International Journal of Innovative Computing, Information and Control
Volume **16**, Number **3**, June **2020**

ICIC International ©2020 ISSN 1349-4198
pp. 867–877

RESEARCH ON NODE AUTHORIZATION TRUSTED UPDATE MECHANISM BASED ON AGENT RE-ENCRYPTION IN IOT CLOUD

Shunhui Li, Qiubi Sun and Haidong Wu*

School of Economics and Management
Fuzhou University
No. 2, Xueyuan Road, University Town, Fuzhou 350108, P. R. China
sunfast@foxmail.com; sunqiubi@126.com; *Corresponding author: haliyo2121@163.com

Received November 2019; revised March 2020

Abstract. *This paper describes a trusted update mechanism of node authorization based on proxy re-encrypted in IoT cloud. It defines that the system consists of IoT data server, node authorization management server, re-encrypted proxy server and working node. The data is generated by trusted IoT data server through wireless sensor network. The network publishes to the node, which requires the node to register in the node authorization management server. The registered node will obtain the ciphertext data in the re-encrypted proxy server in the cloud, and obtain the decryption parameters in the node authorization management server for data decryption. This paper designs a secure and efficient cloud data sharing protocol, which satisfies the security of adaptive selection ciphertext, ensures the security of data storage and sharing, and also has the characteristics of unidirectionality, universality, non-interaction, non-transferability and anti-collusion attack. This paper also analyzes the computing and communication costs of data owners, cloud service provider (CSP) and data receivers under the data sharing protocol. The results show that in the cloud data sharing protocol based on this scheme, the data owner's one-time communication cost is 2368 bits, which is better than several comparison schemes and can meet the practical application requirements.*

Keywords: Node authorization trusted update mechanism, Internet of Things (IoT), Cloud computing, Sharing protocol, Communication overhead, Cloud service provider (CSP)

1. **Introduction.**

Cloud computing, as a new service mode, can easily provide computing, storage and data sharing services for remote users. For example, government agencies allow their members to upload and share files in the public cloud, that is, members of the organization can access data uploaded and shared by other members without local storage and maintenance of shared files [1-3]. In addition, the shared data in the cloud can be accessed by members of the government agency through the Internet at any time and anywhere. Despite the benefits of cloud data sharing, once data is uploaded to cloud service provider (CSP), data owners will lose control over the data [4,5]. Since CSP is usually semi-trusted and outsourced data containing sensitive information can only be accessed by authorized users, this will inevitably lead to many security challenges. Therefore, for security reasons, data owners need to encrypt sensitive information before uploading outsourced data.

The combination of the IoT and cloud computing promotes the penetration of the cloud in people's lives, and also improves the communication and data processing capabilities of the IoT nodes. The IoT cloud platform is mainly aimed at data sharing and dissemination application scenarios. However, the combination of the two also creates more security

DOI: 10.24507/ijicic.16.03.867

issues. First of all, the access of IoT nodes has caused a surge in data in the cloud, including more and more sensitive data and private information. Taking smart homes as an example, the data obtained by home cameras is a true record of the user's daily private life, related to the privacy protection of users' lives. Regarding how the data collected by the nodes are transmitted to the cloud, how they are acquired by legitimate nodes, and how to prevent illegal theft and mining of service providers are important; again, the computing capabilities and anti-attack capabilities of various sensor nodes are far less than computer terminal equipment. It is easy to be destroyed and controlled by attackers. When a legitimate node is attacked or destroyed, how to ensure that the node can no longer obtain data in the cloud is also an urgent problem to be solved.

Aiming at the security problems arising from the combination of IoT and cloud computing, this paper introduces the idea of proxy re-encrypted (PRE) into the update and management of access authorization of IoT cloud platform nodes, and proposes a trusted update mechanism of IoT cloud node authorization based on agent re-encryption. The main contributions of this mechanism are as follows.

