the offloading of the data access operations in manager-based re-encryption and cloud-based re-encryption schemes, the mobile user still performs computationally intensive paring-based encryption and decryption operations using limited capabilities of mobile device. In this paper, we proposed Cloud-Manager-based Re-encryption Scheme (CMReS) that combines the characteristics of manager-based re-encryption and cloud-based re-encryption for providing the better security services with minimum processing burden on the mobile device. The experimental results indicate that the proposed cloud-manager-based re-encryption scheme shows significant improvement in turnaround time, energy consumption, and resources utilization on the mobile device as compared to existing re-encryption schemes.

**Keywords** Cloud computing · Mobile cloud computing · Security · Privacy

A. N. Khan · M. L. M. Kiah · S. Shamshirband (✉)
Faculty of Computer Science & Information Technology, University of Malaya, Kuala Lumpur, Malaysia
e-mail: shamshirband@um.edu.my

M. Ali
Department of Electrical and Computer Engineering, North Dakota State University, Fargo, USA

A. N. Khan · M. Ali
Department of Computer Science, COMSATS, Abbottabad, Pakistan

A. N. Khan
e-mail: anasir@ciit.net.pk

M. Ali
e-mail: mazhar@ciit.net.pk

A. R. Khan
College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia
e-mail: dr@attaurrehman.com

## 1 Introduction

The growth in mobile applications and emergence of cloud computing [1, 2] attracted the researchers for introducing the application models and data storage services for the mobile device that uses the remote cloud services. The integration of elastic cloud services [3, 4], ubiquitous wireless infrastructure, and resource constraint mobile devices laid foundation for new computing paradigm called mobile cloud computing [5–7]. The objectives of such integration are to reduce the power consumption on the mobile device, improve the storage/processing capacity of the mobile device, and enhance the reliability of mobile application and data. To achieve the aforementioned objectives, the new generation mobile applications offload the computational intensive jobs from the resource constraint mobile device to the resource enriched cloud that reduces the power consumption on mobile device in completing the execution of jobs. Moreover, the migration of computational intensive jobs on cloud completes execution quickly as compared to the local execution. Likewise, the utilization of the cloud storage service, such as Amazon S3 [8] and ZumoDrive [9], allows mobile user to store enormous amount of data on the cloud storage. The backup of data and mobile applications on multiple cloud servers improves the reliability of mobile user data and applications. The architecture of the mobile cloud computing is shown in Fig. 1.

The cloud client applications are developed using mobile application development platform and deployed on mobile devices. The cloud client applications utilize the mobile network services, such as wireless network (e.g. *Wi-Fi*, *Wi-Max)*, cellular network (e.g. *3G* or *4G),* or Satellite network for communicating with cloud controller. The cloud controller handles the mobile users requests for providing corresponding cloud services [10–14].

According to analysis presented in ABI Research and Juniper Research reports [15, 16], the mobile cloud computing subscribers and cloud-based mobile applications are rapidly increasing. Another study of the International Data Corporation (*IDC*) [17] concludes that the top most challenge due to which 74 % of business organizations are not willing for adoption of the cloud services is security and privacy. This can be concluded from the analysis of the aforementioned reports that the security and privacy improvement in cloud services may increase the cloud's subscribers. The important parameters that need to be considered while designing a security scheme for mobile cloud computing environment are computational complexity of security scheme and resource limitation of the mobile device. The computational secure schemes are based on complex and computational intensive operations that are not suitable for mobile device. Due to the resource capacity limitation of the mobile devices, the existing security schemes offload the computational intensive security operations on the third-party/cloud in a trusted mode. Alternatively, few security schemes are focusing on the reduction of the computational complexity of the cryptographic algorithms. However, the reduction of the computational complexity of cryptographic algorithms may affect the privacy of the uploaded data. For offloading of data access operations, the most of the existing schemes are based on proxy re-encryption. Although the proxy re-encryption schemes provide support for offloading of computationally intensive re-encryption operations, the mobile user has to perform the encryption and decryption that involve massive multiplication and exponential operations of large numbers.

In this paper, we proposed cloud-manager-based re-encryption scheme that utilizes the characteristics of the existing manager-based re-encryption and cloud-based re-encryption schemes [18] for distributing the computational tasks among mobile device, trusted-entity, and cloud. The Manager-based Re-encryption Scheme (*MReS*) offloads the computationally intensive data access operations on the trusted entity called manager for reducing the processing burden of the data access operations from the mobile device. To minimize the responsibilities of the manager in *MReS*, the Cloud-based Re-encryption Scheme (*CReS*) offloads the major portion of data access operations on the cloud without revealing the security keys and data contents. Although manager-based re-encryption and cloud-based re-encryption schemes offload the re-encryption tasks on trusted-entity/cloud, the mobile user has to perform the computationally intensive encryption and decryption. Moreover, in cloud-based re-encryption scheme, a single key is shared among all the group members of particular data partition on the cloud. Therefore, the existing/leaving group member can decrypt the uploaded data on the data partition of the cloud. For securing the scheme from the existing group member, the key is updated whenever there is a change in a group member of particular data partition. The side-effect of updating the key is that entire data partition needs to be re-encrypted using the cloud computational power. Furthermore, compromising of the legitimate group member affects the privacy of the whole system. By considering the limited capabilities of mobile devices, our proposed *CMReS* distributes encryption, decryption, and re-encryption operations among mobile device, trusted entity, and cloud. Moreover, the proposed scheme protects the data partition from **(a)** the existing/leaving group members and **(b)** misbehavior of current legitimate group member without introducing any processing overhead on the cloud unlike *CReS*.

The rest of the paper is organized as follows. Section 2 presents the existing security schemes proposed for mobile cloud computing environment. Section 3 covers detail description and limitation of the *MReS* and *CReS*. In section 4, we define the proposed cloud-manager-based re-encryption scheme. The results of the comprehensive empirical analysis with some critical remarks are presented in Section 5. We conclude our work in Section 6 and address future research directions.

## 2 Existing Security Schemes for Mobile Cloud Computing

Most of the existing security schemes that are design for mobile cloud computing environment are based on traditional cryptographic methods or pairing-based cryptography. The schemes presented in [19–21], and [22] provide the security features for mobile user in the cloud environment using the traditional cryptography methods. The rest of the security schemes discussed in [18, 23–25], and [26] are based on pairing-based cryptography for secure offloading of the data access operations on the cloud/third-party in a trusted mode. The classification of the security schemes for mobile cloud computing is shown in Fig. 2.

Few of the schemes presented in the classification execute the entire security operations on the mobile device. The rest of the schemes delegate the security management operations on the cloud, trusted entity under the control of client organization, or trusted entity under the control of the third party. In the category of local execution, a small number of schemes focus on the reduction of computational complexity of cryptographic algorithms for reducing the processing burden from the mobile device. In offloading

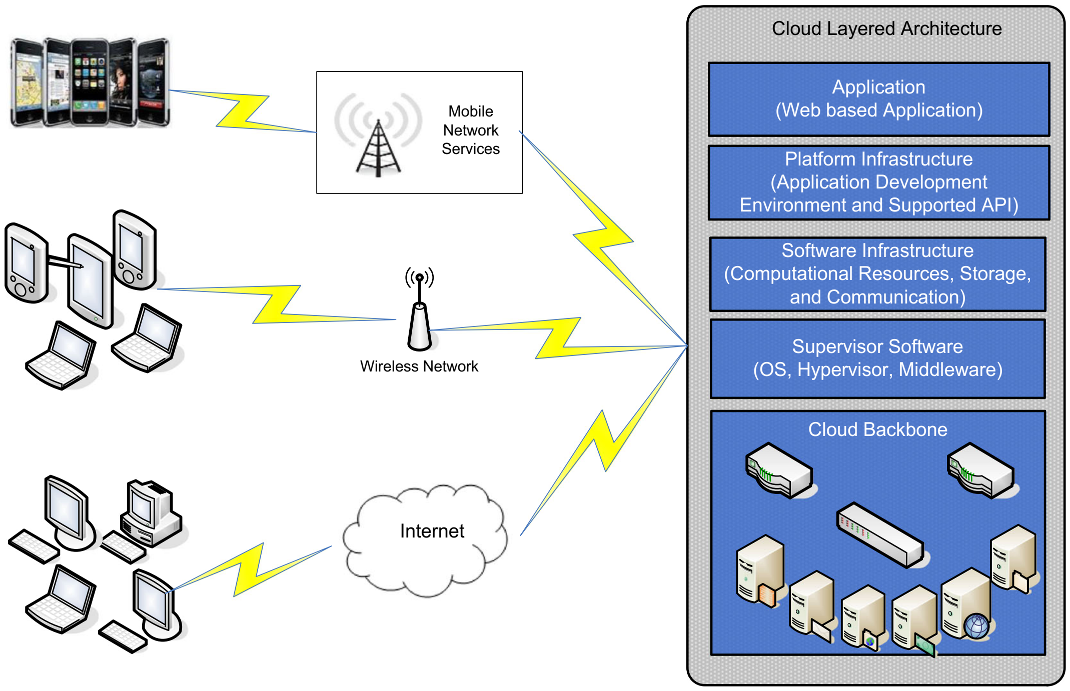


**Fig. 1** Architecture of mobile cloud computing

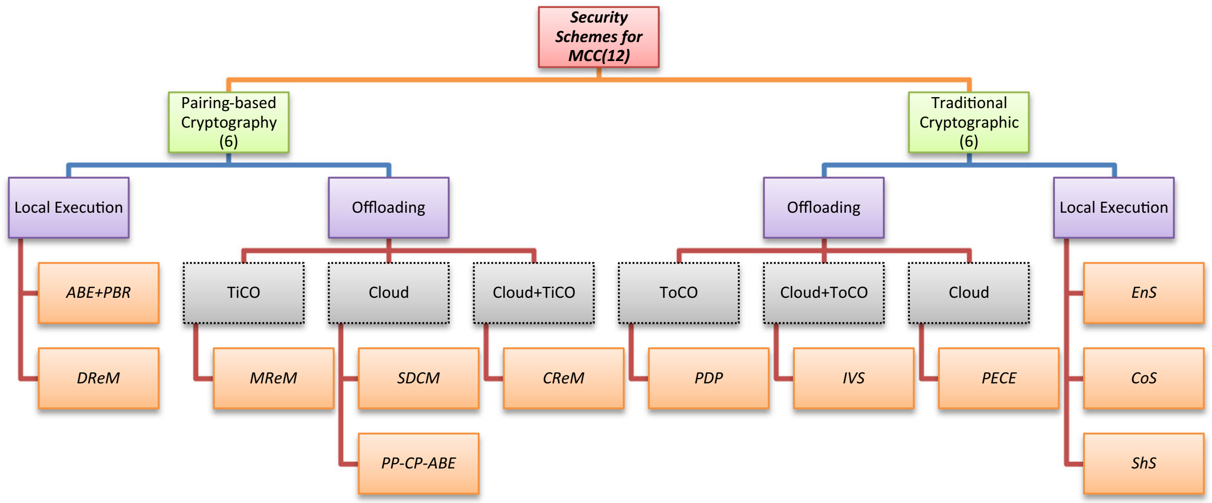


**Fig. 2** Classification of data security schemes for mobile cloud computing environment

category, all existing security schemes offload the computational intensive security operations on the cloud, trusted entity, or combination of cloud and trusted entity for reducing the processing burden from mobile device.

The authors in [19] proposed a method for trusted data sharing on untrusted cloud servers using Progressive Elliptic Curve Encryption (*PECE*) scheme. The *PECE* allows the encryption of message multiple times with different keys that can be decrypted in a single run with a single key. During read operation, the data owner grants the access permission to the requesting entity by sending the credential to the cloud service provider and requesting entity. The cloud service provider re-encrypts the uploaded ciphertext using the received credential from the data owner that can be decrypted with a credential delivered to the requesting entity. Subsequently, the cloud service provider delivers the re-encrypted message to the requesting entity. The requesting entity decrypts the message using received credential from the data owner. The involvement of the data owner/ data receiver in performing multiple encryption, decryption, and re-encryption makes this scheme infeasible for the mobile cloud computing environment. In [20], Yang et al. proposed a Provable Data Possession (*PDP*) scheme for ensuring privacy and confidentiality of the mobile users in a cloud environment with the help of a trusted entity under the control of third-party. The trusted entity executes encoding/decoding, encryption/decryption, signatures generation, and verification on behalf of mobile users. The involvement of trusted entity under the control of third-party for the execution of mobile users' jobs may degrade the performance of the system with increase in number of mobile users. The authors in [21] introduce Integrity Verification Scheme (IVS) for mobile users in the cloud environment by using the services of the trusted entity under the control of third-party. The mobile user offloads integrity verification operations on the cloud and trusted-entity. However, the privacy of mobile user is overlooked in the proposed scheme. Moreover, the involvement of trusted entity under the control of third-party may degrade the performance of the system with increase in mobile users attached with the trusted entity. In [22], authors proposed three schemes for providing confidentiality and integrity services for the mobile users in a cloud environment. The focus of the proposed schemes is to reduce the computational complexity of the cryptography operations instead of offloading the computationally intensive operations on the cloud. In the first scheme, author uses standard cryptographic operation for providing confidentiality and integrity services to the mobile users in the cloud environment. However, the execution of the traditional cryptographic operations on the mobile device consumes considerable amount of resources. To remove the limitation of the first scheme, the authors in [22] introduce two more schemes that achieve the confidentiality using matrices multiplication and *Exclusive-OR* operations. Although the reduction of the computational complexity of the cryptography algorithm makes scheme suitable for mobile cloud computing environment, the involvement of light-weight operations may results in compromised privacy. The aforementioned limitations of the schemes presented in [22] are removed in [27].

