CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
*Concurrency Computat.: Pract. Exper.* 2016; **00**:1–16
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

# A New Outsourcing Conditional Proxy Re-Encryption Suitable for Mobile Cloud Environment †

Junggab Son[1], Donghyun Kim[1], Md Zakirul Alam Bhuiyan[2], Rasheed Hussain[3], and Heekuck Oh*

[1]*Department of Computer Science, Kennesaw State University, Marietta, GA 30060, USA. E-mail: {json, donghyun.kim}@kennesaw.edu*
[2]*Department of Computer and Information Sciences, Fordham University, Bronx, NY 10458, USA E-mail: zakirulalam@gmail.com*
[3]*Institute of Information Sciences, Innopolis University, Innopolis, Russia, E-mail: r.hussain@innopolis.ru*

SUMMARY

The mobile cloud is a highly heterogenous and constantly evolving network of numerous portable devices utilizing the powerful back-end cloud infrastructure to overcome their severe deficiency in computing resource and offer various services such as data sharing. Inherently, in mobile cloud, the risk of user privacy invasion by the cloud operator is high. The conditional proxy re-encryption (CPRE) is a useful concept for secure group data sharing via cloud while preserving the privacy of the shared data from any unintended third parties including the cloud operator. Unfortunately, the state-of-art CPRE is not particularly designed for mobile cloud environment, and therefore imposes heavy burdens to the weak mobile cloud clients. This paper introduces a new CPRE scheme, namely the CPRE for Mobile Cloud (CPRE-M), which utilizes the back-end cloud to the extreme extent so that the overhead of terminals is drastically reduced. Specifically, our scheme outsources a significant amount of computation overhead caused by the followings functions at terminals, (a) re-encryption key generation, (b) condition value change, and (c) decryption, to the cloud. The proposed scheme also allows users to verify the correctness of outsourced computation under refereed delegation of computation (RDoC) model. Our simulation results show CPRE-M outperforms its existing alternatives.


Received . . .

KEY WORDS: Cloud computing; mobile cloud; proxy re-encryption; conditional proxy re-encryption; outsourcing computation.

## 1. INTRODUCTION

Thanks to the recent advancements in mobile computing and cloud computing technologies, the concept of mobile cloud computing (MCC) is born [1]. In comparison to the existing distributed network systems, MCC is distinguished by its heterogeneity as it consists of (a) various mobile terminals, most of which are personally owned, resource-constraint, and battery-powered, and (b) resource-rich back-end (mostly-public) cloud whose main role is to support the weak mobile

*Correspondence to: Department of Computer Science and Engineering, Hanyang University, Ansan, South Korea. E-mail: hkoh@hanyang.ac.kr.
†This research was supported in part by the NRF (National Research Foundation of Korea) grant funded by the Korea government MEST (Ministry of Education, Science and Technology) (No. NRF-2015R1D1A1A09058200). This research was also supported in part by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2016-H8501-16-1018) supervised by the IITP (Institute for Information & communications Technology Promotion).

terminals. Many of recent reports suggest that mobile cloud will be tightly interrelated with and improve the quality of our daily life. However, such benefits would not be realized unless various security and privacy concerns are addressed [2, 3, 4].

Mobile devices such as smart phones and tablets are equipped with various kind of sensors such as camera, microphone, and GPS, and tend to produce a significant amount private data for various applications which rely on the back-end cloud for data sharing among the users in the applications. This means that the back-end cloud is one of the most vulnerable points in the mobile cloud applications where the private information of the users can be leaked. Therefore, in order to expedite the adoption of the mobile cloud applications, sensitive user information which are being processed at the cloud has to be carefully managed such that no user privacy can be infringed, even by the cloud operators [5, 6, 7]. Apparently, this is a very challenging issue because, by nature, as a cloud operator has less information of the user data, it will be able to provide less services.

A proxy re-encryption (PRE) is a cryptosystem for secure data sharing via a public storage, and therefore, is a useful concept to offer a secure and privacy preserving group data sharing via public cloud for mobile cloud clients. Unfortunately, the existing PRE schemes are not particularly designed for mobile cloud environment and thus suffer from one common security concern called the abuse of re-encryption. In detail, once a user $U_i$ sends a re-encryption (privilege) key to the cloud storage for a new group member $U_j$, $U_j$ can collude with the cloud storage operator to decrypt all of the old messages originated from $U_i$ for the other group members even though $U_j$ is not authorized to access them. Instead, the *conditional proxy re-encryption (CPRE)*, which is also not particularly designed for mobile cloud environment, can be used for secure data sharing [8, 9] among mobile cloud clients.

In CPRE, $U_i$ creates encrypted messages as well as re-encryption keys (one for each member) with a certain condition value (for the whole group). In detail, given a group, $U_i$ first selects a condition value $w$. Then, use $w$ and its own key to encrypt the messages to be shared. At the same time, $U_i$ creates a re-encryption key for each member of the group using $w$ and the public key of the member. Then, the encrypted messages, $w$, and the re-encryption keys are sent to the cloud storage for message re-encryption and distribution like PRE. Once the membership of the group is changed, $U_i$ selects a new condition value $w'$ and repeat the process for the new messages generated after the group change as well as the existing messages in the cloud storage. In this way, any revoked member left the group cannot access new messages, and any new member cannot access old messages.

While the CPRE is suitable for secure data sharing in MCC, it suffers from one significant drawback that it imposes very high overhead on weak mobile cloud terminals, especially when the group is large and the shared data is dynamically changing, and this is likely to happen in MCC environment as we explained earlier. Previously, in [10], Son et al. pointed out that inefficiency of the existing CPRE scheme in point of mobile users, and proposed *efficient CPRE (E-CPRE)* to reduce number of generated re-encryption key which is burden of users. However, E-CPRE still imposes lots of overhead on weak mobile devices. To address this issue, this paper attempts to further reduce the overhead on the weak mobile devices by further utilizing the back-end cloud.

**Main Contributions.** In this paper, on the basis of [11], we introduce a new version of CPRE, namely the CPRE for mobile cloud computing (CPRE-M). The proposed scheme can delegate a partial computation during the process of data sharing using CPRE in three folds.

First, our scheme will move a part of user overhead at the initial re-encryption key generation stage. Second, when the membership of the group changes, in CPRE-M, the originator only needs to select a new condition value and upload it to the cloud. Unlike the other existing CPRE schemes including E-CPRE, in CPRE-M, the only tasks required for the originator of messages to perform when the group membership is changed are computing a *condition value changing key (CCK)*, which includes a new condition value, and sending the CCK to the cloud storage. Then, the cloud storage will use the CCK to transform existing ciphertexts such that the messages inside the ciphertexts are encrypted using the new conditional value. Third, our scheme also will move a part of user overhead at the decryption process. Our scheme reduces computation overhead drastically by adopting key



DOI: 10.1002/cpe

blind technique. In addition, we proposed another approach for reducing security risk as well as checkability with *refereed delegation of computation (RDoC)* model.

Clearly, the overhead of the message originator under the group membership change (this means condition value will be changed in our scheme) is drastically reduced. In addition, the advantage of CPRE-M over E-CPRE (and the other existing CPRE schemes) is getting magnified as the amount of the data in the cloud storage increases. As a result, CPRE-M is much more desirable than the other existing CPRE schemes for secure data sharing in MCC environment.