1) The mechanism makes full use of the role of cloud platform in data distribution of the IoT. The deployment and cloud-side re-encrypted proxy servers generate a large number of node ciphertext data. At the same time, the above ciphertext generation is based on the initial ciphertext generated by the data server, which guarantees the confidentiality and privacy of data.

2) The mechanism revokes the decryption parameters of hijacked nodes by the authorization management server, and the re-encrypted ciphertext in the cloud will not be decrypted, which ensures the credibility of the authorization revocation and does not require repeated encryption by the user server.

The whole article is arranged as follows: Chapter 2 introduces the current related research; Chapter 3 describes the proxy re-encryption system in detail; Chapter 4 presents the proposed trusted update mechanism of node authorization for proxy re-encryption; Chapter 5 presents the performance evaluation experiment; Chapter 6 summarizes the proposed method and points out the future research direction.

2. Relevant Research.

Generally, symmetric encryption algorithm can effectively guarantee the security of outsourced data in cloud environment, and there is no cumbersome key management problem. However, in order to realize the secure sharing of outsourced data in cloud environment, it is not advisable for the data owner to inform the data receiver directly of the symmetric encryption key, and it is impossible for the data owner to inform. The simplest and safest method is that the data owner downloads the cloud data and decrypts it locally, then encrypts the decrypted message with the public key of the data receiver and sends it to the data receiver. At this time, the data receiver can obtain the shared data by using his own private key. Obviously, this method sacrifices the data owner's computing cost, communication bandwidth and local storage resources, which is not in line with the user's original intention of saving resource overhead through cloud computing. Another idea is that the data owner defines the access policy in advance and encrypts the shared data using the attribute-based encryption algorithm. Only the authenticated users whose attributes satisfy the access policy can access and decrypt the shared data. However, once data access policies are frequently updated, data owners still have to sacrifice a lot of computing resources to complete shared data downloading, decryption and re-encryption. Therefore, the traditional public key cryptography scheme cannot solve the problem of data security sharing in cloud storage. [6] proposes the first chosen-ciphertext attack (CCA) secure one-way broadcast proxy re-encryption (BPRE), which satisfies adaptive chosen ciphertext security under the standard model. [7] proposes a key selection security model, which allows adversaries to choose public keys for malicious users. Based on this model, they propose an efficient proxy re-encryption scheme without random oracle model and CCA security. [8] introduces symmetric password and message authentication code into the access authentication of the nodes of the IoT to ensure data. In [9], the user's key is divided into two parts: deception key and private key. The decryption key of the user is generated by two parts of operation. The key revocation is realized by updating the deception key.

[10] proposes a layered access control scheme for the perception layer, which divides the perception nodes into layers according to the security level and sets partial order management access control mechanism for the nodes of different layers. [11] constructs a one-way and multi-purpose PRE scheme. However, the distribution and management of the two parts of key are not discussed, and the deterministic replacement of key data is not explained in essence. In addition, the deterministic update schemes of traditional data often require recalculation of ciphertext data. Due to the characteristics of cloud platform, this work needs to be delivered to a trusted IoT database, which will consume a large amount of resources of users and lead to cloud servers staying at the storage level only.

3. **Proxy Re-Encryption System.**

Proxy re-encryption system is a new type of public key encryption system with the function of ciphertext security conversion. In the proxy re-encryption system, a semi-trusted agent (Proxy) plays the role of ciphertext conversion, which can convert the ciphertext encrypted by the Delegator's public key into the ciphertext encrypted by the Delegatee's public key, and then the client can use his own private key to decrypt the converted ciphertext. In the process of ciphertext conversion, the agent must have a ciphertext conversion key (re-encrypted key) authorized by the principal for the principal, and the agent cannot obtain any information about the plaintext. Proxy re-encryption has become a research hotspot in the field of cryptography and information security, and has accumulated a lot of research results. This chapter will briefly describe the formal definition, security model and characteristics of PRE.