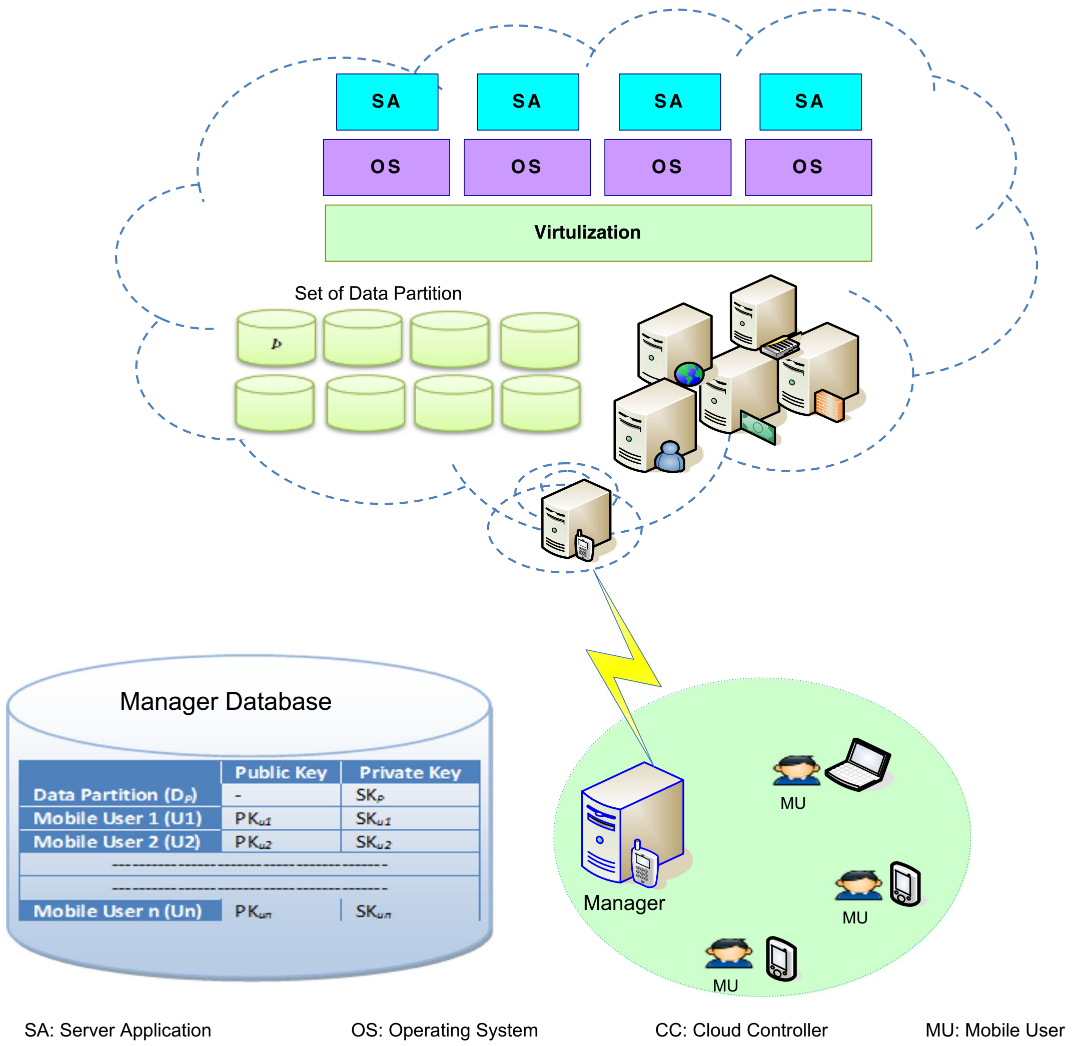
The authors in [23] introduce a merger of attribute-based encryption and proxy-based re-encryption [28] for offloading the computational intensive data access operations on an untrusted cloud server without disclosing any data contents. The proposed scheme is not suitable for mobile cloud environment due to the involvement of heavy computational load on the data owner. The data owner has to generate keys for new/existing user that join/leave the system and re-encrypt the affected files. Another merger of proxy re-encryption [29] and identity base encryption schemes [30] was introduced in [24] called Secure Data Service Mechanism (*SDSM*) that outsource data and security management on the cloud without disclosing any user information. The offloading of data and security management operations on the cloud without disclosing the security keys and data contents improve the security of the system with minimum processing overhead on the mobile device. Similarly, a Privacy Preserving Cipher Policy Attribute-Based Encryption (*PP-CP-ABE*) is introduced in [25] for ensuring the privacy of the mobile user data on untrusted cloud servers. Most of the computational intensive and storage demanding security operations are migrated on the cloud without affecting the user privacy. The size of the ciphertext grows linearly with increase in number of ciphertext attributes [31] that involves more pairing evaluation and exponential operations while decrypting the ciphertext. To make the scheme suitable for mobile cloud computing environment, the scheme transform plaintext into fixed ciphertext regardless of ciphertext attributes. Ateniese et al. [18, 26] introduces a scalable and efficient key distribution scheme for the resource constraint mobile devices in the cloud environment. The proposed scheme achieves data confidentiality for the mobile users using the proxy re-encryption scheme. The proxy re-encryption helps the mobile user to delegate the data access operation on the trusted-entity under the control of client organization without disclosing the users' contents. Tysowski et al. [18] proposed a Cloud-based Re-encryption Scheme (*CReS*) that reduces the responsibility of the trusted entity assigned in Manager-based Re-encryption Scheme (*MReS*) [18, 26] for improving the scalability of the system. The trusted entity involved in [18, 26] is under the control of client organization and handles the security related operations on behalf of the organizational employees. The association of the limited users with trusted entity improves the scalability of the system.

Although the aforementioned offloading schemes reduce the processing burden from the resource constraint mobile device, there are few issues that need to be considered. In [23], the data owner must be available for providing access privileges to the requesting entity that makes the scheme impractical for mobile cloud computing environment due to the mobility of the mobile users. The schemes presented in [18, 24], and [22] offload the data access operations on the cloud/third-party in trusted mode. However, the data owner has to transform the plaintext into ciphertext, and vice versa that involves execution of computationally intensive multiplication and exponential operations of large numbers on resource constraint mobile device.

By keeping in view the resource limitation of mobile device, the proposed *CMReS* distributes encryption, decryption, and re-encryption operations on mobile device, trusted entity under the control of client organization, and cloud. Our proposed scheme presented in this paper is based on *MReS* and *CReS* techniques defined by Tysowski and Hasan in [18]. Both schemes are based on the proxy and cloud re-encryption paradigms and can be analyzed as a combination of well-known atomic *BBS* re-encryption [32], Ateniese, Fu, Green and Hohenberger algorithm (AFGH) [26], SHA2 function [33], and data partition encryption in the cloud [34]. In next section, we first formulate the basic definitions of the proxy re-encryption, *BBS* scheme and bilinear maps, and then characterize both *MReS* and *CReS* methodologies with a simple critical analysis.

## 3 MReS and CReS Re-Encryption Strategies

In 1998 Blaze, Bleumer, and Strauss [32] have introduced an atomic re-encryption scheme (*BBS*), based on El-Gamal cryptosystem [35] and the idea of re-encryption key $rk_{A\rightarrow B}$ between parties $A$ (Alice) and $B$ (Bob), which allows re-encryption from one secret key to another without ever learning the plaintext. To achieve the re-encryption properties, both *MReS* and



| | Public Key | Private Key |
|---|---|---|
| Data Partition ($D_P$) | - | $SK_P$ |
| Mobile User 1 (U1) | $PK_{u1}$ | $SK_{u1}$ |
| Mobile User 2 (U2) | $PK_{u2}$ | $SK_{u2}$ |
| ------------ | ------------ | ------------ |
| ------------ | ------------ | ------------ |
| Mobile User n (Un) | $PK_{un}$ | $SK_{un}$ |

**Fig. 3** System model for manager-based re-encryption

*CReS* are based on the concept of a bilinear mapping. The central idea is the construction of a mapping between two cryptographic groups which allows to reduce the complexity of the whole problem and its transformation to a different, usually easier problem in the other group.

**Definition 1** Let $G_1$, $G_2$, and $G_3$ be the cyclic groups of the same order $q$.[1] A function $e : G_1 \times G_2 \rightarrow G_3$ is *a bilinear map,* if the following condition is satisfied:

$$\forall g \in G_1, h \in G_2; \quad a, b \in Z_q \quad e\left(g^a, h^b\right) = e\left(g, h\right)^{ab}, \tag{1}$$

where$Z_q^* = \{1, \ldots, q-1\}$.

In cryptography, bilinear maps are used as major pairing-based constructs. It is assumed additionally that mapping $e$ is non-degenerate and efficiently computable, which means that $e$ is *an Admissible Bilinear Map (ABM)* according to the following definition:

**Definition 2** Let $e : G_1 \times G_2 \rightarrow G_3$ be a bilinear map and$g \in G_1, h \in G_2$.The mape*isan admissible* bilinear*map (ABM)* if

[1]In most of the cryptographic approaches $\boldsymbol{q}$ is considered to be a ***prime*** primitive.

a) $e$is non-degenerate, i.e. if $g$ generates $G_1(< g >= G_1)$ and $h$ generates $G_2$ $(< h >= G_2)$, then $e(g, h)$ generates $G_3(< e(g, h) >= G_3)$;
b) $e$ is efficiently computable (possibly in a polynomial time); and
c) there exists an isomorphism from $G_1$ to $G_2$ (we can assume then that $G_1 = G_2$).

From now on we will focus on just *ABM* maps in our analysis.

## 3.1 Manager-Based Re-Encryption Scheme

A manager-based re-encryption scheme (*MReS*) defined in [18] achieves the data confidentiality for the mobile users by using the proxy re-encryption strategy. The proxy re-encryption helps the mobile user to delegate the data access operation on third-party. *MReS* methodology can be implemented in the system, the model of which is presented in Fig. 3. There are three main groups of users in this system, namely **(a)** mobile users, **(b)** cloud service provider, and **(c)** manager as a ***fully*** trusted entity. The cloud service providers offer computational and storage services to the mobile users. The mobile users upload/download data to/from the data partition of the cloud with the assistance of the manager. In [18], the manager is considered as a ***fully*** trusted entity under the control of client organization that is responsible for generation and distribution of the public and secret keys for the mobile users and data partition. Tysowski and Hasan assume also in [18] that the manager can hold the private keys of the cloud users belonging to the same virtual organization (VO) and use them for the re-encryption.

Following the characteristics presented in [18], the *MReS* can be defined as follows:

*Setup* *MReS* operates over two groups $G_1$ and $G_2$ of prime order $q$ with a bilinear map $e : G_1 \times G_1 \to G_2$. The system parameters are random generators $g \in G_1$ and $Z = e\ (g,\ g) \in G_2$,the users belong to the virtual organization $U_P$ and have an authorized access to a data partition $P$ in the cloud.

*Key Generation* The manager generates the private and public keys for the authorized users and data partition. A pair of secret and public keys $(SK_i, PK_i)$ for the $i^{th}$ user is defined by the following formula:

$$SK_i = x_i \ PK_i = g^{x_i} \quad x_i \in Z_q^* \tag{2}$$

Similarly, the manager also generates following pair of public and secret keys for the data partition $P$:

$$SK_P = x_P \ PK_P = g^{x_P} \ x_P \in Z_q^* \tag{3}$$

The partition's public key information is stored in a shared directory in the cloud for the access of the mobile users. Alternatively, the partition's public key information can be delivered to the authorized mobile users of the data partition on request. The private keys of the users are delivered to the authorized group member by the manager, but, as it is assumed in [18], the manager should also hold them and use for re-encryption.

*Encryption* To encrypt a message $M \in G_2$, the mobile user $i$ generates a random variable $r \in Z_q^*$. The randomly generated variable together with the partition public key (downloaded from the shared directory in cloud) and the system parameter '$Z$' are used for generation of the ciphertext as expressed in the following equation:

$$C_P = \left(Z^r \cdot M, g^{r x_P}\right) \tag{4}$$

The ciphertext is uploaded on the data partition of the cloud. To decrypt an uploaded ciphertext, there is a need of a data partition secret key that is only known to the manager.

*Re-encryption* The manager is the only entity that has a secret key to decrypt the uploaded encrypted message. When an authorized mobile user wants to download data from the data partition in the cloud, the download request should be made through the manager. The manager ensures mobile user authenticity using the access control list. The access control list contains the information about the authorized users of the data partition in the cloud. The manager serves the authenticated mobile user by generating a re-encryption key $(RK_{P\to j})$ and downloads the corresponding ciphertext from the $P$ data partition (in the cloud). The re-encryption key for the $j^{th}$ user is generated using the system parameter '$g$' along with the $j^{th}$ user and partition secret keys in the following way:

$$RK_{P\to j} = g^{\frac{x_j}{x_P}} \tag{5}$$

The re-encryption key is used to transform the ciphertext encrypted with partition public key into a ciphertext encrypted with the requesting entity public key. During the transformation process, the manager computes the following equation:

$$e\left(g^{rx_P}, g^{\frac{x_j}{x_P}}\right) e\,(g, g)^{rx_j} = Z^{rx_j} \tag{6}$$

$$C_j = \left(Z^r \cdot M, Z^{rx_j}\right) \tag{7}$$

The manager delivers the re-encrypted ciphertext to the requesting mobile user $j$.