*Organization of Paper.* The rest of this paper is organized as follows. Section 2 represents state-of-art proxy re-encryption scheme and some previous work about secure outsourcing computation of attribute based encryptions. Section 3 describes some preliminaries. Section 4 discusses our system and security models. Our main contribution, the new conditional proxy re-encryption scheme with much lower user overhead, is presented in Section 5. In section 6, we propose another construction with checkability under RDoC model. The security and efficiency analysis of the proposed scheme is in Section 7. Finally, we conclude this paper in Section 8.

## 2. RELATED WORK

A concept of *proxy re-encryption (PRE)* which can delegate decryption rights was introduced by Mambo and Okamoto [12]. At this time, PRE was just an efficiency improvement of traditional cryptosystem. A user has to perform the process of decrypt and encrypt again to share encrypted data with the other user. Using PRE, a user can simply generate re-encrypted ciphertext with one encryption operation, rather than decrypt and then encrypt process. In 1998, M. Blaze et al. proposed the first PRE scheme, which the re-encryption key has bi-directional property [13].

Ateniese et al. [14] proposed proxy re-encryptions to secure distributed storage. In this scheme, the originator of data encrypts data with symmetric data keys, and encrypts the result with master public key. If users want to access the data, the server uses proxy re-encryption to directly re-encrypt the appropriate data key from the master public key to a granted user's public key. However, there is a side effect. If Alice creates re-encryption key for Bob, all messages encrypted by Alice's private key including previously created and future ciphertext can be re-encrypted by the re-encryption key. We called this side effect as abuse of re-encryption key. And this scheme has security flaw of collusion attack between the untrusted server and a revoked malicious user, which enables them to learn the decryption keys of all the encrypted blocks.

To address this problem, the *conditional proxy re-encryption (CPRE)* has been introduced in the literature [8, 9]. In CPRE, The originator creates ciphertext as well as re-encryption keys with a certain condition value. The condition value in the encrypted data is cancelled out during re-encryption process, then a target user can receive re-encrypted ciphertext without condition value.

Nevertheless CPRE can solve the abuse of re-encryption key problem, it still has drawback in point of efficiency. If a user has access permission more than one (it can be represented by the number of condition values), different as PRE, the originator has to generate re-encryption keys as many as the number of condition values.

Recently, the research issues on the PRE and CPRE are focused on its functionality. In 2009, G. Ateniese et al. proposed *key-private PRE (KPRE)* to hide a relationship between a re-encryption key and ciphertext [15]. M. Green and G. Ateniese proposed *identity based PRE (IPRE)*, where ciphertexts are transformed from one identity to another [16]. To improve the rack of an abundant expressions on the condition values of CPRE schemes, Fang et al. proposed *interactive CPRE with fine grain policy (ICPRE-FG)* [17]. In 2011, Shao et al proposed *identity based CPRE (IBCPRE)* which is allowed to transform a subset of an originator's ciphertext under an identity to other ciphertexts under another identity [18]. In 2012, L. Fang et al. proposed *Hierarchical CPRE (HCPRE)* which is the hierarchical extension of CPRE by adopting of a vector of keywords as condition value [19].

In [10], Son et al. pointed out that the existing CPRE can suffer from a serious efficiency issue if the size of the group is large and the membership of the group changes frequently. This is because



DOI: 10.1002/cpe

Table I. Notations.

| Notation | Description |
|---|---|
| $q$ | $k$-bit prime number |
| $\mathbb{Z}_q$ | Integers modulo $q$ |
| $\mathbb{G}, \mathbb{G}_T$ | Cyclic group with prime order $q$ |
| $g$ | Generator of $\mathbb{G}$ |
| $e$ | Bilinear pairing that satisfies with $\mathbb{G} \times \mathbb{G}_T$ |
| $U_i$ | The user $i$ |
| $pk_i, sk_i$ | Public/private key pair of $U_i$ |
| $m$ | Data, $m = \{0,1\}^k$ |
| $w$ | Condition value |
| $s$ | Random number that is included in $\mathbb{Z}_q^*$ |
| $CT_i$ | Ciphertext generated by $U_i$ |
| $RK_{part\_j}$ | Partial re-encryption key for $j$ |
| $RK_{(i \to j)}$ | Re-encryption key for $U_i \to U_j$ |
| $CCK$ | Condition value Changing Key |
| $H_1$ | $\{0,1\}^* \to \mathbb{Z}_q$ |
| $H_2$ | $\{0,1\}^* \to \mathbb{G}$ |
| $H_3$ | $\mathbb{G} \to \mathbb{Z}_q$ |
| $H_4$ | $\mathbb{G} \to \{0,1\}^*$ |

whenever membership changes, the re-encryption key for each group member is newly generated by each data originator and the existing messages in the cloud have to be encrypted again using this new condition value. To alleviate the overhead, they proposed an *efficient CPRE (E-CPRE)* in which whenever the membership of the group changes, a data originator needs to select a new condition value and sends it to the cloud server, but does not need to create and upload the re-encryption keys for the other group members. When the group membership changes, E-CPRE works more efficiently than the existing CPRE since it reduces the overhead of the data originator to compute new re-encryption keys and upload them to the cloud. However, both CPRE and E-CPRE require to encrypt the existing messages on the cloud using the new condition value and upload them. As a result, they are inefficient for data sharing in mobile cloud environment.

With recent advances in cloud computing, several related work about outsourcing *attribute based-encryption (ABE)* schemes are proposed to mitigate its drawback in which decryption process needed a lot of computation [20, 21, 22]. Recently, J. Li et al. proposed outsourcing ABE that can delegate a computation during key generation and decryption processes with checkability [23]. Consequently, ABE becomes one of the most promising solution for data sharing among resource-constraint mobile devices. Meanwhile, the PRE and CPRE scheme still cannot use the property of cloud computing. Since it already can delegate computation of re-encryption, it has possibility of improvement of user efficiency by delegating more computation to the cloud computing. Therefore, it can be better solution for data sharing scheme in mobile cloud environment, if the process of re-encryption key generation, decryption, and revocation processes are also delegable.

## 3. PRELIMINARIES

### 3.1. Notations

The notations used in this paper are listed in Table I.

### 3.2. Cryptographic Background

Next, we introduce two important definitions.



DOI: 10.1002/cpe

*Definition 1* (Bilinear map)
A bilinear map is a map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties [24].

(a) Computable: there exists an efficiently computable algorithm for computing $e$,

(b) Bilinear: for all $h_1, h_2 \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p, e(h_1^a, h_2^b) = e(h_1, h_2)^{ab}$, and

(c) Nondegenerate: $e(g, g) \neq 1$, where $g$ is a generator of $\mathbb{G}$.