3.1. **Formal definition of PRE.**

A PRE scheme consists of algorithms KeyGen, ReKey, Encrypt, ReEncrypt and Decrypt:

1) KeyGen $\left(1^k\right) \to (pk_i, sk_i)$: Enter the security parameter 1^k, $k \in \kappa$, and the key generation algorithm KeyGen outputs a pair of public/private keys (pk_i, sk_i) for user i.

2) ReKey$(pk_A, sk_A, pk_B, sk_B) \to rk_{A\to B}$: Input Delegator's public/private key pair (pk_A, sk_A) and Delegatee's public/private key to (pk_B, sk_B) (Here sk_B is optional), proxy re-encryption key generation algorithm ReKey outputs a proxy re-encryption key $rk_{A\to B}$. Note: At this time, Delegator is the principal and Delegatee is the principal.

3) Encrypt$(pk_i, m) \to c_i$: Input user i's public key pk_i and message $m \in M$, and output message m's ciphertext $c_i \in C_1$ by encryption algorithm.

4) ReEncrypt$(rk_{A\to B}, c_A) \to c_B$: Enter a proxy re-encrypted key $rk_{A\to B}$ and the ciphertext c_A of Delegator, and the proxy re-encrypted algorithm ReEncrypt outputs the re-encrypted ciphertext $c_B \in C_2$ for Delegatee.

5) Decrypt$(sk_i, c_i) \to m$: Input user i's private key sk_i and ciphertext c_i, decryption algorithm Decrypt output message m or error symbols indicate that ciphertext c_i is illegal.

Correctness: For message $m \in M$ and any two public/private keys (pk_A, sk_A), (pk_B, sk_B) $\leftarrow$ KeyGen $\left(1^k\right)$ in any plaintext space, the above correctness must satisfy the following two conditions.

Decrypt$(sk_A,$ Encrypt$(pk_A, m)) \to m$ Decrypt$(sk_B,$ ReEncrypt(ReKey$(pk_A, sk_A, pk_B, sk_B), c_A)) \to m$

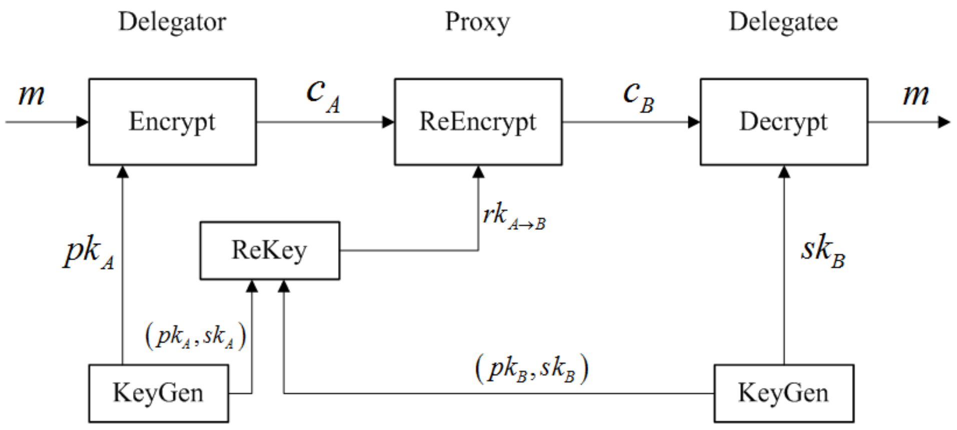

Figure 1. Proxy re-encryption system model

As shown in Figure 1, in the aforementioned proxy re-encryption definition, a semi-trusted proxy with proxy re-encryption key $rk_{A\to B}$ can re-encrypt the ciphertext $c_A \in C_1$ under the Delegator public key to the ciphertext $c_B \in C_2$ for the same plaintext $m \in M$ under the Delegatee public key. Delegatee can then decrypt and retrieve the message, while the agent is unable to obtain any information (e.g., sk_A, sk_B and m).