*Decryption* The $j^{th}$ mobile user can transform the re-encrypted ciphertext into a plaintext by computing the following equation:

$$\left(\frac{Z^r \cdot M}{(Z^{rx_j})^{\frac{1}{x_j}}}\right) = \left(\frac{Z^r \cdot M}{Z^r}\right) = M \tag{8}$$

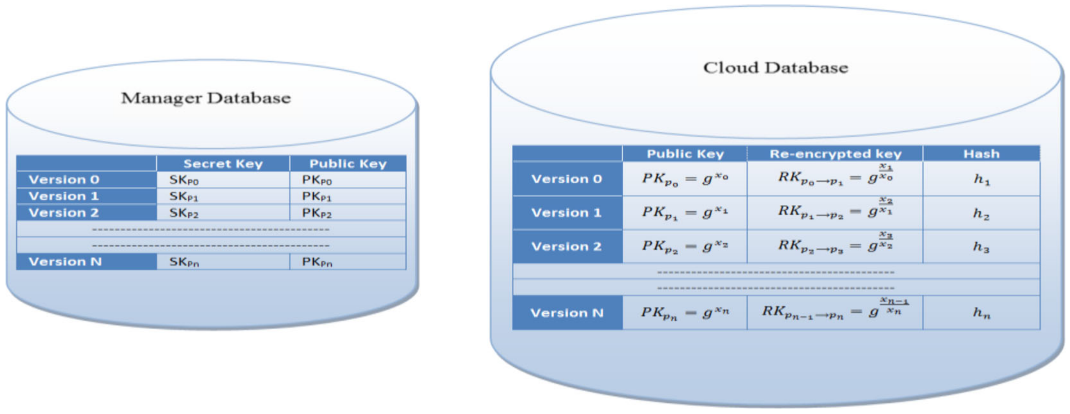
*Analysis* The manager generates keys for the users/data partition, maintains access control list, produces re-encryption keys, and transforms ciphertext into re-encrypted ciphertext. Additionally, the manager has to perform the re-encryption for each data read request, even the request is from the owner of data. The re-encryption process involves bilinear paring that is more expensive than scalar multiplication [30]. Due to the intensive responsibilities of the manager, the growing number of authorized mobile users for the data partition may seriously affect the performance of the system. It can be noted that for the encryption of the message, the user $i$needs only the public data partition key, which is known to all users of data partition $U_P$. For the decryption of this ciphertext, the knowledge of the data partition secret key is needed. Only the manager holds this key. Alternatively, it would be seemed as a fair protection in the case of the fully trustfulness of the manager, but on the other hand the user $i$ is unable to decrypt his own encrypted text (he does not need his private key for the encryption and does not know the data partition secret key), so he must wait for completing of the re-encryption by the manager and request him to send the re-encrypted text.

*Modifications* In order to improve the *MReS* scheme, tysowki and hasan proposed [18] two modifications of this cipher. The first variation is user-managed keys technique, and it is a simple modification of the classical *AFGH* algorithm. On successful authentication, the manager transforms the message encrypted with $A's$ public key into message encrypted with $B's$ public key. The mobile user '$B'$ decrypts the message using personal private key. If the data owner (mobile user $A$) wants to access the uploaded data, there is no need to re-encrypt the message for the data owner. This variation is suitable in an environment where the owner of the message frequently access the personal messages uploaded on the data partition of the cloud. In case of other users, the user-managed keys variation is almost similar to the manager-based re-encryption scheme.

In the second variation called partial ciphertext fetch by user, the mobile users are directly allowed to download $(Z^r.M)$ from the cloud and wait for the other parameter $(g^{rxp})$. The later parameter is delivered to the manager for transformation of the ciphertext encrypted with partition key into a ciphertext encrypted with the requesting user's key. The direct downloading of the former parameter from the cloud storage removes the communication burden from the manager. The manager processes the received parameter to transform '$g^{rxp}$' to '$g^{rxi}$' as discussed above. The generated '$g^{rxi}$' is delivered to the corresponding mobile user. The mobile user utilizes the received information to produce the plaintext from the re-encrypted ciphertext. The undesirable effect of this variation is that the leaving/existing group members can download and access the uploaded encrypted data. In this variant however, the manager hold all keys, just like in the classical *MReS*.

## 3.2 Cloud-Based Proxy Re-Encryption Scheme

Another cryptographic scheme defined by Tysowski and Hasan in [18] is cloud-based re-encryption scheme. The proposed scheme covers the limitations of the existing manager-based re-encryption scheme and the variations of manager-based re-encryption based on user-managed key/partial ciphertext fetch by user. In *CReS*, there are same three groups of system users as discussed in *MReS* with a bit different role of manager. The manager generates and distributes the data partition key to the authenticated mobile users using a secure channel. The mobile users can join or leave the system. In such a case, the manager updates the existing keys, generates new re-encryption key, and distributes re-encryption key information to the



| | Secret Key | Public Key |
|---|---|---|
| Version 0 | $SK_{P0}$ | $PK_{P0}$ |
| Version 1 | $SK_{P1}$ | $PK_{P1}$ |
| Version 2 | $SK_{P2}$ | $PK_{P2}$ |
| ---------------------------------------- | | |
| ---------------------------------------- | | |
| Version N | $SK_{Pn}$ | $PK_{Pn}$ |

| | Public Key | Re-encrypted key | Hash |
|---|---|---|---|
| Version 0 | $PK_{p_0} = g^{x_0}$ | $RK_{p_0 \rightarrow p_1} = g^{\frac{x_1}{x_0}}$ | $h_1$ |
| Version 1 | $PK_{p_1} = g^{x_1}$ | $RK_{p_1 \rightarrow p_2} = g^{\frac{x_2}{x_1}}$ | $h_2$ |
| Version 2 | $PK_{p_2} = g^{x_2}$ | $RK_{p_2 \rightarrow p_3} = g^{\frac{x_3}{x_2}}$ | $h_3$ |
| ---------------------------------------- | | | |
| ---------------------------------------- | | | |
| Version N | $PK_{p_n} = g^{x_n}$ | $RK_{p_{n-1} \rightarrow p_n} = g^{\frac{x_{n-1}}{x_n}}$ | $h_n$ |

**Fig. 4** Databases maintained at cloud and manager

cloud service provider for re-encryption of the existing messages stored on the data partition. The unique feature of the scheme is that the mobile user can directly download/decrypt the data from the data partition without the involvement of the manager. The computationally intensive re-encryption tasks are migrated in the cloud system. A limited activity of the manager makes the cloud-based re-encryption scheme more efficient and scalable. The *CReS* encryption scheme can be defined in the following way:

*Setup* *CReS* operates over two groups $G_1$ and $G_2$ of prime order $q$ with a bilinear map $e : G_1 \times G_1 \rightarrow G_2$. The system parameters are random generators $g \in G_1$ and $Z = e\,(g, g) \in G_2$, the users belong to the virtual organization $U_P$ and have an authorized access to a data partition $P$ in the cloud.

*Key Generation* The manager creates initial version of the following public and private keys for the data partition in the cloud:

$$SK_{p0} = x_0 \;\; PK_{p0} = g^{x_0} \; x_0 \in Z_q^* \tag{9}$$

The data partition's private key information is securely disseminated among all authorized users and the public key information is stored on the shared directory of the cloud.

*Encryption* The mobile user encrypts the message $(M \in G_2)$ using a random variable $r \in Z_q^*$, system parameter 1$Z$', and the partition's public key as follows:

$$C_{p0} = (Z^r.M, g^{rx_0}) \tag{10}$$

Subsequently, the encrypted message is uploaded on the data partition of the cloud.

*Decryption* To decrypt a message, the mobile user downloads the encrypted message from the data partition of the cloud. The authorized users of the data partition know the partition's private key and capable of decryption. To get the original message, the authenticated user computes the following equations.

$$e\left(g^{rx_0}, g^{\frac{1}{x_0}}\right) = e\,(g, g)^r = Z^r \tag{11}$$

$$\left(\frac{Z^r \cdot M}{Z^r}\right) = M \tag{12}$$

*Re-encryption* A single data partition key is shared among all authorized group members; therefore the existing/leaving group members can decrypt the uploaded data using a data partition private key. To keep the data partition secure from the existing/leaving group members, the manager updates the data partition key whenever there is a change in authorized group members. For updating the data partition keys, the manager generates a random salt $h \in \mathbb{Z}$ and applies secure hash function *SHA2* on $h$ and $SKp_{v-1}$ in order to generate a new version of the partition secret key and corresponding public key:

$$SK_{p_v} = x_v = f_{SHA2}(SK_{p_{v-1}}, h_v) \tag{13}$$

$$PK_{p_v} = g^{x_v} \tag{14}$$

This updating of the partition keys implicates the re-encryption of the entire data partition with the updated data partition key. The manager generates

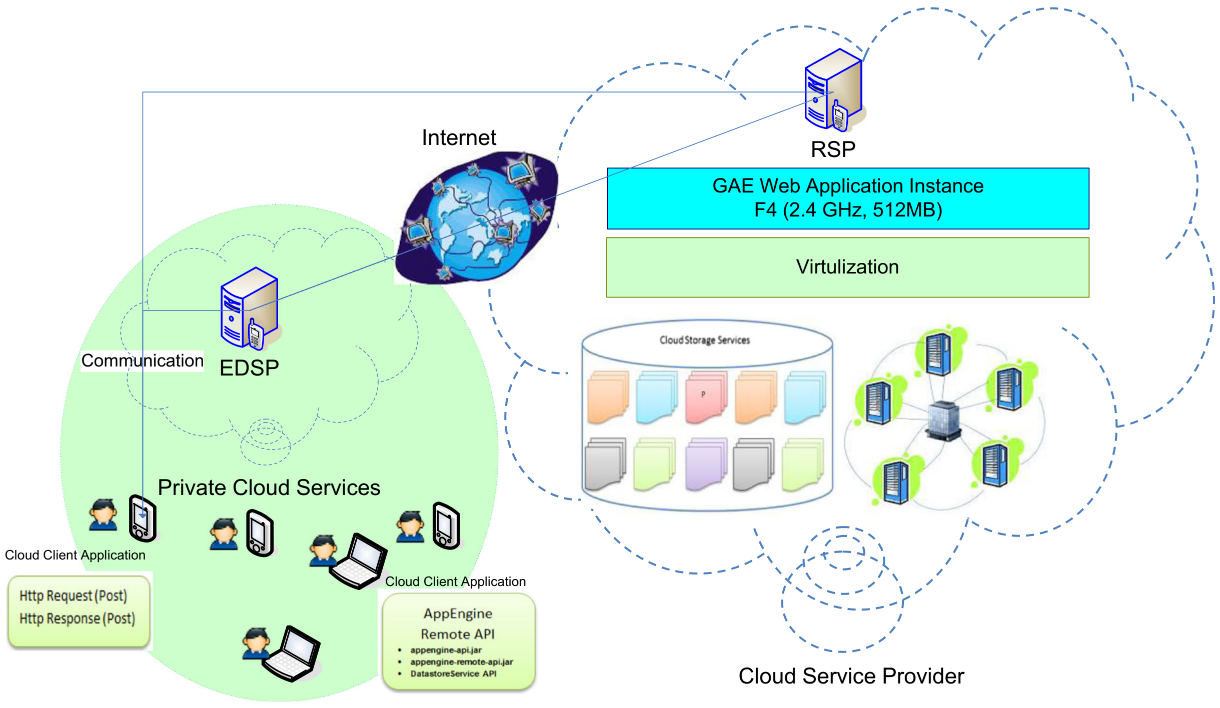


**Fig. 5** System model for the proposed scheme

the re-encryption key and delivers it to the cloud service provider for the transformation of the ciphertext encrypted with the previous key into a ciphertext encrypted with an updated key for particular data partition. The re-encryption key is produced using the following procedures:

$$SK_{p_1} = x_1 = f_{SHA2}(SK_{p_0}, h) \tag{15}$$

$$RK_{P_0 \to P_1} = g^{\frac{SK_{p1}}{SK_{p0}}} = g^{\frac{x_1}{x_0}} \tag{16}$$

Subsequently, the cloud service provider computes the following equation for the transformation of the ciphertext encrypted with $SK_{P0}$ into a ciphertext encrypted with $SK_{P1}$.

$$e\left(g^{rx_0}, g^{\frac{x_1}{x_0}}\right) = e(g, g)^{rx_1} = Z^{rx_1} \tag{17}$$

$$C_{p_1} = (Z^r . M, Z^{rx1}) \tag{18}$$

The cloud service provider keeps the historical information of the key version number, partition public key, re-encryption key, and hash values used to produce re-encryption keys. Figure 4 shows the detail information about the key storage at manager and cloud.

The re-encryption process starts, when user wants to access the uploaded data on the cloud. If user requests for data encrypted with the older version of the secret key, the user may request for re-encryption of the encrypted data with the key in the user possession. If uploaded message is encrypted with the more recent version of the secret key, the requesting user downloads the hash chain history to upgrade the private key for decryption of the message.