*Definition 2* (DBDH)
The Decisional Bilinear Diffie-Hellman (DBDH) problem in groups $(\mathbb{G}, \mathbb{G}_T)$ is, given a tuple $(g, g^a, g^b, g^c, Z) \in \mathbb{G}^4 \times \mathbb{G}_T$ with unknown $a, b, c \in_R \mathbb{Z}_q$, whether $Z = e(g, g)^{abc}$. A polynomial-time algorithm $\mathcal{B}$ has advantage $\epsilon$ in solving the DBDH problem in groups $(\mathbb{G}, \mathbb{G}_T)$, if

$$|(Pr[(g, g^a, g^b, g^c, \mathbb{Z} = e(g, g)^{abc}) = 1] - Pr[(g, g^a, g^b, g^c, \mathbb{Z} = e(g, g)^d) = 1])| \geq \epsilon,$$

where the probability is taken over the random choices of $a, b, c, d \in \mathbb{Z}_q$, the random choice of $g$ in $\mathbb{G}$, and random bits consumed by $\mathcal{B}$.

## 4. SYSTEM MODEL, THREAT MODELS AND SECURITY MODELS

### *4.1. System Model*

Fig. 1. illustrates a system model for data sharing in MCC. We define three different entities: outsourcing server, cloud storage, and users. In real application, cloud computing service provides both computation and storage. However, we divide it into two different entities to help understanding. The detail description for entities is as follows.

- **Outsourcing Server**: This entity has important role in our system model. Outsourcing Server completes delegated expensive operations to overcome disadvantages that the re-encryption key generation and decryption phase in typical CPRE requires a lot of overload operations at users.
- **Cloud Storage**: It provides the users with computing and storage resources. Basically, the cloud storage maintains the users' data in encrypted form. To process users' request, such as store, download, and re-encryption, cloud storage sends proper data to the outsourcing server.
- **Users**: This entity has large data files to be stored in the cloud and relies on the cloud for data maintenance and computation. Users can be either individual consumers or organizations. The user can connect to the cloud with various devices. It includes resource-constrained mobile devices such as smart phone or tablet. Therefore, it can be relieved of the burden of maintaining and computation by storing the large data files in the cloud storage. In a real application, the user can be the originator of data files as well as destination of data sharing. However, we define two entities: originator who creates a content, users in sharing group who consume the content.

Based on the system model, we consider a following service scenario: Alice who is originator of contents (we will use the term "contents" as valuable data) wants to share contents securely with a group of users, and there are a number of sharing groups that Alice participated. Mobile messenger application can be the most relevant example to explain our system model.

Alice may want to share contents with a user, but also may generate a sharing group depending on their interest. To do so, Alice has to generate re-encryption key for the users in the sharing group as an initial step. Once the contents are stored at the cloud storage, all of the users in the sharing group can obtain the contents after re-encryption process which should be done by the outsourcing server.

Alice also controls sharing of the contents among groups using condition values of CPRE. Alice allocates a condition value to the re-encryption key for users in a sharing group, and encrypts the contents with the same condition value. By doing this, only users in the sharing group can obtain the contents.



DOI: 10.1002/cpe

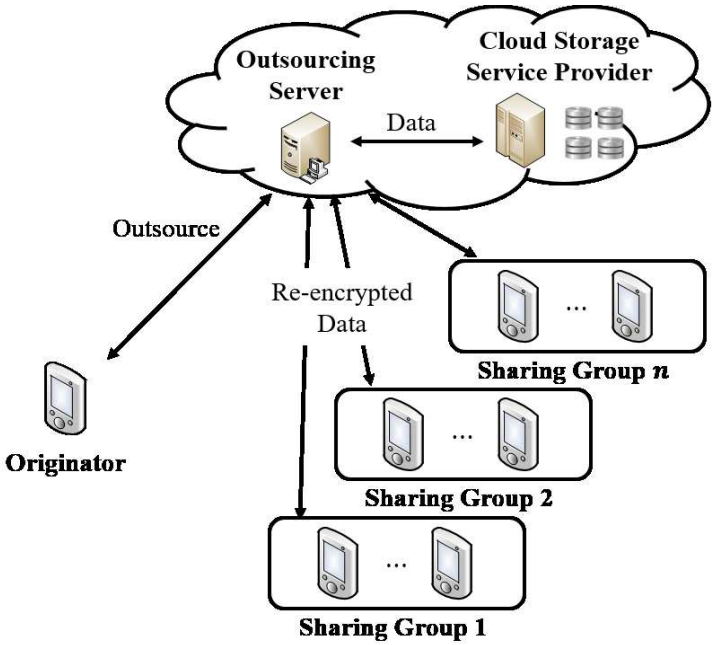


Figure 1. System Model.

### *4.2. Threat and Security Models*

The aim of the proposed scheme is delegating partial heavy computation of three major functions: re-encryption key generation, decryption, and condition value changing process, which can be burden to mobile users, to the outsourcing server. Based on the system model and our aims, we consider that the following threat models are possible.

We assume an honest-but-curious outsourcing server. The outsourcing server in this model runs a given protocol as best as it can and pays attention to users' data simultaneously. Therefore, there is possibility of passive attacks such as looking at contents or eavesdropping in data transfer process. Since the cloud server can take on a role of outsourcing server, in the following descriptions we will use outsourcing server and cloud server interchangeably.

In addition, users can be a potential attacker. A user may try to obtain contents which is not allowed to the user. However, the user can try only a few kind of attack methods because most operations of the proposed scheme are done by outsourcing server. The user can eavesdrop ciphertext, re-encryption key, partially decrypted ciphertext, and condition value changing key. Using these information, we assume that the user tries to obtain contents.

We argue that the data sharing scheme is secure when it satisfies the following conditions.

(a) No polynomial time extractor exists that can generate re-encryption key illegally by carrying out partial re-encryption key computation.

(b) No polynomial time extractor exists that can recover the condition value or insert another condition value to the ciphertext.

(c) No polynomial time extractor exists that can recover the original data files by carrying out partial decryptions.



DOI: 10.1002/cpe

## 5. PROPOSED SECURELY OUTSOURCING CPRE

In this chapter, we design a new CPRE scheme, namely CPRE-M, which can partially delegate the operations which are traditionally burdened to the users in the existing CPRE to the cloud and further reduce the overhead of user in three ways.

First, our scheme delegates the re-encryption key generation process. In detail, we introduce a new three-phase strategy for $RK$ generation. At the first phase, an originator initiate the process with generating and sending a $DK$. The second phase is performed by the outsourcing server, which is the process of generating partial re-encryption keys. Finally, $RK$ will be finalized by the originator after receiving the result of the second phase. Although the re-encryption key generation process is performed only once at the initial setup of the system, our scheme can significantly reduce a burden of the originators by delegating heavy exponential computation to the cloud.

Second, our scheme moves a part of the decryption process from the users to the cloud securely by applying a random *blind factor* strategy [21, 23]. For every request of partial decryption, the outsourcing server performs pairing operation with public key and re-encryption key of requested user. After receiving result of the partial decryption from the cloud, the user can complete the decryption with a single exponential operation without leaking of a private key or a plaintext.

Third, our scheme proposes *condition value changing key (CCK)* to deal with a member join and leave. Since a condition value is assigned to a sharing group, it must be renewed when the member of sharing group is changed. To renew the condition value, the originator should receive all of ciphertexts from the cloud storage, which allocated to the sharing group, encrypt again with a new condition value, and then send these ciphertexts to the cloud storage. On the other hand, our scheme uses the CCK to avoid this complicated and cumbersome procedure. When a condition value needs to be changed, the originator has to generate new condition value. Then, the originator generates CCK which contains new condition value and sends it to cloud. Once the outsourcing server receives CCK, it performs pairing operations with CCK and multiplies it with ciphertexts to change an old condition value to new one. Since the outsourcing server can transform all the ciphertexts which include old condition value using CCK, the originator does not need to manually retrieve the ciphertexts to change condition value like the other existing CPRE schemes.