pk_A in the ReKey algorithm defined by PRE above, the private key of Delegatee is optional. Generally, when Delegatee's private key sk_B does not participate in the generation of the proxy re-encryption key, the proxy re-encryption scheme has unidirectional and non-interactive nature; on the contrary, the ReKey algorithm outputs a two-way proxy re-encryption key, and the PRE scheme is interactive. In addition, Encrypt algorithm and ReEncrypt algorithm output ciphertext space C_1 and C_2, respectively. When $C_1 = C_2$, Encrypt algorithm and ReEncrypt algorithm output have the same ciphertext space, only one decryption algorithm Decrypt can decrypt the ciphertext output of the above two algorithms at the same time; when $C_1 \neq C_2$, for Encrypt and ReEncrypt algorithm, two different Decrypt algorithms are required.

3.2. **Characteristics of PRE.** The following nine important features of PRE will be introduced according to the definition of proxy re-encryption mentioned above (It is easier to understand with Figure 2).

1) **Unidirectional**: In the one-way PRE scheme, the proxy re-encryption key is one-way, that is, the proxy can use the one-way proxy re-encryption key $rk_{A\to B}$ to convert Delegator's ciphertext to Delegatee's ciphertext, but not Delegatee's ciphertext to Delegator's ciphertext; on the contrary, bidirectional PRE not only allows the proxy to convert Delegator's ciphertext to Delegator's ciphertext. Text is converted to Delegatee's ciphertext, and vice versa [12].

2) **Multi-use**: In a multiplexing PRE scheme, the output of Encrypt algorithm and ReEncrypt algorithm can be used as the input of ReEncrypt algorithm again; on the contrary, single-use PRE only allows the output of Encrypt algorithm as the input of ReEncrypt algorithm.

3) **Private-proxy**: In a secret proxy re-encryption, the proxy is honest and can ensure the privacy of the proxy re-encryption key, that is, the attacker cannot obtain the proxy re-encryption key from the ciphertext conversion process; conversely, in public-proxy PRE, the attacker can calculate the proxy re-encryption key by observing the input and output of the proxy.

4) **Transparent**: In a transparent PRE scheme, the agent is transparent, that is, the ciphertext output by Encrypt algorithm is computationally indistinguishable from the ciphertext output by ReEncrypt algorithm.

5) **Key-optimal**: In the PRE scheme of key optimization, no matter how many delegators or delegatees exist, users only need to protect and store a small amount of secret data.

6) **Non-interactive**: In the non-interactive PRE scheme, the proxy re-encryption key is generated by the principal's public/private key pair and the principal's public key, that is, the principal does not participate in the authorization process of the proxy re-encryption key.

7) **Non-transitive**: In the non-transitive PRE scheme, the proxy re-encryption key is non-transitive. Given the proxy re-encryption key $rk_{A\to B}$ and $B \to C$ of $A \to B$, the proxy re-encryption key $rk_{B\to C}$ of $A \to C$ cannot be calculated by the proxy re-encryption key $rk_{A\to C}$.

8) **Temporary**: In the temporary PRE scheme, the proxy re-encryption key is revocable, that is, the principal has the right to withdraw the decryption authorization entrusted by the principal.

9) **Collusion-resistant**: In the PRE scheme against collusion attack, the proxy re-encrypted key can resist collusion attack, that is, when the principal colludes with the proxy, they cannot reveal the principal's private key and the plaintext information.