The cloud-based re-encryption scheme offloads the computationally intensive tasks on the cloud without disclosing the secret key and user contents. The computational power of the cloud can scale instantly to handle the growing number of user gracefully. The user can directly download the data from the cloud without the involvement of the manager. The manager is only responsible to handle the keys for the data partitions. The minimization of the manager's responsibilities makes the cloud-based re-encryption scheme more scalable as compared to *MReS*. When a user leaves a group, the manager updates the secret key using the aforementioned key modification strategy. The leaving group members can collude with the cloud service provider to retrieve the information stored on the data partition of the cloud. To avoid any possibility of the collusion attack, hash history can be maintained and published by the manager. Alternatively, the manager can periodically reset the hash history on the cloud with a new initial secret key version. Hence, the leaving/existing group members cannot decrypt the uploaded data due to the adoption of the aforementioned strategies.

*Summary* In cloud-based re-encryption scheme, there is an update in data partition keys for the particular data partition whenever a group member leaves the group. However, all the files uploaded on the particular data partition needs to be re-encrypted with a new data partition key. The re-encryption operation involves the massive pairing and exponential operations that is performed on the cloud in cloud-based re-encryption scheme. Although the cloud is capable to handle the re-encryption requests gracefully, the excessive use of the cloud resources overcharged to the organization. Moreover, the cloud-based re-encryption scheme is not suitable in an environment where the group members are continuously changing. The compromising of the legitimate group member may affect the privacy of the whole system in cloud-based re-encryption scheme due to the usage of same partition public key for encrypting the particular data partitions. Furthermore, the execution of computationally intensive encryption and decryption operations consume considerable amount of energy on the mobile device. In the *MReS* and *CReS*, the mobile user requests for public data partition's key from the cloud service provider. The cloud service provider may collude with adversary and transfer adversary generated public key to the legitimate mobile user. The legitimate mobile user encrypts the message using adversary's key and uploads on the data partition of the cloud. Hence, adversary can decrypt all the uploaded encrypted messages using the corresponding private key.

### 3.3 Device-Based Re-Encryption Scheme (DReS)

To see the behavior of pairing-based cryptography on mobile device, we have designed device-based re-encryption scheme in which mobile user is responsible for the execution of encryption, decryption, and re-encryption operations. We have designed this scheme to identify the improvement of the proposed scheme in terms of energy consumption on the mobile device while offloading the encryption and decryption operations on the trusted entities. The main limitation of the device-based re-encryption scheme is that the data owner must be available all the time for providing the access privileges to other users which is infeasible due to the mobility of the mobile users.

*Contribution* Our proposed *CMReS* is based on *MReS* and *CReS* technique and provides the following additional features:

- The mobile user can offload encryption, decryption, and re-encryption operations on trusted entity and cloud that reduce the battery consumption on the mobile device.
- Each group member encrypts the message with the personal private key; therefore compromising of the legitimate group member does not affect the privacy of the whole system.
- The data partition is secured from the existing/leaving group members without putting any processing overhead on the cloud unlike *CReS*.
- In *CMReS*, each user is responsible to generate personal key pair and disseminate the personal public key information unlike *MReS* and *CReS*.
- In the proposed scheme, the owner of the data is able to directly download and decrypt the message without the involvement of trusted entity unlike *MReS*.
- Moreover, the re-encryption services are migrated on cloud without disclosing any private keys unlike *MReS* and *CReS*.

## 4 The Proposed Cloud-Manager-Based Encryption Scheme (CMReS)

By combining the characteristics of the manager-based re-encryption and cloud-based re-encryption schemes, we proposed a cloud-manager-based re-encryption scheme for offloading the complex computational operations on the trusted-entity and cloud. Moreover, from the experimental results presented in next sections, this can be concluded that the energy consumption during encryption and decryption is directly proportional to the size of the file. Increase in file size also increases the total number of encryption and decryption operations with constant re-encryption operations. Therefore, there is a need of security scheme that can offload the encryption and decryption operations on the cloud/third-party in a trusted mode. In the proposed *CMReS*, the encryption, decryption, and re-encryption tasks are distributed between the trusted entity and cloud.

There are four main modules in this system, namely **(a)** cloud client application hosted on the mobile users, **(b)** Encryption/Decryption Service Provider (*EDSP*) module hosted on private cloud inside the client organization, **(c)** Re-encryption Service Provider (RSP) module hosted on the public cloud, and **(d)** cloud storage services available on public cloud as illustrated in Fig. 5.

The cloud service provider offers computational and storage services to the mobile users. The mobile users upload/download the data to/from the data partition of the cloud through the cloud client application. The *EDSP* is a ***fully*** trusted entity under the control of a client organization whose prime responsibility is to provide encryption and decryption services to the authorized mobile users. The *RSP* module is hosted on public cloud which is responsible for maintaining the re-encryption keys and providing the re-encryption services for each authorized mobile user. The *RSP* module only holds the re-encryption keys of the cloud users belonging to the same virtual organization for providing re-encryption services. The unique feature of the scheme is that the *RSP* is hosted on the cloud and provides re-encryption services without knowing the private keys of the mobile users.

The *CMReS* can be defined in the following way:

*Setup* *CMReS* operates over two groups $G_1$ and $G_2$ of prime order $q$ with a bilinear map $e : G_1 \times G_1 \rightarrow G_2$. The system parameters are random generators $g \in G_1$ and $Z = e(g, g) \in G_2$, the users belong to the virtual organization $U_P$ and have an authorized access to a data partition '$P$' in the cloud.

*Key Generation* In *CMReS*, the authorized users (belonging to the data partition '$P$') and *EDSP* generate the public and private keys using the following procedure:

$$SK_i = x_i \quad PK_i = g^{x_i} \; x_i \in Z_q^* \tag{19}$$

$$SK_{EDSP} = x_{EDSP} \quad PK_{EDSP} = g^{x_{EDSP}} \; x_{EDSP} \in Z_q^* \tag{20}$$

Each authorized group member communicates with the *EDSP* to get the authorized user's list of the particular data partition '$P$' for generation of the re-encryption keys. The authorized user's list of the data partition is also forwarded to the *RSP* module for re-encryption operation. The mobile user '$A$' generates the re-encryption keys for authorized user's list '$U$' using users' public key and personal private key as shown below:

$$RK_{i->p} = \left(g^{x_i}\right)^{\frac{1}{x_A}}, \forall u_i \in U \tag{21}$$

The *RSP* holds re-encryption keys for each authorized user of the data partition '$P$' for providing the re-encryption services on behalf of the mobile users. After completion of key generation phase, authorized users of the data partition '$P$' have the personal private key, *RSP* holds the re-encryption keys of each authorized user, and *EDSP* has the personal private key for providing encryption and decryption services to the mobile users.

*Encryption* Let a mobile user '$A$' wants to upload a message ($M \in G_2$) on the data partition '$P$' of the cloud. The mobile user '$A$' requests for encryption services to *EDSP*. In return, *EDSP* generates a large prime number '$p_{sk1}$', encrypts the generated prime with the public key of the mobile user '$A$', and delivers to the requesting entity. The generated prime number is used as a session key between the mobile user and *EDSP* to keep the communication secure inside the client organization. Subsequently, the mobile user transforms the Message '$M$' into the first level encrypted message '$C_{flevel}$' by multiplying the message with session key as shown below:

$$C_{flevel} = (M.p_{sk_1}) \tag{22}$$

Afterwards, the mobile user '$A$' delivers $C_{flevel}$ to *EDSP*. The *EDSP* generates the$r1 \in Z_q^*$ randomly and encrypts the message using the following procedure:

$$Z_{new}^{r_1} = \frac{Z^{r_1}}{p_{sk_1}} \tag{23}$$

$$(C_{flevel} \,.\, Z_{new}^{r_1}) = (Z^{r_1}.\, M) \tag{24}$$

$$C = Z^{r_1}.\, M,\; C_A = g^{r_1 x_A} \tag{25}$$

The *EDSP* uploads the encrypted message '$C$' and '$C_A$' on the data partition of the cloud on behalf of the mobile user '$A$'. At the end of the encryption phase, the cloud storage contains the encrypted message in the form shown in (25).

*Re-encryption* Let a mobile user '$B$' wants to download and decrypt the uploaded message of the mobile user '$A$'. The mobile user '$B$' requests to the *RSP* for the uploaded message '$M$'. The *RSP* checks the access control list for the requested message. If a mobile user '$B$' has access rights to the uploaded message, the *RSP* gets the encrypted message '$C$' and '$C_A$' from the data partition,and transforms '$C_A$' into '$C_{EDSP}$' using the re-encryption key $RK_{A \longrightarrow EDSP}$ and '$C_A$' as shown in the following equation:

$$\begin{aligned} C_{EDSP} &= e\left(RK_{A\rightarrow EDSP},\ C_A\right) \\ &= e\left(g^{\frac{x_{EDSP}}{x_A}}, g^{x_A r_1}\right) \\ &= e(g,\ g)^{x_{EDSP}\, r_1} = Z^{x_{EDSP}\, r_1} \end{aligned} \tag{26}$$

Afterwards, the *RSP* requests *EDSP* for decryption services by providing '$C$' and '$C_{EDSP}$' on behalf of the mobile user '$B$'.

*Decryption* The *EDSP* generates a large prime number '$p_{sk2}$', encrypts the number with the mobile user '$B$' public key, and delivers the encrypted prime number to the mobile user '$B$'. Subsequently, the *EDSP* transforms the received encrypted message into first level encrypted message using the following procedure:

$$\left(Z^{r_1 x_{EDSP}}\right)^{\frac{1}{x_{EDSP}}} = Z^{r_1} \tag{27}$$

$$\left(\frac{\left(\frac{Z^{r_1}.M}{Z^{r_1}}\right)}{p_{sk_2}}\right) = C_{flevel} \tag{28}$$

The *EDSP* handovers the first level encrypted message '$C_{flevel}$' to the mobile user '$B$'. The mobile user '$B$' transforms '$C_{flevel}$' into $M$ using the following procedures:

$$M = C_{flevel} \cdot p_{sk_2} \tag{29}$$

Due to the involvement of the session key, the communication between the mobile device and *EDSP* is secured inside the client organization. Although mobile user has to encrypt and decrypt the whole message, the transformation of the '$M'$ message into first level ciphertext with comparatively light-weight operation improves the overall turnaround time and energy consumption on the mobile device as compared to the existing re-encryption schemes. Furthermore, the *RSP* provides the re-encryption services without knowing the private key of the mobile users. The know attributes to the *RSP* are re-encryption keys. Therefore, the *RSP* needs to know the private keys in order to get the information from the encrypted messages. If adversary downloads the encrypted messages from the cloud storage, the adversary needs to know the private key of the data owner. The proposed scheme is secured from the existing/leaving group members due to the fact that the uploaded messages are encrypted with the data owner public key and can be only decrypted with the private key of the data owner. Furthermore, each group member encrypts the message with the personal private key; therefore compromising of the legitimate group member does not affect the privacy of the whole system In *MReS*, the messages are encrypted /decrypted using the limited computational power of the mobile devices and the manager is responsible for re-encryption of the message. In *CReS*, the messages are also encrypted/decrypted using the limited computational power of the mobile devices and re-encryption operations are migrated on the cloud in a trusted mode. While providing re-encryption services, *MReS* poses burden on trusted entity which may affect the scalability. Alternatively, *CReS* poses burden on cloud having enormous computation power while providing re-encryption services. However, the entire data partition must be re-encrypted in *CReS* for keeping the data partition secure from the existing/leaving group members. In the proposed scheme, the *RSP* is hosted on cloud and provides the re-encryption services without revealing the data contents and private keys unlike *MReS*. Moreover, there is no need

**Table 1** Components used for encryption, decryption, and re-encryption

| | | Encryption | Decryption | Re-encryption |
|---|---|---|---|---|
| **DReS** | Mobile device | ✓ | ✓ | ✓ |
| | Cloud | | | |
| | Trusted-entity | | | |
| **MReS** | Mobile device | ✓ | ✓ | |
| | Cloud | | | |
| | Trusted-entity | | | ✓ |
| **CReS** | Mobile device | ✓ | ✓ | |
| | Cloud | | | ✓ |
| | Trusted-entity | | | |
| **CMReS** | Mobile device | ✓ | ✓ | |
| | Cloud | | | ✓ |
| | Trusted-entity | ✓ | ✓ | |

**Table 2** Hardware specifications

| Sony Xperia S | | Trusted entitles | |
|---|---|---|---|
| CPU | Dual-core 1.5 GHz | CPU | Inter(R) Core ™i5-2400 CPU @ 3.10 GHz 3.10 GHz |
| RAM | 1 GB | RAM | 4 GB |
| Storage | 32 GB | Storage | 200 GB |
| OS | Android OS v4.0.4 | OS | Window 7.0 professional 64 bits |
| Mobile application development toolkit | Android SDK | Application development toolkit | JDK 1.6 |
| Battery | 1750 mAh | – | – |
| Internet connectivity | Wi-Fi | – | – |

to re-encrypt the entire data partition for keeping the data partition secure from the existing/leaving group members. The *RSP* module of*CMReS* poses burden on cloud having enormous computation power while providing re-encryption services. The *EDSP* module of*CMReS* poses burden on private cloud for providing encryption and decryption services. Table 1 shows the components involve in processing of encryption, decryption, and re-encryption operations. In *CMReS*, trusted-entity is comparatively overburdened as compared to *CReS* due to additional responsibilities of the encryption and decryption operations on behalf of the mobile users. However, the manager has to perform the re-encryption services in *MReS*. Similarly, *CMReS* uses cloud services for providing re-encryption services with comparatively less process burden on cloud as compared to *CReS*. *MReS* and *CReS* keep mobile device overburden as compared to *CMReS* while performing encryption and decryption operations. In *CMReS*, the mobile device transforms the message into first level ciphertext with comparatively lightweight operation as compared to *CReS* and *MReS* and leave rest of the encryption/decryption task for *EDSP*.