### 5.1. Function Definition

Following definition describes the functions which will be used for our proposed scheme.

- $Setup(global)$: This algorithm takes global parameters as input. It outputs a public/private key pair $(pk, sk)$.
- $RkGen_{init}(sk)$: This is the initialization algorithm to delegate re-encryption key generation. It takes a private key $sk$ as input, and outputs a delegation key $DK$.
- $RkGen_{part}(pk_j, DK)$: This is the delegated re-encryption key generation algorithm. It takes a target user's public key $pk_j$ and a delegation key $DK$ as input, and outputs a partially generated re-encryption key $RK_{part_j}$.
- $RkGen(RK_{part_j})$: This algorithm completes a re-encryption key. It takes a partially generated re-encryption key $RK_{part_j}$ as input, and outputs a $RK_{i \to j}$.
- $Enc_1(m, pk)$: This is the first level encryption algorithm. It takes a plaintext $m$ and a public key $sk$ as input, and outputs a first level ciphertext $CT$.
- $Enc_2(m, w, pk)$: This is the second level encryption algorithm. It takes a plaintext $m$, condition value $w$, and a public key $pk$ as input, and outputs a second level ciphertext $CT_i$.
- $ReEnc(CT_i, RK_{i \to j}, w)$: This re-encryption algorithm converts a $U_i$'s ciphertext to $U_j$'s ciphertext. It takes a ciphertext $CT_i$ and a re-encryption key $RK_{i \to j}$, and condition value $w$ as input, it outputs a $U_j$'s ciphertext $CT_j$.
- $CCV(CT_i, CCK)$: This is the delegated condition value changing algorithm. It takes a second level ciphertext $CT_i$ and condition value changing key $CCK$ as input, and outputs renewed ciphertext $CT'$.
- $Decrypt_1(CT, sk)$: This algorithm decrypts first level ciphertext. It takes a first level ciphertext $CT$ and private key $sk$, and outputs a plaintext $m$.



DOI: 10.1002/cpe

- $Decrypt_{part}(CT, sf)$: This is the delegated decryption algorithm. It takes a ciphertext $CT_i$ and a secret factor $sf$ as input, and outputs a partially decrypted ciphertext $CT_{part}$.
- $Decrypt(CT_{part}, \gamma)$: This algorithm fully decrypts a partially decrypted ciphertext. It takes $CT_{part}$ and a nonce $\gamma$ that used to compute the $sf$ as input, and outputs plaintext $m$.

### 5.2. Setup

On input a security parameter $1^k$, the setup process first determines a large prime number $q$, $\mathbb{Z}_q$ which is integers modulo $q$, $\mathbb{G}$ and $\mathbb{G}_T$ which are two cyclic groups with prime order $q$, and bilinear pairing function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Next, the cloud chooses generator $g \in_R \mathbb{G}$, and four hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_q, H_2 : \{0,1\}^* \to \mathbb{G}, H_3 : \mathbb{G} \to \mathbb{Z}_q$, and $H_4 : \mathbb{G} \to \{0,1\}^*$. The global parameters are

$$((q, \mathbb{G}, \mathbb{G}_T, e), g, H_1, H_2, H_3, H_4).$$

A user generates a public/private key pair by performing $Setup(global)$ algorithm. It picks $sk_i \in_R \mathbb{Z}_q$ as private key, and computes $g^{sk_i}$ as public key. It outputs public/private key pair $(pk_i, sk_i)$.

### 5.3. Re-encryption Key Generation

To outsource the re-encryption key generation process, we divide the re-encryption key generation process into three different phases.

First phase should be done by an originator $U_i$. The $U_i$ generates the re-encryption key for a user $U_j$ as meaning of that the $U_i$ allows the $U_j$ to share the data. For the first phase, the $U_i$ performs $RkGen_{init}(sk)$: It picks $s \in_R Z_q$ and computes $g^{-s \cdot sk_i}$. It outputs $DK = g^{-s \cdot sk_i}$.

In the second phase, the $U_i$ requests partial re-encryption computation for a set of target users $\{U_j\}_{0 \le j \le \ell}$, where $\ell$ is number of users in sharing group, with sending the $DK$. Then outsourcing server performs $RkGen_{part_j}(pk_j, DK) = \{pk_j^{H_3(g^{-s \cdot sk_i})}, DK^{H_3(pk_j)}\}$ to generate a set of partial re-encryption keys, $\{RK_{part_j}\}_{0 \le j \le \ell}$. In this point, the outsourcing server computes some partial computation that needs heavy exponential operation.

Third, after receiving $\{RK_{part_j}\}_{0 \le j \le \ell}$, The $U_i$ performs $RkGen(RK_{part_j})$ to complete re-encryption keys.

$$\begin{aligned} RK_{(i \to j)} &= (rk_1, rk_2) \\ rk_1 &= (pk_j^{H_3(g^{-s \cdot sk_i})})^s \\ rk_2 &= g^{-s \cdot sk_i \cdot H_3(pk_j)}. \end{aligned}$$

We refer [8, 25] which makes relationship between sub re-encryption keys for the robustness against chosen ciphertext attack. These schemes are based on $n$-Quotient Bilinear Diffie-Hellman Assumption, and the schemes proved security by calculating complexity of DBDH. Inspired from the schemes, we designed the re-encryption key consisting of two values $(rk_1, rk_2)$. As the $rk_1$ includes $H_3(rk_2)$ and $rk_2$ includes $H_3(pk_j)$, it makes relationship between $rk_1$ and $rk_2$. In this way, the proposed scheme has robustness against chosen ciphertext attack.

Originally, in general CPRE schemes, an originator needs to receive a set of public keys of users to make re-encryption key. The originator in our scheme needs to pre-computed partial re-encryption key instead of the public keys. Hence, only sending the DK is calculated as communication overhead.

### 5.4. Data Encryption

A CPRE scheme has two types of encryption. The first level encryption generates a ciphertext which does not allow re-encryption. Therefore, the first level ciphertext is generated without condition value. And the second level encryption generates a ciphertext which allows re-encryption. The second level ciphertext includes condition value which is used to control decrypt permission. The user can choose the encryption level depending on the importance of the data. The first level and the second level encryption algorithms are as follows:



DOI: 10.1002/cpe

- $Enc_1(m, pk_i)$ : It picks $R \in_R \mathbb{G}$. It computes $r = H_1(m, R)$, and generates the first level ciphertext $CT = (C_1, C_2, C_3, C_4)$, $C_1 = g^r$, $C_2 = R \cdot e(g, pk_i)^{s \cdot H_3(pk_i^{-s})}$, $C_3 = m \oplus H_4(R)$, $C_4 = g^{s \cdot H_3(-s \cdot sk_i) \cdot H_3(pk_i)}$
- $Enc_2(m, w, pk_i)$ : It picks $R \in_R \mathbb{G}$. It computes $r = H_1(m, R)$, and generates the second level ciphertext $CT_i = (C_1, C_2, C_3, C_4), C_1 = g^r, C_2 = R \cdot e(pk_i, g^s)^W, C_3 = m \oplus H_4(R), C_4 = g^{s \cdot H_3(-s \cdot sk_i)}$,

where $W = H_1(w||H_4(C_4))$.