**Table 3** Dataset to evaluate the performance of the security schemes

| No. | File size | Files | Operations |
|---|---|---|---|
| Dataset I | | | |
| 1) | 1024 Bytes | 200 | 3200 |
| 2) | 1024 Bytes | 300 | 4800 |
| 3) | 1024 Bytes | 400 | 6400 |
| 4) | 1024 Bytes | 500 | 8000 |
| 5) | 1024 Bytes | 600 | 9600 |
| Dataset II | | | |
| 1) | 1024 Bytes | 200 | 3200 |
| 2) | 5120 Bytes | 200 | 16000 |
| 3) | 10240 Bytes | 200 | 32000 |
| 4) | 15360 Bytes | 200 | 48000 |
| 5) | 20480 Bytes | 200 | 64000 |
| Dataset III | | | |
| 1) | 1024 Bytes | 200,300,400,500,600 | 3200, 4800, 6400, 8000, 9600 |
| 3) | 10240 Bytes | 200,300,400,500,600 | 32000, 48000, 64000, 80000, 96000 |
| 5) | 20480 Bytes | 200,300,400,500,600 | 64000, 96000, 128000, 160000, 192000 |

## 5 Empirical Analysis

In this section, we provide a simple evaluation analysis of the proposed methodologies along with *DReS*, *MReS*, and *CReS*. The schemes are evaluated on the basis of turnaround time, *CPU* utilization, energy, and memory consumption on mobile device while performing encryption, decryption, and re-encryption operations. To evaluate the resource utilization, we have developed a mobile application that executes the android top command [36] to get the resources utilization information of running processes for every second. The energy consumption is evaluated before the start and after the completion of security operation using the following formula [37]:

$$Battery\,Consumpton(BC) = \left(\frac{l}{s}\right) \times 100 \qquad (30)$$

where maximum battery level is represented with '$s$' and current battery level is represented with '$l$'. Hence, total battery consumption in percentage can be easily calculated by subtracting the two values. Similarly, we have used the *System.currentTimeMillis()* [38] API to get the current time in milliseconds before the start and after the completion of security operation. The total turnaround time of a security operation is calculated by subtracting the two values.

The hardware characteristics of the mobile devices and physical layer of the cloud system are presented in Table 2. We have used Sony Xperia S smartphones with Android operating system. A single frontend instance of class *F4* having 512 MB of memory and 2.4 GHz of *CPU* capacity is hosted on the Google App Engine (*GAE*) [39]. The cloud client application communicates with the hosted instance through App Engine Remote *API* [40, 41] or http request/response methods (*GET* and *POST*). The cloud client application sends the request for the cloud services to a hosted instance on the *GAE*. The hosted instance receives the client requests and provides appropriate services to the requesting application. The trusted entities involved in the schemes are implemented in *JDK 1.6* and deployed on single/multiple system(s) depending on the computational requirement of an organization. Pairing-based cryptographic operations are implemented by using the java Pairing Based Cryptography library (*jPBL*) [42, 43]. We assume a Type A pairing construed on curve $y^2 = x^3 + x$ on a prime field $F_q$ for a prime number $q$ of *512* bits. For secure cryptosystem, the value of $q$ should be large to keep maximum points on the elliptic curve. According to Standards for Efficient Cryptography [44], the recommend value of $q$ is 112-521 bits.

The main security operations involved in the selected schemes are encryption, decryption, and re-encryption. To study the impact of each scheme on the mobile device while performing the aforementioned operations, we used three datasets for experiment with the characteristics presented in the Table 3. The dataset I is used to study the impact of increase in number of files with constant file size while performing security operations in each scheme. In the dataset II, the behaviors of the aforementioned security schemes are studied with constant number of files and variable file size while performing the above-mentioned security operations. To investigate the impact of the selected security schemes with variable number of files and file size, the dataset III is used.

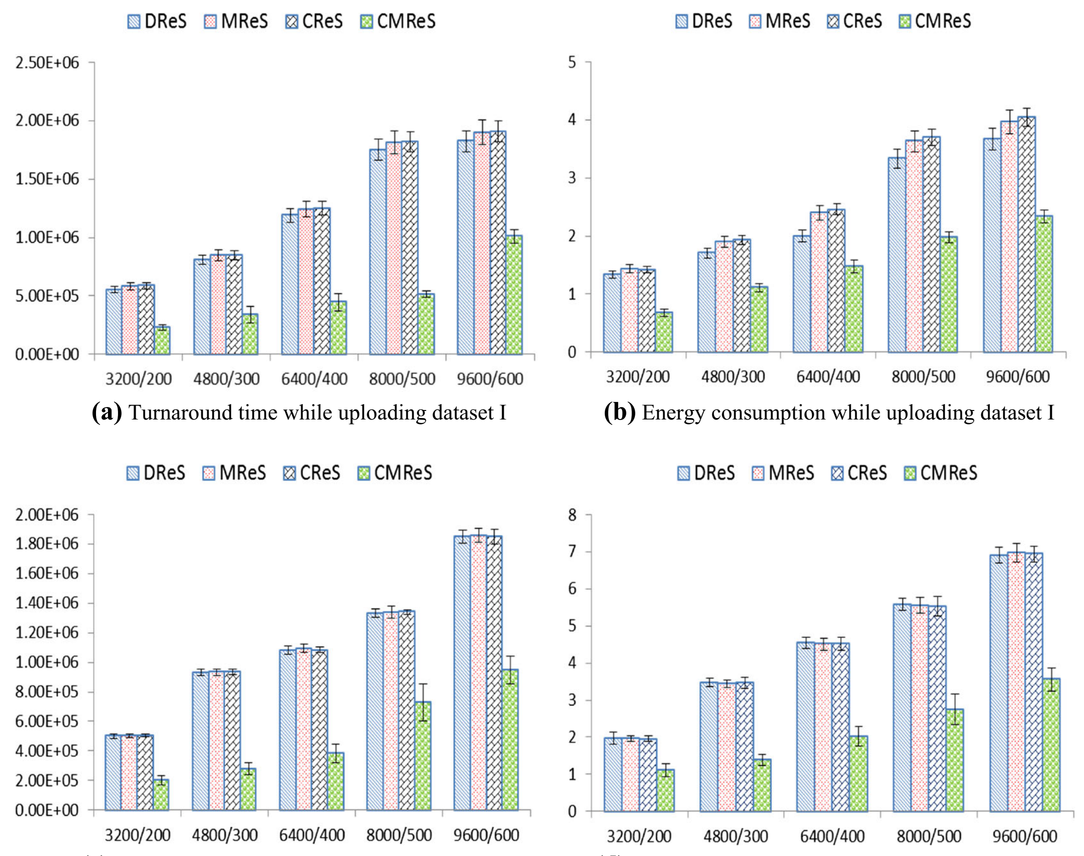
In each selected scheme, the maximum 64 bytes data block can be encrypted at any given time due to the size of prime field $F_q$. Therefore, total number of multiplication operations ($M.Z^r$) during encryption and division operation ($(M.Z^r)/Z^r$) during decryption can be calculated and given in Table 3. Each experiment is repeated ten times under the same system configuration. The obtained result of each experiment is used to evaluate the confidence interval using the following equation:

$$\bar{X} \pm t \times \frac{\sigma}{\sqrt{n}} \qquad (31)$$

where X$^-$ is the mean value of result, '$n$' represents sample size, '$\sigma$' is the representation of standard deviation, '$t$' is the representation of confidence coefficient having 1.96 value for 95 % confidence interval. The average results of experiments along with the confidence intervals are presented in the graphs. The confidence intervals show that there is 95 % probability that the value of experiment lies within the range.

### 5.1 Performance Evaluation on the Mobile Devices

In this section, we analyzed the turnaround time, energy consumption, and resource utilization on the mobile device while performing uploading, downloading, encryption, decryption, and re-encryption operations on dataset I, II, and III for *DReS*, *MReS*, *CReS*,



**(a)** Turnaround time while uploading dataset I

**(b)** Energy consumption while uploading dataset I

**(c)** Turnaround time while downloading dataset I

**(d)** Energy consumption while downloading dataset I

**Fig. 6** Turnaround time and energy consumption for dataset I

and *CMReS*. The turnaround time and energy consumption are shown in the y-axis of the following graphs which are measured in milliseconds and percentage, respectively. The x-axis of the graphs show the total files along with the total operations required for encryption/decryption

### *5.1.1 Uploading and Downloading of Dataset I and II*

The total time needed for completing the uploading/downloading requests is calculated as a direct sum of the files reading time '$t_{FR}$', encryption/decryption time '$t_{ED}$', uploading/downloading time '$t_{UD}$', and time consumed in communication with trusted entity/cloud '$t_{TE}$' (if applicable) as it is presented in the following equation:

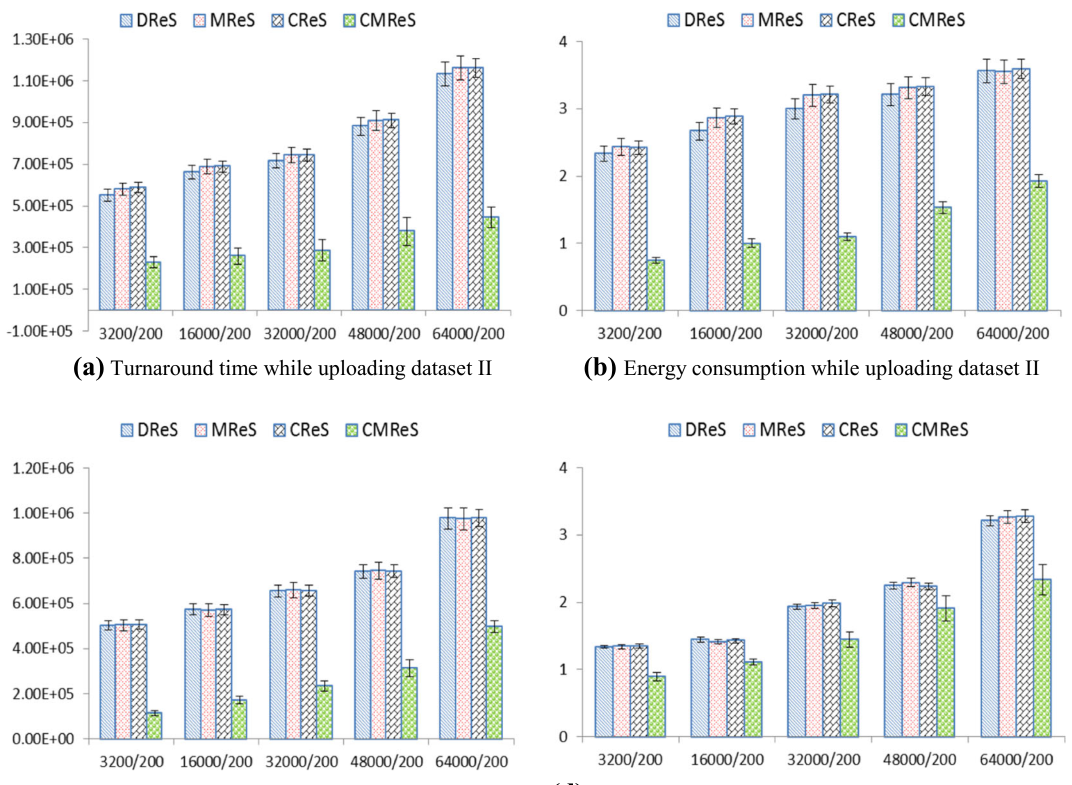
$$Total\ Turnaround\ Time\ (T_{TT}) = t_{FR} + t_{ED} + t_{UD} + t_{TE} \tag{32}$$

Similarly, the total energy consumption for completing the uploading and downloading requests is evaluated using the following equation:

$$Total\ Energy\ Consumed\ (T_{EC}) = E_{Comm}+E_{FR}+E_{ED}+E_{TE} \tag{33}$$

where '$E_{Comm}$' represents the energy consumption on the mobile device for completing the file uploading and downloading from the cloud storage, '$E_{FR}$' stands for the energy consumption in files reading from the mobile device's local storage, '$E_{TE}$' is the representation of energy consumed in communication with trusted entity/cloud(if applicable), and '$E_{ED}$' is the representation of the energy consumption for performing the pairing, exponential, and multiplication operations during encryption/decryption.

*Experiment I* In this experiment, we compared the turnaround time and energy consumption for the



**(a)** Turnaround time while uploading dataset II

**(b)** Energy consumption while uploading dataset II

**(c)** Turnaround time while uploading dataset II

**(d)** Energy consumption while uploading dataset II

**Fig. 7** Turnaround time and energy consumption for dataset II

uploading/downloading the dataset I on the cloud storage along with the confidence interval. The results are presented in Fig. 6.

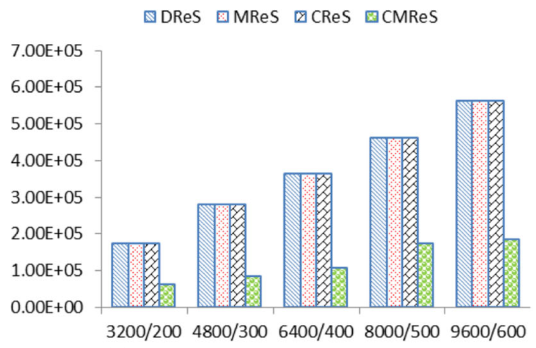
In *DReS*, *MReS*, and *CReS*, the entire encryption and decryption operations are executed on the mobile device. Therefore, the time and energy consumption by performing the pairing, exponential, and multiplication operations during encryption and decryption process are almost similar. For offloading the re-encryption operations on the trusted entities/cloud, the *MReS* and *CReS* deliver the file access control list to the trusted entities that may take slightly more time in uploading as compared to the *DReS*. However, the downloading process takes same time for *DReS*, *MReS*, and *CReS*. In the proposed *CMReS*, the mobile user transforms the message into the first level ciphertext with comparatively light-weight operation. Although mobile user has to encrypt and decrypt the whole message, the transformation of the message into first level ciphertext with comparatively light-weight operation improves the overall turnaround time and energy consumption on the mobile device as compared to the existing schemes. The *64* bits data blocks of the original message are mapped into $G_2$ element of size *512* bits and multiplied with *90* bits long prime number to generate a first level encrypted code. To increase the security level of first level encrypted message, the size on the session key need to be increased. However, increase in key size also increases the resource utilization on the mobile device. One solution to the aforementioned issue is to deicide the session key size by considering the sensitivity of the data.

*Experiment II* This experiment investigates the impact on turnaround time and energy consumption while uploading/downloading the dataset II on the cloud storage for each selected scheme. The experimental results are presented in Fig. 7.

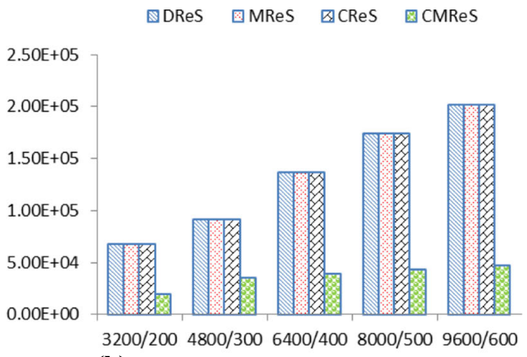
In the dataset II, there is an increase in the file size with constant number of files. The increase in the file size increases the total number of exponential and multiplication operation during encryption/decryption that takes more time and consumes more energy to complete the uploading/downloading request. Therefore, varying the total number of files or varying the size of file during uploading and downloading affect the turnaround time and energy consumption on the mobile device. From the experiment I and II, the proposed *CMReS* complete the uploading/downloading of dataset I and II on the cloud storage with minimum turnaround time and energy consumption on mobile device as compared to existing re-encryption schemes.

### 5.1.2 Encryption/Decryption

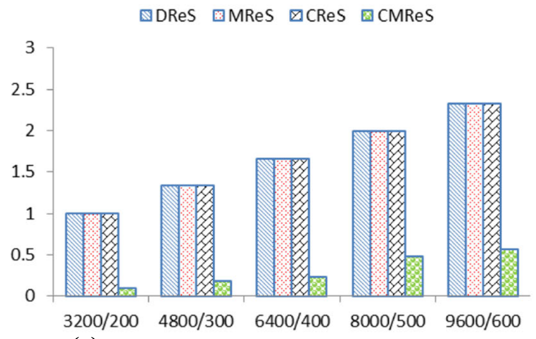
To see the performance relationship between *CMReS*



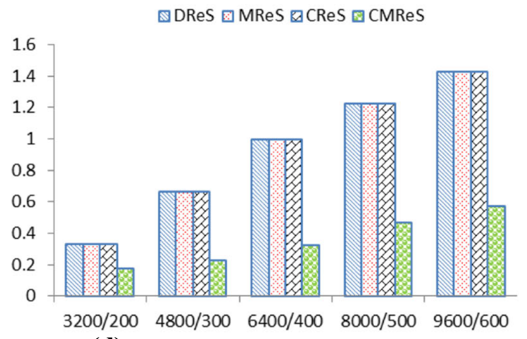
**(a)** Turnaround time while encrypting files each having 1024 bytes



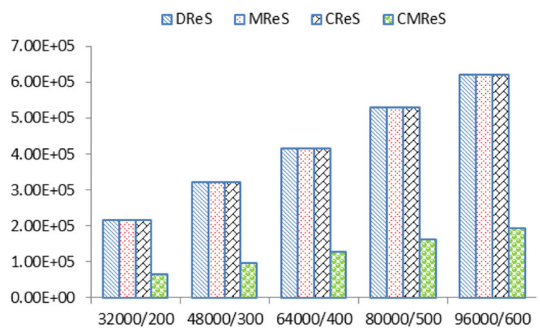
**(b)** Energy Consumption while encrypting files each having 1024 bytes



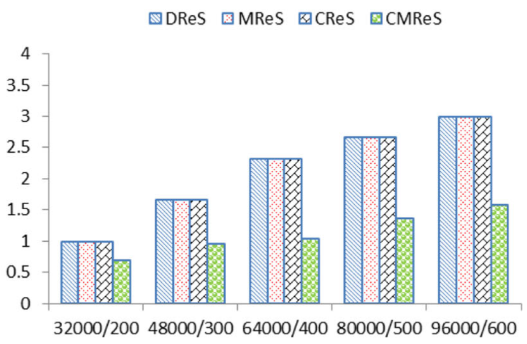
**(c)** Turnaround time while decrypting files each having 1024 bytes



**(d)** Energy Consumption while decrypting files each having 1024 bytes

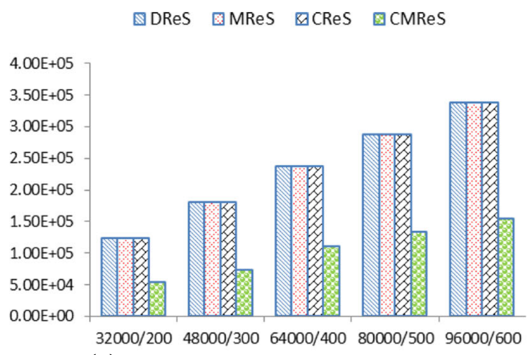


**(e)** Turnaround time while encrypting files each having 10240 bytes

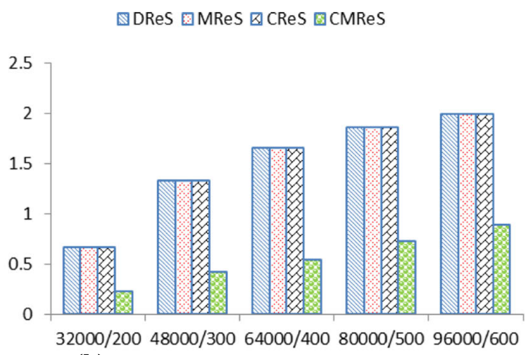


**(f)** Energy Consumption while encrypting files each having 10240 bytes

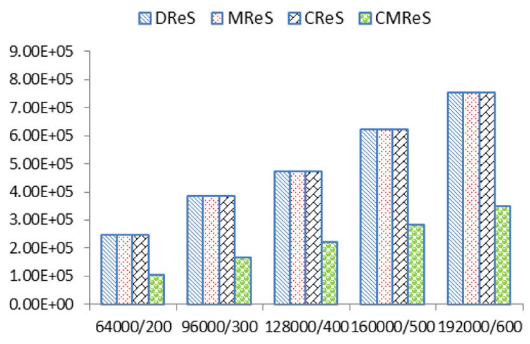
**Fig. 8** Turnaround time and energy consumption for dataset III



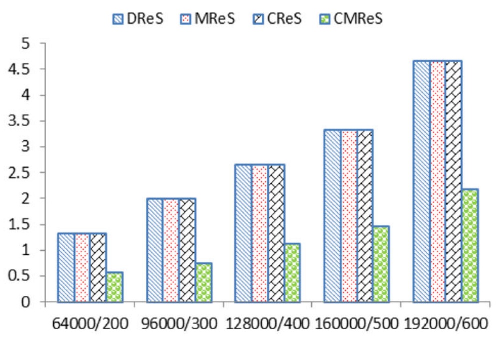
**(g)** Turnaround time while decrypting files each having 10240 bytes



**(h)** Energy Consumption while decrypting files each having 10240 bytes



**(i)** Turnaround time while encrypting files each having 20480 bytes



**(j)** Energy consumption while encrypting files each having 20480 bytes

**Fig. 8** (continued)

and the existing schemes, we have re-evaluated the turnaround time and energy consumption while performing only the core encryption/decryption operations (excluding the communication cost) for each scheme on the mobile device using dataset III. The experimental results are presented in Fig. 8.

This can be concluded that the proposed*CMReS* completes the encryption/decryption requests comparatively less time with less energy consumption on the mobile device as compared to the existing re-encryption schemes. The results presented in the graph also confirm that the increase in the number of files/file size also increases the request compilation time and energy consumption on the mobile device.

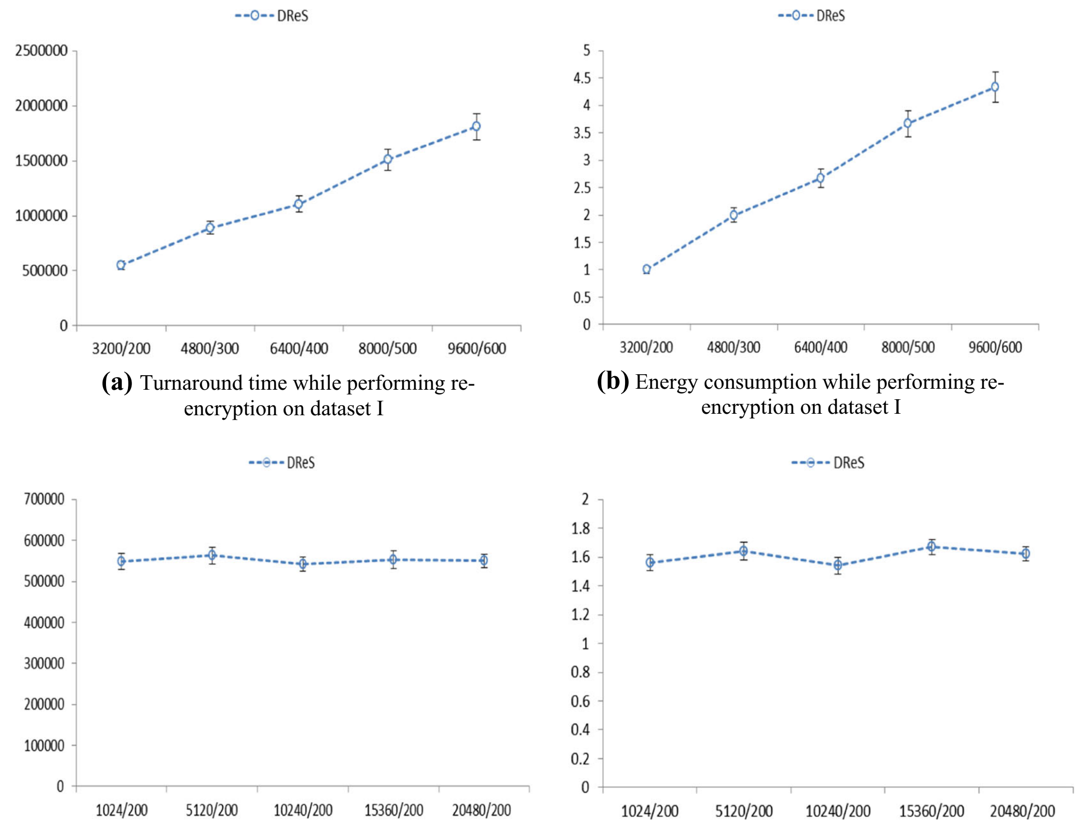
### *5.1.3 Re-Encryption*

*MReS*, *CReS*, and *CMReS* offload the re-encryption operations on trusted entity/cloud; however *DReS* performs the re-encryption operations on mobile device. Therefore, in this experiment we examined the turnaround time and energy consumption on the mobile device while performing the re-encryption operations on dataset I and II in *DReS*. The experimental results are shown in Fig. 9.

It can be observed from the results presented in Figs. 9a and b that the increase in the number of files increases the turnaround time and energy consumption for completing the re-encryption operations on the mobile device. The increase in turnaround time and energy consumption is due to the increase in number of re-encryption operations while increasing the number of files. However, Figs. 9c and d show the increase in the file size does not effect on number of re-encryption operations which keeps the turnaround time and energy consumption almost same on the mobile device. This experiment concludes that the total re-encryption operations are directly proportional to the total number of files regardless of file size.