### 5.5. Data Re-encryption

The outsourcing server performs $ReEnc(CT_i, RK_{i \to j}, w)$ to re-encrypt the $CT_i$ by $U_j$'s request. It computes $CT_j$ as follows:

$$\begin{aligned}
CT_j &= (\overline{C}_1, \overline{C}_2, \overline{C}_3, \overline{C}_4) \\
\overline{C}_1 &= C_1, \\
\overline{C}_2 &= R \cdot C_2 \cdot e(g, rk_2^W \cdot rk_1) \\
&= R \cdot e(g, g)^{s \cdot sk_i \cdot H_3(pk_j) \cdot W} \cdot e(g, g^{-s \cdot sk_i \cdot W} \cdot pk_j^{s \cdot H_3(g^{-s \cdot sk_i})}) \\
&= R \cdot e(g, pk_j^{s \cdot H_3(g^{-s \cdot sk_i}) \cdot H_3(pk_j)}), \\
\overline{C}_3 &= C_3 = m \oplus H_4(R), \\
\overline{C}_4 &= C_4 = g^{s \cdot H_3(-s \cdot sk_i)}.
\end{aligned}$$

Through the re-encryption process, a second level ciphertext will be transformed to a first level ciphertext of the user $U_j$.

### 5.6. Changing Condition Value

In previous CPRE scheme, an originator has to retrieve its own ciphertext from the cloud storage and encrypt it again with new condition value whenever a condition value changed. It is an inefficient and cumbersome task to the originator, and therefore we design the system that the cloud can renew a condition value in a ciphertext with the originator issued CCK. For this, the originator performs following process to generate CCK with newly generated condition value $w'$. The originator computes $t = W' - W$, where $W' = H_1(w'||H_4(C_4))$, and sets $CCK = pk_o^t$, where $pk_o$ is public key of the outsourcing server (related secret key of the outsourcing server is $sk_o$). The user sends $CCK$ to the outsourcing server. Now, the outsourcing server performs $CCV(CT_i, CCK)$ as follows to generate $CT_i'$ which has newly issued condition value:

$$\begin{aligned}
C_2' &= C_2 \cdot e(pk_i^{-sk_o}, CCK) \\
&= R \cdot e(pk_i, g^s)^{W'}.
\end{aligned}$$

Now, the renewed ciphertext is $CT_i' = (C_1, C_2', C_3, C_4)$.

### 5.7. Partial Decryption

Surely, a user can decrypt a ciphertext with whole decryption process by itself as well as delegate a partial decryption process to the outsourcing server. For the delegation, we design the user computes secret factor $\gamma$ to hide plaintext against the outsourcing server while it performs delegated heavy pairing operations. The user $U_j$ receives $C_4$ from the outsorcing server for the partial decryption, and it picks $\gamma$, which is a nonce of $\mathbb{Z}_q$. Then, the $U_j$ computes $sf = g^{s \cdot H_3(g^{-s \cdot sk_i}) \cdot -sk_j \cdot H_3(pk_j) \cdot \gamma}$, and sends it back to outsourcing server. For a user's request of partial decryption, the outsourcing server performs $Decrypt_{part}(CT, sf)$ to compute $CT_{part}$ for $U_j$:

$$\begin{aligned}
CT_{part} &= C_2 \cdot e(g, sf) \\
&= R \cdot e(g, g)^{-\gamma}
\end{aligned}$$

The outsourcing server sends the $CT_{part}$ to the $U_j$.



DOI: 10.1002/cpe

### 5.8. Data Decryption

The $U_j$ can decrypt the first level ciphertext by computing $Decrypt_1(CT, sk_j)$, and also can decrypt the $CT_{part}$ by computing $Decrypt(CTpart, w, sk_j)$:

- $Decrypt_1(CT, sk_j)$: First, the $U_j$ computes $R = C_2 \cdot e(g, C_4)^{-sk_j}$, and $m = C_3 \oplus H_3(R)$. Next, the $U_j$ checks $g^{H_1(m,R)} = C_1$ to confirm the validity of the data.
- $Decrypt(CT_{part}, \gamma)$: $U_j$ can simply decrypt the partial decrypted ciphertext by following process. $R = CT_{part} \cdot e(g, g)^{\gamma}$, and $m = C_3 \oplus H_3(R)$. Next, the $U_j$ checks $g^{H_1(m,R)} = C_1$ to confirm the validity of the data.

## 6. CHECKABILITY OF THE PROPOSED SCHEME

The proposed scheme derives benefit by delegating partial computation to the cloud computing. However, a cloud service provider may be selfish to save computation, bandwidth or any other kind of resources. This selfishness can causes users to obtain incorrect keys or messages and it can effect on the delegation processes of the proposed scheme.

A service provider may execute only fraction of $CCV$ and partial decryption operation. Moreover, a service provider generates $RK_{part}$ with its own key, instead of a designated user. If an originator cannot check this abnormal behavior, the service provider will have $RK_{i \to SP}$, and will be able to re-encrypt a stored ciphertext. Although, the service provider cannot obtain plaintext from re-encrypted ciphertext without condition value, outsourcing computation has to detect this dishonest action. Therefore, we propose another function to provide checkability during outsourcing operations under the refereed delegation of computation model (RDoC).

RDoC model is formalized by [26, 27], and is widely used for outsourced computation [28, 29, 23]. In RDoC model, the originator is able to interact with multiple servers, in which $k$ number of servers cooperatively work together and it has a right output as long as there exists one server that follows the proposed protocol [23]. Compared with traditional single server model, RDoC has the advantages in terms of the security risk can be diffused to multiple servers which involved in. Using RDoC model, we will re-design the proposed scheme to detect dishonest action during outsourcing computations.

### 6.1. Re-encryption Key Generation

Generally, the originator can obtain public key and it's certificate to check a validity of the public key before generating a re-encryption key. However, this is an inconvenient task to users and we re-design it under RDoC model. To reduce security risk on the partial re-encryption key generation, the originator picks $s \in_R \mathbb{Z}_q$ and performs $RkGen_{init}(sk)$ to obtain $DK$. After this, the originator sends the $DK$ to $k$ different servers and each server returns result of $RkGen_{part}$. The originator will obtain a set of partially computed re-encryption key $\{RK_{part}[\ell]\}_{2 \le \ell \le k}$. Then the originator can detect the dishonest action by checking all the received data computed from $k$ different servers. Since the messages from all the honest servers will be same in honest computation, a user can obtain a set of same messages unless $k - 1$ servers did dishonest action.

### 6.2. Changing Condition Value

The condition value changing process is done by the outsourcing server and it stored to the cloud storage directly. Let us assume that the outsourcing server may ignore the request due to save computation. This attack will be easily succeeded as it is impossible for the originator to check the delegated computation. To deal with this, we re-design $CCV$ that any two or more servers among $k$ can change condition value of the remotely stored ciphertext. To make this possible, we generates $CCK$ with (2,t) secret sharing scheme [30]. Firstly, a user picks new condition value $w'$ and computes $W' = H_1(w'), t = W' - W$. Then, a user divides it to $k$-secret $\{t_\ell\}_{2 \le \ell \le k}$ using (2, $k$) threshold secret sharing scheme. A $CCK$ will be $CCK_\ell = (g^{W'}, r^{t_\ell}), 2 \le \ell \le k$. A user sends it randomly to $k$-different server. Suppose $j$ honest server applies $CCK$ on the stored ciphertext.



DOI: 10.1002/cpe

Table II. Comparison of Computational Overhead in user side

| | CPRE [8] | Efficient CPRE [10] | Proposed Scheme |
|---|---|---|---|
| Re-encryption key generation | $4t_e + 3t_m + 2t_h$ | $3t_e + 2t_m + 1t_h$ | $2t_e + 1t_m$ |
| Decryption | $1t_p + 2t_e + 1t_m + 2t_h$ | $1t_p + 2t_e + 3t_m + 2t_h$ | $1t_e + 2t_m + 1t_h$ |
| Changing a condition value | $1t_p + 4t_e + 4t_m + 3t_h$ | $1t_p + 2t_e + 1t_m + 2t_h$ | $2t_e + 1t_h$ |

Then the $CCV(CT_i, CCK)$ will be applied as follows: $C_1' = g^{W'}$, $C_2' = C_2 \cdot \prod e(pk_i, r_\ell^t) = C_2 \cdot e(pk_i, r)^{\sum t_\ell}, 2 \leq \ell \leq j$.

### 6.3. Partial Decryption

The user can detect a dishonest action during partial decryption process by using the same strategy as the process of the re-encryption key generation. Also, it can be detected by the idea of *redundancy* [23], which can provide checkability and detectability efficiently against the dishonest action. In the encryption process, a user firstly appends a redundancy $0^k$ on a message $m$, $m_T = m||0^k$, where $||$ is string concatenation. And a user encrypts it with the proposed scheme. After decryption, a user obtains $m_T$ and detects dishonest action by checking whether a redundancy $0^k$ is appended on the $m$.

## 7. ANALYSIS OF PROPOSED SCHEME

### 7.1. Security

In this section, we analyze security of the proposed scheme based on the security model described in Section 4.2. The main purpose of the proposed scheme is delegating heavy computational overhead during data sharing process using CPRE. Therefore, our security proofs will focus on the delegation to prove that the proposed scheme can securely delegate computation of CPRE.

*Theorem 1*
An attacker cannot recover a plaintext by carrying out multiple partial decryptions.

*Proof*
To outsource the partial decryption securely, we use a nonce as secret factor, which is represented $\gamma$, and is generated by a user who requests the partial decryption. Suppose there exists an oracle machine (oracle in short) $\mathcal{O}$ which can solve the partially decrypted message. Then, we can define the oracle as follows: $\mathcal{O}(e(g,g)^\gamma, \gamma')$. The oracle will output $true$, if it can determine $\gamma = \gamma'$ in polynomial time, otherwise output $false$. However, using the oracle, we can solve discrete logarithm problem. we assume that the attacker tries to solve $a$ from $g^a$, then the attacker can try following query: $\mathcal{O}e(g, g^a), a$. The discrete logarithm problem is well known difficulty and already proven that it is not a polynomial time solvable problem. Therefore, recovering plaintext in our scheme is as hard as discrete logarithm problem by a reduction. □

Through this blinding factor, our scheme is able to delegate partial decryption securely to the outsourcing server. The secret factor generates randomly in $\mathbb{Z}_q$, we argue that the obtaining information from partially decrypted ciphertexts is as hard as solving DBDH problem.

*Theorem 2*
An attacker cannot generate a new re-encryption key by carrying out multiple partial re-encryption keys.



DOI: 10.1002/cpe

*Proof*
A re-encryption key $rk_1$ is generated through two steps. The outsourcing server computes pre-process of a re-encryption key generation and sends it to an originator. Then the originator computes a secret part and completes a re-encryption key by merging two parts. Here, first part of $rk_1$ has only public value and the attacker can obtain it by eavesdropping a communication section. Although the attacker has a lot of this public values and re-encryption keys, it needs to know additional information to generates a new re-encryption key, which represents $s$ in the proposed scheme. The $s$ is a secret to prevent illegal computation of re-encryption key, and it will be offset by re-encryption. The only way to get the $s$ is solving discrete logarithm problem [31]. Therefore, illegal generation of re-encryption key is computationally hard under the problem. □

*Theorem 3*
An attacker cannot recover the condition value or insert another condition value to the ciphertext by carrying out multiple condition value changing keys.

*Proof*
Suppose that the attacker eavesdropped $CCK$s from communication section between an originator and an outsourcing server. Using these information, the attacker may try to obtain condition value to re-encrypt a ciphertext. This attempt is only possible in case if the attacker also eavesdropped the message and re-encryption key. To do so, the attacker needs to solve discrete logarithm problem to obtain a hashed condition value. The detail description of proof is same as Theorem 1. Assume that the attacker gets the hashed condition value, then the attacker can obtain the condition value with following probability by strong one-wayness of the cryptographic hash function [32]:

$$Pr[A(f(h_n), 1^n \in f^{-1}(f(h_n))] < \frac{1}{p(n)},$$

where $A$ is every probabilistic polynomial-time algorithm, $p(\cdot)$ is every polynomial, $f$ is hash function satisfies $f : \{0,1\}^* \rightarrow \{0,1\}^n$, $f^{-1}$ is inverse function of $f$, and $h_n$ is a random number over $0, 1^n$.

In addition, the attacker may eavesdrop a $CCK$ and try to change its condition value. In this case, the attacker also has to infer two condition values of $CCK$, and its probability is $\frac{1}{p(n^2)}$.Therefore, it is hard to succeed the attacks by carrying out multiple $CCK$s in polynomial time. □

### *7.2. Efficiency*

Our main design objective of CPRE-M is making efficient data sharing scheme for MCC by delegating the heavy computational overhead on the user side (especially the data originator) to the cloud during data sharing in mobile cloud. The greatest advantage of our scheme is that the cost of expensive re-encryption key generation, decryption and condition value changing are significantly reduced by partially delegating the tasks to server. Apparently, CPRE-M performs much more efficiently than the existing CPRE schemes from the user perspective.

First, we measure a computing time for four major operations, and then, numerically show that how much our scheme can delegate computational overhead of three functions: re-encryption key generation, decryption, changing a condition value. Finally, we verify it through actual simulation results. In Table II, we present the result of computational overhead comparison among CPRE, E-CPRE, and CPRE-M. In this Table, $t_p, t_e, t_m, t_h$ represent the computational cost of a bilinear pairing, an exponentiation, a multiplication, and a hash, respectively.

Our experiment is implemented on an Intel core (TM) i7 processor running at 1.5 GHZ, 8.00 GB of RAM, and SSD Serial ATA 3.0 Gbit/s drive with a 16 MB buffer. All algorithms are implemented using C language in Linux Ubuntu 12.04 LTS 32 bit. Our code uses pairing based cryptography (PBC) library version 0.5.12. All experimental results represent the average of 10 trials. In addition, we use 80-bit security parameter for a security parameter. It may not enough to guarantee security in modern computing environment. However, our aim of this experimental analysis is how much computation can be delegated to cloud server. In this reason, we can say our analysis is valid.