### *5.1.4 Resource Utilization*

In this section, we investigated the resources utilization on the mobile device while performing



**(a)** Turnaround time while performing re-encryption on dataset I

**(b)** Energy consumption while performing re-encryption on dataset I

**(c)** Turnaround time while performing re-encryption on dataset II

**(d)** Energy consumption while performing re-encryption on dataset II

**Fig. 9** Turnaround time and energy consumption while performing re-encryption on dataset I and dataset II

the encryption/decryption/re-encryption operations on *200* hundred files of size *20480* bytes. We are more interested in *CPU* utilization, private memory, and proportional Set Size (*PSS*) memory of the mobile device. The private memory is the amount of memory that is released when the process is terminated. The *PSS* is shared memory among the processes and evaluated in *KB* using the following equation [45].

$$PSS = \left(\frac{S_{Mem}}{N}\right) \tag{34}$$

where '$S_{Mem}$' represents the size of shared memory among the processes, and '$N$' denotes the total number of process using the shared memory. To evaluate the resource utilization, we have developed a mobile application that executes the android top command [36] to get the resources utilization information of running processes for every second. The average results are presented in the Tables 4, 5, and 6 while performing encryption, decryption, and re-encryption operations, respectively.

Tables 4, 5, and 6 show that the proposed re-encryption scheme consumes few resources on the mobile device while performing security operations as compared to the existing re-encryption schemes. However, there is a minor difference in

**Table 4** Resources utilization while encrypting 200 files of size 20480 bytes

| | Total operation | DReS | MReS | CReS | CMReS |
|---|---|---|---|---|---|
| Private memory (KB) | 64000 | 15939.69 | 15900.21 | 15221.22 | 15652.21 |
| Shared memory (KB) | 64000 | 16902.56 | 16221.66 | 16011.95 | 16221.32 |
| CPU (%) | 64000 | 56.33 | 57.66 | 55.95 | 56.21 |

**Table 5** Resources utilization while decrypting 200 files of size 20480 bytes

| | Total operation | DReS | MReS | CReS | CMReS |
|---|---|---|---|---|---|
| Private memory (KB) | 64000 | 15498.95 | 15900.65 | 15745.66 | 15399.66 |
| Shared memory (KB) | 64000 | 17746.05 | 16900.25 | 17502.95 | 16795.22 |
| CPU (%) | 64000 | 58.66 | 57.95 | 57.78 | 57.11 |

the resources utilization while performing the encryption/decryption operations on the mobile device. The important parameter is a time to which the mobile device resources are kept busy that can be investigated from the Section 5.1.2. The Table 6 presents the resource consumption on the mobile device while performing the re-encryption operation for the *DReS*. There is no data for the other schemes due to the offloading of re-encryption tasks on the trusted entities/cloud. The aforementioned tables and graphs presented in the Section 5.1.2 prove that the proposed *CMReS* consume less energy on the mobile device as compared to the existing schemes.

## 5.2 Formal Analysis and Verification

We have used High Level Petri Nets (*HLPN*), Satis?ability Modulo Theories Library (*SMT-Lib*), and a *Z3* solver to verify the working of our proposed scheme. *HLPN* is used to represent the system graphically and mathematically. It defines mathematical rules for the working of the system under consideration. *SMT-Lib* validates the mathematical rules defined over the system. *Z3* solver is used along with the *SMT-Lib* to prove the satisfiability of rules. Interested readers can read [46] to get details about use of *HLPN*, *SMT*, and *Z3* solver to validate working and security properties of a system.

The *HLPN* representation of the *EMReS* is depicted in Fig. 10. For development of petri net model, we have identified data types, places, and mappings of data types to places. Tables 7 and 8 show the data types and mappings, respectively. In *HLPN* model, the rectangular black boxes are the representation of transitions and circles are place.

The working of *EMReS* is discussed in previous sections. In this section, we define formulas to map on the transitions. Key generation is done at *EDSP* according to the procedure define in (20). The following rule maps to the *Key_Gen* in *HLPN* model.

$$\begin{aligned}&R(Key\_Gen) = \forall x_1 \in X_1 |\\ &x_1[4] := Gen\_g() \wedge x_1[2] := Gen\_SK_i(x_1[1]) \wedge x_1[3]\\ &:= Gen\_PK_i(x_1[1], x_1[4], x_1[2]) \wedge\\ &X_1' = X_1 \cup \{x_1[4], x_1[2], x_1[3]\}\end{aligned} \tag{35}$$

The authorized users' list is sent to all of the users belonging to same data partition by *EDSP*. The following formula is defined on transition *S_UL*.

$$\begin{aligned}&R(S\_UL) = \forall x_2 \in X_2, \forall x_3 \in X_3 |\\ &x_3[1] := x_2[1] \wedge\\ &X_3' = X_3 \cup \{x_3[1]\}\end{aligned} \tag{36}$$

Each user generates personal public private key pair according to the procedure defined in (19). The following rule map to the transition *Gen_UK*.

$$\begin{aligned}&R(Gen\_UK) = \forall x_{1a} \in X_{1a} |\\ &x_{1a}[2] := Gen\_SK_i(x_{1a}[1]) \wedge x_{1a}[3] := Gen\_PK_1(x_{1a}[1]) \wedge\\ &X_{1a}' = X_{1a} \cup \{x_{1a}[2], x_{1a}[3]\}\end{aligned} \tag{37}$$

The user list is also forwarded to *RSP* according to the following rule.

$$\begin{aligned}&R(UL) = \forall x_4 \in X_4, \forall x_5 \in X_5 |\\ &x_5[1] := x_4[1] \wedge\\ &X_5' = X_5 \cup \{x_5[1]\}\end{aligned} \tag{38}$$

**Table 6** Resources utilization while performing 200 re-encryption operations

| | Total operation | DReS | MReS | CReS | CMReS |
|---|---|---|---|---|---|
| Private memory (KB) | 64000 | 15862.46 | – | – | – |
| Shared memory (KB) | 64000 | 17999.87 | – | – | – |
| CPU (%) | 64000 | 55.33 | – | – | – |

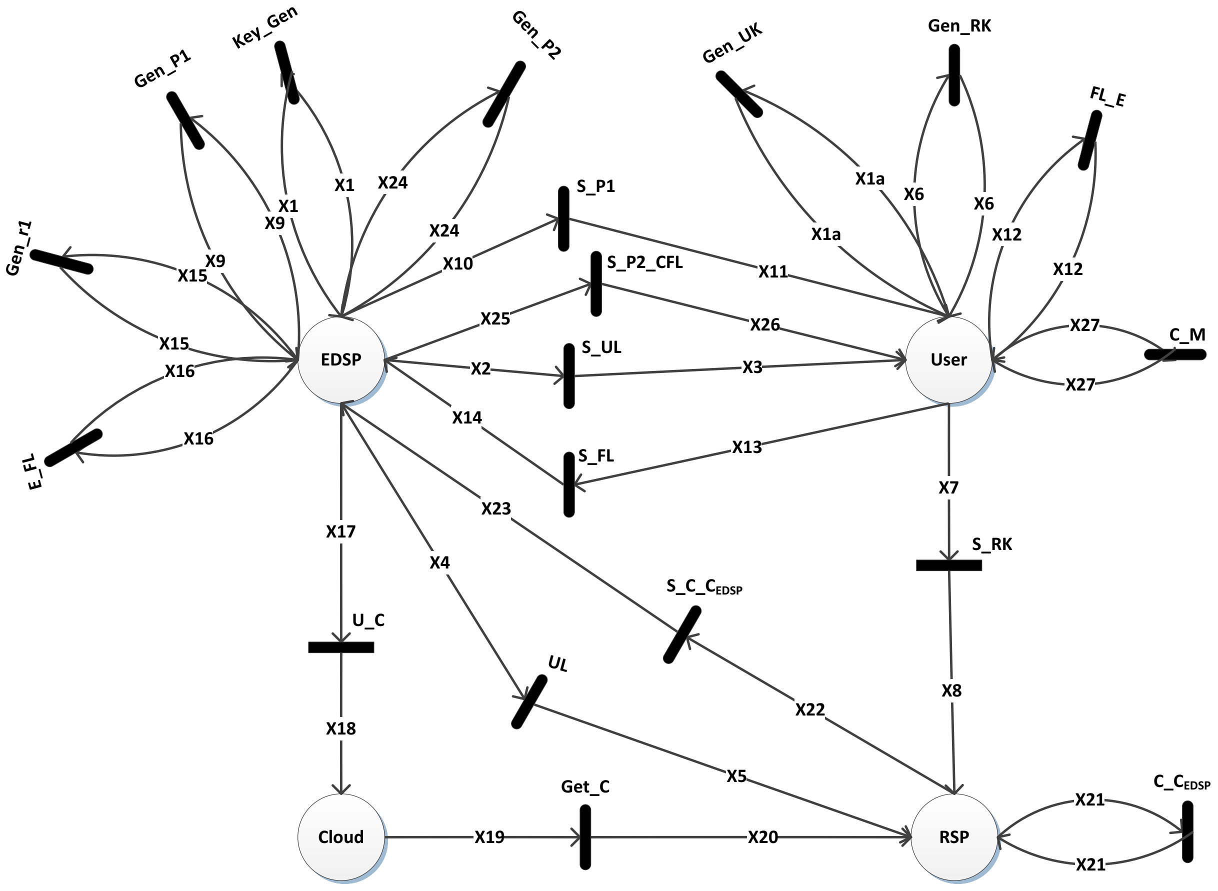


**Fig. 10** HLPN model for EMReS

Each user generates re-encryption keys for all other users in the list. Following rule depicts the process.

$$\begin{aligned} &R(Gen\_RK) = \forall x_6 \in X_6| \\ &x_6[4] := Gen\_RK(x_6[1], x_6[2], x_6[3]) \wedge \\ &X_6' = X_6 \cup \{x_6[4]\} \end{aligned} \tag{39}$$

Users send re-encryption keys to the *RSP*. The following rule is mapped to the transition *S_R_K* to depict the process.

$$\begin{aligned} &R(R\_S\_K) = \forall x_7 \in X_7, \forall x_8 \in X_8| \\ &x_8[5] := x_7[4] \wedge \\ &X_8' = X_8 \cup \{x_8[5]\} \end{aligned} \tag{40}$$

To encrypt the data, *EDSP* first generates a prime number and sends it to the requesting user. The following two transitions depict the process.

$$\begin{aligned} &R(Gen\_P1) = \forall x_9 \in X_9| \\ &x_9[6] := Gen\_\Pr ime() \wedge \\ &X_9' = X_9 \cup \{x_9[6]\} \\ &R(S\_P1) = \forall x_{10} \in X_{10}, \forall x_{11} \in X_{11}| \\ &x_{11}[5] := x_{10}[6] \wedge \\ &X_{11}' = X_{11} \cup \{x_{11}[5]\} \end{aligned} \tag{41}$$

The user transforms the message into first level encrypted message according to (22). The process is highlighted by the following formula.

$$\begin{aligned} &R(FL\_E) = \forall x_{12} \in X_{12}| \\ &x_{12}[7] := encrypt(x_{12}[5], x_{12}[6]) \wedge \\ &X_{12}' = X_{12} \cup \{x_{12}[7]\} \end{aligned} \tag{42}$$

**Table 7** Data types for EMReS

| Data type | Description |
|---|---|
| **g** | Number belonging to group $G_1$ of prime order q |
| **Z** | A number, e(g, g) belonging to group $G_2$ |
| $\mathbf{U_i}$ | A number identifying user i |
| $\mathbf{SK_i}$ | Secret key of user i |
| $\mathbf{PK_i}$ | Public key of user i |
| $\mathbf{RK_{i\longrightarrow j}}$ | Re-encryption key from user i to j |
| $\mathbf{SK_{EDSP}}$ | Secret key of EDSP |
| $\mathbf{PK_{EDSP}}$ | Public key of EDSP |
| $\mathbf{P_{sk1}}$ | A large prime number |
| **M** | String representing message to be encrypted |
| $\mathbf{C_{flevel}}$ | First level encryption of M |
| **r1** | A random number |
| $\mathbf{Z_{r1}}$ | A number calculated by division of $Z^{r1}$ and $P_{sk1}$ |
| **C** | Final encrypted text |
| $\mathbf{C_A}$ | Number representing $g^{r1xi}$ |
| $\mathbf{P_{sk1}}$ | A large prime number |
| $\mathbf{C_{EDSP}}$ | Number representing $Z^{r1xEDSP}$ |

Afterwards, the user sends the first level encrypted message to *EDSP*.

$$\begin{aligned}&R(S\_FL) = \forall x_{13} \in X_{13}, \forall x_{14} \in X_{14}|\\&x_{14}[7] := x_{13}[7] \wedge\\&X'_{14} = X_{14} \cup \{x_{14}[7]\}\end{aligned} \tag{43}$$

*EDSP* generates r1 and transforms the first level encrypted message to final encrypted message. The following two transitions show the process.

$$\begin{aligned}&R(Gen\_r1) = \forall x_{15} \in X_{15}|\\&x_{15}[8] := Gen\_random() \wedge\\&X'_{15} = X_{15} \cup \{x_{15}[8]\}\\&R(E\_FL) = \forall x_{16} \in X_{16}|\\&x_{16}[9] := Gen\_Zr1(x_{16}[5], x_{16}[8], x_{16}[6]) \wedge\\&x_{16}[10] := encrypt(x_{16}[7], x_{16}[9]) \wedge\\&x_{16}[11] = Cal\_CA(x_{16}[4], x_{16}[2]) \wedge\\&X'_{16} = X_{16} \cup \{x_{16}[9], x_{16}[10], x_{16}[11]\}\end{aligned} \tag{44}$$

The final encrypted message is uploaded to the cloud by *EDSP*.

$$\begin{aligned}&R(U\_C) = \forall x_{17} \in X_{17}, \forall x_{18} \in X_{18}|\\&x_{18}[1] := x_{17}[10] \wedge x_{18}[2] := x_{17}[11] \wedge\\&X'_{18} = X_{18} \cup \{x_{18}[1], x_{18}[2]\}\end{aligned} \tag{45}$$

When any user requests *RSP* to decrypt *M*, *RSP* transforms $C_A$ to $C_{EDSP}$ and sends to *EDSP*.

$$\begin{aligned}&R(Get\_C) = \forall x_{19} \in X_{19}, \forall x_{20} \in X_{20}|\\&x_{20}[2] := x_{19}[1] \wedge x_{20}[3] := x_{19}[2] \wedge\\&X'_{20} = X_{20} \cup \{x_{20}[2], x_{20}[3]\}\\&R(C\_C_{EDSP}) = \forall x_{21} \in X_{21}|\\&x_{21}[4] := Cal\_C_{EDSP}(x_{21}[3], x_{21}[5]) \wedge\\&X'_{21} = X_{21} \cup \{x_{21}[4]\}\\&R(S\_C\_C_{EDSP}) = \forall x_{22} \in X_{22}, \forall x_{23} \in X_{23}|\\&x_{23}[12] := x_{22}[4] \wedge\\&X'_{23} = X_{23} \cup \{x_{23}[12]\}\end{aligned} \tag{46}$$

Afterwards, *EDSP* generates a prime number that is sent to the requesting user.

$$\begin{aligned}&R(Gen\_P2) = \forall x_{24} \in X_{24}|\\&x_{24}[13] := Gen\_Prime() \wedge\\&X'_{24} = X_{24} \cup \{x_{24}[13]\}\\&R(S\_P2\_CFL) = \forall x_{25} \in X_{25}, \forall x_{26} \in X_{26}|\\&x_{26}[8] := x_{25}[13] \wedge\\&x_{26}[7] := Cal\_C_{flevel}(x_{25}[10], x_{25}[9], x_{25}[13]) \wedge\\&X'_{26} = X_{26} \cup \{x_{26}[8], x_{26}[7]\}\end{aligned} \tag{47}$$

The user decrypts the encrypted message using (29).

$$\begin{aligned}&R(C\_M) = \forall x_{27} \in X_{27}|\\&x_{27}[6] := decrypt(x_{27}[7], x_{27}[13]) \wedge\\&X'_{27} = X_{27} \cup \{x_{27}[6]\}\end{aligned} \tag{48}$$

*Verification property* The aim of verification was to ensure that the proposed system works according to the specifications and produce the results correctly. The properties that are verified are following:

– Encryption of the message by user and *EDSP* is done correctly and as specified by the system.

**Table 8** Places and mappings used in HLPN model of EMReS

| Place | Mapping |
|---|---|
| **$\varphi$ (EDSP)** | $\mathbb{P}$ (Ui× SKi× PKi× g× Z× $P_{SK1}$× $C_{flevel}$× $r_1$× $Zr_1$× Zr1× C× $C_A$× $C_{EDSP}$× $P_{SK2}$ ) |
| **$\varphi$ (RSP)** | $\mathbb{P}$ (Ui× C× $C_A$× $C_{EDSP}$× $RK_{i\longrightarrow j}$ ) |
| **$\varphi$ (User)** | $\mathbb{P}$ (Ui× SKi× PKi× $RK_{i\longrightarrow j}$× $P_{SK1}$× M× $C_{flevel}$× $P_{SK2}$ ) |
| **$\varphi$ (DSP)** | $\mathbb{P}$ (C× $C_A$) |

– Decryption requests are handled correctly by *EDSP* and *RSP*, and after decryption user gets the original data that was uploaded by the uploading user.
– Unauthorized users are not able to decrypt the message.

The above given model was translated to *SMT-Lib* and verified thorough *Z3* solver. The solver showed that the model is workable and executes according to the specified properties. *Z3* solver took 0.0321 seconds to upload data of user after encryption and download and decrypt for another user in the group.

## 6 Conclusion

In the *CMReS*, most of the encryption, decryption, and re-encryption operations are offloaded on trusted entity and cloud that improve the resource utilization on the mobile device. However, the trusted entities are under the control of the client organization and can be upgraded for gracefully handling the requests from the entire organization. Additionally, the re-encryption responsibilities of the trusted entity are also offloaded on the cloud without affecting the privacy of the user that improves the scalability of the system. The compromising of the authorized group member does not affect the security of the whole system due to the transformation of the uploaded messages with the personal private key of the data owners in the proposed *CMReS*. Moreover, there is no need to re-encrypt the entire data partition unlike *CReS* due to change in group members that reduces the processing burden from the trusted entities and cloud. Furthermore, the proposed *CMReS* consume fewer resources on the mobile device while performing encryption, decryption, and re-encryption operations as compared to the existing schemes.

The trusted-entity is under the control of client organization and responsible to handle the requests from the users belonging to the same virtual organization. Although the *CMReS* offloads the encryption and decryption operation on trusted entity, the involvement of trusted entity may affect the scalability of system. We are planning to enhance the proposed *CMReS* so that the encryption/decryption tasks can be offloaded to the cloud in a trusted mode with minimum processing burden on the mobile device.

## References

1. Shamsi, J., Khojaye, M.A., Qasmi, M.A.: Data-intensive cloud computing: requirements, expectations, challenges, and solutions. Journal of Grid Computing **11**, 281–310 (2013)
2. Khan, A.R., Othman, M., Madani, S.A., Khan, S.U.: A survey of mobile cloud computing application models. Communications Surveys & Tutorials, IEEE **16**, 393–413 (2014)
3. Kumar, K., Lu, Y.H.: Cloud computing for mobile users: Can offloading computation save energy? Computer **43**, 51–56 (2010)
4. Lorido-Botran, T., Miguel-Alonso, J., Lozano, J.A.: A review of auto-scaling techniques for elastic applications in cloud environments. Journal of Grid Computing **12**, 559–592 (2014)
5. Khan, A.N., Mat Kiah, M., Khan, S.U., Madani, S.A.: Towards secure mobile cloud computing: a survey. Futur. Gener. Comput. Syst. **29**, 1278–1299 (2013)
6. Khan, A.R., Othman, M., Khan, A.N., Abid, S.A., Madani, S.A.: MobiByte: an application development model for mobile cloud computing. J. Grid Comput. (2015). doi:10.1007/s10723-015-9335-x
7. Shiraz, M., Gani, A., Shamim, A., Khan, S., Ahmad, R.W.: Energy efficient computational offloading framework for mobile cloud computing. Journal of Grid Computing **13**, 1–18 (2015)
8. Murty, J.: Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB: O'Reilly Media, Incorporated (2008)
9. (February 07, 2013). *Setting up the HP Cloud Drive*. Available: http://h10025.www1.hp.com/ewfrf/wc/document?cc=us&lc=en&dlc=en&docname=c02948489
10. Nathani, A., Chaudhary, S., Somani, G.: Policy based resource allocation in IaaS cloud. Futur. Gener. Comput. Syst. **28**, 94–103 (2012)
11. Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, I.M., Montero, R., Wolfsthal, Y., Elmroth, E., Caceres, J.: The reservoir model and architecture for open federated cloud computing. IBM J. Res. Dev. **53**, 1–11 (2009)
12. Force.com Apex Code Developer's Guide. Available: http://www.salesforce.com/us/developer/docs/apexcode/index.htm
13. (September 02, 2012). Google App Engine. Available: https://developers.google.com/appengine
14. Ali, M., Khan, S.U., Vasilakos, A.V.: Security in cloud computing: Opportunities and challenges. Inf. Sci. **305**, 357–383 (2015)
15. Hashemi, S.M., Ardakani, M.R.M.: Taxonomy of the security aspects of cloud computing systems-a survey. Int. J. Appl. Inf. Syst. **4**, 21–28 (2012)

16. (April 24, 2013). Mobile cloud computing: $9.5 billion by 2014, Juniper, Technical Report (2010). Available: http://www.juniperresearch.com/reports/mobile_cloud_applications_and_services
17. (December 20, 2012). Security in the Cloud, Clavister White Paper (2009). Available: http://www.ciosummiteu.com/media/whitepapers/Clavister-security-in-the-cloud.pdf
18. Tysowski, P.K., Hasan, M.A.: Re-encryption-based key management towards secure and scalable mobile applications in clouds. IACR Cryptology ePrint Archive **668**, 2011 (2011)
19. Zhao, G., Rong, C., Li, J., Zhang, F., Tang, Y.: Trusted data sharing over untrusted cloud storage providers, presented at the IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom '10), Washington, DC, USA (2010)
20. Yang, J., Wang, H., Wang, J., Tan, C., Yu, D.: Provable data possession of resource-constrained mobile devices in cloud computing. Journal of Networks **6**, 1033–1040 (2011)
21. Itani, W., Kayssi, A., Chehab, A.: Energy-efficient incremental integrity for securing storage in mobile cloud computing, presented at the International Conference on Energy Aware Computing (ICEAC '10) Cairo, Egypt (2010)
22. Ren, W., Yu, L., Gao, R., Xiong, F.: Lightweight and compromise resilient storage outsourcing with distributed secure accessibility in mobile cloud computing. Tsinghua Science & Technology **16**, 520–528 (2011)
23. Yu, S., Wang, C., Ren, K., Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud computing, presented at the Proceedings IEEE (INFOCOM '10) NJ, USA (2010)
24. Jia, W., Zhu, H., Cao, Z., Wei, L., Lin, X.: SDSM: A secure data service mechanism in mobile cloud computing, presented at the IEEE Conference on Computer Communications Workshops (INFOCOM '11) Shanghai, China (2011)
25. Zhou, Z., Huang, D.: Efficient and secure data storage operations for mobile cloud computing, presented at the 8th International Conference on Network and Service Management (CNSM '12), AZ, USA (2012)
26. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. (TISSEC) **9**, 1–30 (2006)
27. Khan, A.N., Kiah, M.M., Ali, M., Madani, S.A., Shamshirband, S.: BSS: block-based sharing scheme for secure data storage services in mobile cloud environment. J. Supercomput. **70**, 946–976 (2014)
28. Khan, A.N., Kiah, M.M., Madani, S.A., Ali, M., Shamshirband, S.: Incremental proxy re-encryption scheme for mobile cloud computing environment. J. Supercomput. **68**, 624–651 (2014)
29. Ivan, A., Dodis, Y.: Proxy cryptography revisited, presented at the Proceedings of the Network and Distributed System Security Symposium (NDSS '03), San Diego, California (2003)
30. Green, M., Ateniese, G.: Identity-based proxy re-encryption, presented at the Applied Cryptography and Network Security (ACNS '07), Zhuhai, China (2007)
31. Emura, K., Miyaji, A., Nomura, A., Omote, K., Soshi, M.: A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. Inf. Secur. Practice Experience **5451**, 13–23 (2009)
32. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) Advances in Cryptology — EUROCRYPT'98, vol. 1403, pp. 127–144. Springer, Berlin Heidelberg (1998)
33. (FIPS 180-3, 2008). SHA-256. Secure Hash Algorithm. National Institute of Science and Technology
34. Curino, C., Jones, E.P., Popa, R.A., Malviya, N., Wu, E., Madden, S., Balakrishnan, H., Zeldovich, N.: Relational cloud: A database-as-a-service for the cloud, presented at the Proceedings of the 5th Biennial Conference on Innovative Data Systems Research, Pacific Grove, CA (2011)
35. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Advances in Cryptology, pp. 10–18. Springer (1985)
36. (September 09, 2012). Android Top Command To Get CPU Usage and Memory Usage. Available: http://www.javachartingandroid.com/2011/04/android-top-command-to-get-cpu-usage-and-memory-usage/
37. Android. (2015, March 23, 2015). Monitoring the Battery Level and Charging State. Available: http://developer.android.com/training/monitoring-device-state/battery-monitoring.html
38. Android. (2015, 19 April, 2015). How to get Current Time. Available: http://developer.android.com/reference/java/lang/System.html
39. (September 05, 2012). Adjusting Application Performance. Available: https://developers.google.com/appengine/docs/adminconsole/performancesettings
40. (August 12, 2012). *Remote API for Java*. Available: https://developers.google.com/appengine/docs/java/tools/remoteapi#Configuring_Remote_API_on_an_App_Engine_Client
41. (August15,2012).Google Cloud Storage Java API Overview. Available: https://developers.google.com/appengine/docs/java/googlestorage/overview
42. De Caro, A., Iovino, V.: "jPBC: Java pairing based cryptography, presented at the IEEE Symposium on Computers and Communications (ISCC '11) Kerkyra (2011)
43. (December 20, 2012). Java Pairing Based Cryptography Library. Available: http://gas.dia.unisa.it/projects/jpbc/index.html
44. (September 2000). Certicom, Standards for Efficient Cryptography, SEC 2: Recommended Elliptic Curve Domain Parameters, Version 1.0. Available: http://www.secg.org/download/aid-386/sec2_final.pdf
45. (August 23, 2012). Proportional Set Size. Available: http://lwn.net/Articles/230975/
46. Ali, M., Dhamotharan, R., Khan, E., Khan, S.U., Vasilakos, A.V., Li, K., Zomaya, A.Y.: SeDaSC: secure data sharing in clouds. IEEE Syst. J. (2015). doi:10.1109/JSYST.2014.2379646