DOI: 10.1002/cpe

Table III. Computing time of operations (msec)

| | |
|---|---|
| Pairing ($t_p$) | 9.92 |
| Exponentiation ($t_e$) | 4.66 |
| Multiplication ($t_m$) | 1 |
| Hash ($t_h$) | 4.64 |

Table III represents computing time for major four operations that is ingredient of CPRE. Table IV represents calculating results based on Tables II and III. Compared with E-CPRE, our scheme can reduce 17.7% of re-encryption key generation process, 64.1% of decryption process, and 52.7% of condition value changing process in user side by delegating partial computation to the cloud computing. The computational overhead of re-encryption key generation process in proposed scheme seems similar to CPRE scheme. Since a user can reuse the part of re-encryption key which is used to complete a re-encryption key, a user has higher efficiency for generating more re-encryption keys in our scheme.

Table IV. Numerical result (msec)

| | CPRE [8] | Efficient CPRE [10] | Proposed Scheme | Delegation (%) |
|---|---|---|---|---|
| Re-encryption key generation | 25.30 | 20.63 | 16.99 | 17.64 |
| Decryption | 31.54 | 31.54 | 11.31 | 64.14 |
| Changing a condition value | 46.51 | 29.54 | 13.97 | 52.71 |

Fig. 2 shows overall simulation results in re-encryption key generation, decryption, and changing a condition value aspects. The left side in the Fig. 2 represents running time (sec) and bottom side represents the number of times a function executed. Fig. 2 (a) shows comparison in point of re-encryption key generation among CPRE schemes.

Our scheme can delegate approximately 92.5% of computation for re-encryption key generation in case of user generated 50 re-encryption keys. This result is quite different with numerical result in Table IV. Since a user can reuse the part of re-encryption key which is used to complete a re-encryption key, a user has higher efficiency for generating more re-encryption keys in our scheme. Additionally, originator receives cloud computed partial re-encryption key instead of public key which is used to generate re-encryption key. Therefore, overall communication overhead is similar as previous CPRE schemes.

Fig. 2 (b) shows comparison of decryption among CPRE schemes. Using partial decryption function of our scheme, we can delegate approximately 64.1% of computation for decryption. Fig. 2 (c) shows comparison of changing condition values. Our scheme can delegate approximately 52.7% of computation by CCK. These delegation means less computation is needed for sharing data using CPRE for mobile users.

In terms of communication overhead, compared with previous CPRE schemes, our scheme need to transfer little more data to the outsourcing server. An originator needs to send data length of $\{0,1\}^k$, which represents as communication overhead of $DK$, for a request of partial re-encryption key generation process. Note that the originator in previous CPRE needs to receive public keys of users in sharing group to generates re-encryption key, and our scheme receives a set of $RK_{part}$s instead of public keys. Hence, only sending $DK$ is treated as communication overhead. Since the $DK$ is irrelevant to number of re-encryption key, the communication overhead a negligible especially in case if number of users in a sharing group is large. Similar as the re-encryption key generation, the originator needs to send only one data which length is $\{0,1\}^k$ for the condition value changing process and the partial decryption process. Its computational overhead is negligible in case if a large number of ciphertexts need to be transformed and decrypted.



DOI: 10.1002/cpe

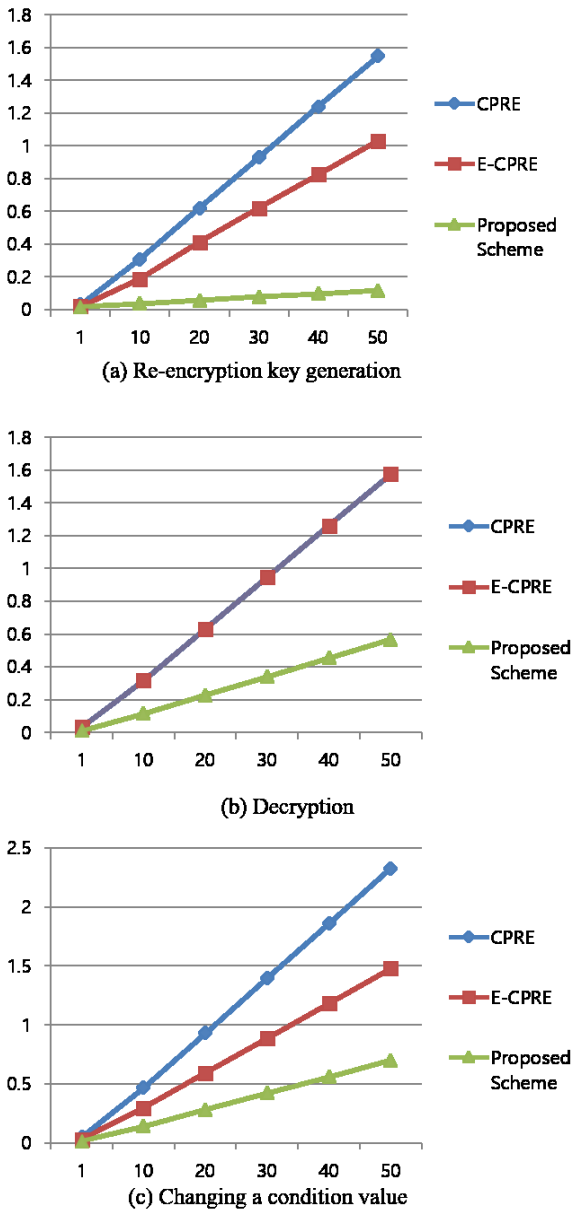


Figure 2. Simulation results

## 8. CONCLUSION

In this paper, we proposed the outsourcing CPRE scheme which can delegate heavy computation securely during data sharing for mobile cloud environment. The proposed scheme introduced outsourcing server which performs following three kinds of delegated operations: (a) partial re-encryption key generation, (b) partial decryption, (c) and condition value changing procedure with CCK. By delegating these partial computation of CPRE, our scheme can significantly reduce burden of a user. In addition, our scheme provides checkability to detect dishonest action of service provider.

The simulation results show that our scheme can delegate 100% of re-encryption overhead, 17.7% of re-encryption key generation overhead, 52.7% of condition value changing overhead,



DOI: 10.1002/cpe

and 64.1% of decryption processes overhead while the state-of-art CPRE can delegate 100% of re-encryption overhead only. This implies that CPRE-M outperforms its alternatives as (a) the size of group is getting larger, (b) the membership of the group is more dynamic, (c) data is more frequently exchanged among members. As a result, compared with previous CPREs, CPRE-M is an efficient solution for data sharing in MCC which uses resource-constraint and battery-powered mobile devices.

## REFERENCES


1. Liu F, Shu P, Jin H, Ding L, Yu J, Niu D, Li B. Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications. *IEEE Wireless Communications* 2013; **20**(3):14–22, doi:10.1109/MWC.2013.6549279.
2. Suo H, Liu Z, Wan J, Zhou K. Security and privacy in mobile cloud computing. *Proceedings of the 9th International Wireless Communications and Mobile Computing Conference (IWCMC)* 2013; :655–659doi:10.1109/IWCMC.2013.6583635.
3. Ragini, Mehrotra P, Venkatesan S. An efficient model for privacy and security in mobile cloud computing. *Proceedings of the International Conference on Recent Trends in Information Technology (ICRTIT)* 2014; :1–6doi:10.1109/ICRTIT.2014.6996177.
4. Gong Y, Zhang C, Fang Y, Sun J. Protecting location privacy for task allocation in ad hoc mobile cloud computing. *IEEE Transactions on Emerging Topics in Computing* 2015; **published online**:1–12, doi:10.1109/ICRTIT.2014.6996177.
5. Xie L, Mantysalo M, Zhou X, Pang Z, Xu L, Kao-Walter S, Chen Q, Zheng L. A health-iot platform based on the integration of intelligent packaging, unobtrusive bio-sensor, and intelligent medicine box. *IEEE Transactions on Industrial Informatics* 2014; **10**(4):2180–2191, doi:10.1109/TII.2014.2307795.
6. He W, Yan G, Xu L. Developing vehicular data cloud services in the iot environment. *IEEE Transactions on Industrial Informatics* 2014; **10**(2):1587–1595, doi:10.1109/TII.2014.2299233.
7. Zanella A, Bui N, Castellani A, Vangellsta L, Zorzi M. Internet of things for smart cities. *IEEE Internet of Things Journal* 2014; **1**(1):22–32, doi:10.1109/JIOT.2014.2306328.
8. Weng J, Yang Y, Deng QTR, Bao F. Efficient conditional proxy re-encryption with chosen-ciphertext security. *Proceedings of the 12th Information Security Conference (ISC)* 2009; :151–166doi:10.1007/978-3-642-04474-8_13.
9. Chu C, Weng J, Chow S, Zhou J, Deng R. Conditional proxy broadcast re-encryption. *Proceedings of the 14th Australasian Conference on Information Security and Privacy (ACISP)* 2009; :327–342doi:10.1007/978-3-642-02620-1_23.
10. Son J, Kim H, Kim D, Oh H. On secure data sharing in cloud environment. *Proceedings of International Conference on Ubiquitous Information Management and Communication (IMCOM)* 2014; doi:10.1145/2557977.2558068.
11. Son J, Kim D, Hussain R, Oh H. Conditional proxy re-encryption for secure big data group sharing in cloud environment. *Proceedings of the 2nd International Workshop on Security and Privacy in Big Data (BigSecurity 2014) in conjunction with INFOCOM 2014* 2014; :547–552doi:10.1109/INFCOMW.2014.6849289.
12. Mambo M, Okamoto E. Proxy cryptosystems: delegation of the power to decrypt ciphertext. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences (TFECCS)* 1997; **E80-A**(1):54–63.
13. Blaze M, Beumer G, Strauss M. Divertible protocols and atomic proxy cryptography. *Proceeding of International Conference on the Theory and Application of Cryptographic Techniques (Eurocrypt)* 1998; :127–144doi:10.1007/BFb0054122.
14. Ateniese G, Fu K, Green M, Hohenberger S. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)* 2006; **9**(1):1–30, doi:10.1145/1127345.1127346.
15. Ateniese G, Benson K, Hohenberger S. Key-private proxy re-encryption. *Proceeings of the RSA Conference on Topics in Cryptology (CT-RSA '09), Lecture Notes in Computer Science* 2009; **5473**:279–294, doi:10.1007/978-3-642-00862-7_19.
16. Green M, Ateniese G. Identity-based proxy re-encryption. *Proceedings of the 5th International Conference on Applied Cryptography and Network Security (ACNS '07), Lecture Notes in Computer Science* 2007; **4521**:288–306, doi:10.1007/978-3-540-72738-5_19.
17. Fang L, Susilo W, Ge C, Wang J. Interactive conditional proxy re-encryption with fine grain policy. *The Journal of Systems and Software* 2011; **84**(12):2293–2302, doi:10.1016/j.jss.2011.06.045.
18. Shao J, Wei G, Ling Y, Xie M. Identity-based conditional proxy re-encryption. *Proceedings of the IEEE International Conference on Communications (ICC)* 2011; :1–5doi:10.1109/icc.2011.5962419.
19. Fang L, Susilo W, Ge C, Wang J. Hierarchical conditional proxy re-encryption. *Computer Standards & Interfaces* 2012; **34**(4):380–389, doi:10.1016/j.csi.2012.01.002.
20. Green M, Hohenberger S, Waters B. Outsourcing the decryption of abe ciphertexts. *Proceedings of the 20th Conferences of USENIX* 2011; :34.
21. Zhou Z, Huang D. Efficient and secure data storage operations for mobile cloud computing. *Proceedings of the 8th International Conference on Network and Service Management* 2011; :37–45.
22. Li J, Jia C, Li J, Chen X. Outsourcing encryption of attribute-based encryption with mapreduce. *Proceedings of International Conference on Information and Communications Security* 2012; :191–201doi:10.1007/978-3-642-34129-8_17.
23. Li J, Huang X, Li J, Chen X, Xiang Y. Securely outsourcing attribute-based encryption with checkability. *IEEE Transactions on Parallel and Distributed Systems* 2014; **25**(8):2201–2210, doi:10.1109/TPDS.2013.271.

24. Boneh D, Lynn B, Shacham H. Short signatures from the weil pairing. *Journal of Cryptology* 2004; **17**(4):297–319, doi:10.1007/s00145-004-0314-9.
25. Deng R, Weng J, Liu S, Chen K. Chosen-ciphertext secure proxy re-encryption without pairings. *Proceedings of the 7th International Conference on Cryptology and Network Security (CANS)* 2008; :1–17doi:10.1007/978-3-540-89641-8_1.
26. Hohenberger S, Lysyanskay A. How to securely outsource cryptographic computations. *Proceedings of Theory of Cryptography, Lecture Notes in Computer Science* 2005; **3378**:264–282, doi:10.1007/978-3-540-30576-7_15.
27. Canetti R, Riva B, Rothblum G. Two protocols for delegation of computation. *Proceedings of Information Theoretic Security, Lecture Notes in Computer Science* 2012; **7412**:37–61, doi:10.1007/978-3-642-32284-6_3.
28. Atallah M, Frikken K. Securely outsourcing linear algebra computation. *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS '10)* 2010; (48–59), doi:10.1145/1755688.1755695.
29. Canetti R, Riva B, Rothblum G. Practical delegation of computation using multiple servers. *Proceedings of the 18th ACM conference on Computer and communications security (CCS '11)* 2011; :445–454doi:10.1145/2046707.2046759.
30. Li H, Cheng C, Pang L. A new $(t, n)$ threshold multi-secret sharing scheme. *Proceedings of International conference on Computational Intelligence and Security, Lecture Notes in Computer Science* 2005; **3802**:421–426, doi:10.1007/11596981_61.
31. Elgamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* 1985; **31**(4):469–472, doi:10.1109/TIT.1985.1057074.
32. Goldreich O. The foundations of cryptography, vol. 1, basic tools. *Cambridge University Press* 2001; .