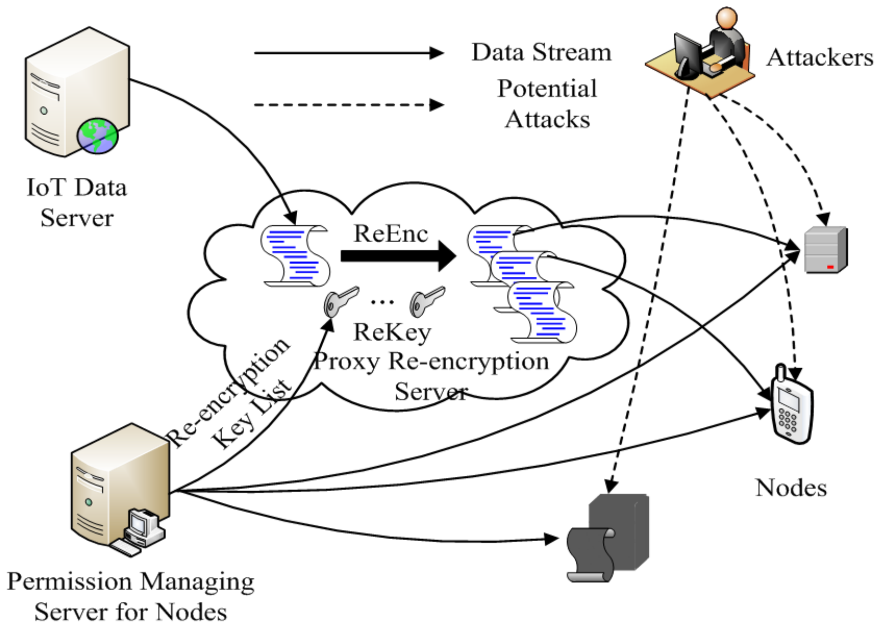

FIGURE 2. The schematic diagram of PRE characteristics

4. Trusted Update Mechanism of Node Authorization for Proxy Re-Encrypted.

4.1. Definition of authorized trusted update mechanism for cloud data nodes.

Consider the following scenario: In a cloud computing environment, Delegator wants to share outsourced sensitive data with data receiver Delegatee. No one, except Delegatee, can decrypt shared data, including CSP. This section will construct a more secure and efficient cloud data sharing protocol based on the idea of CL-PRE. As shown in Figure 3, the cloud data sharing model includes four types of participants: data owner, CSP, data

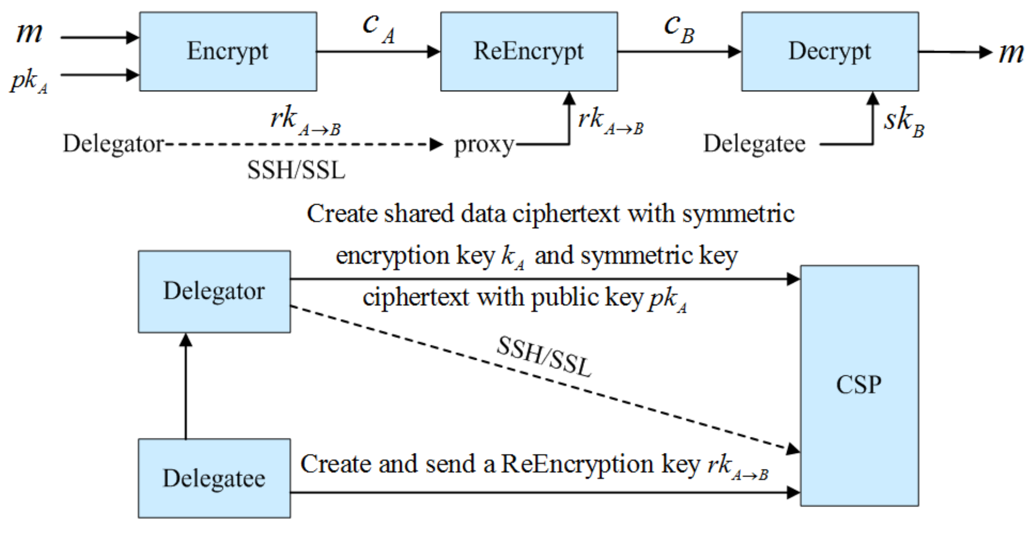
m
pk_A
Encrypt
c_A
ReEncrypt
c_B
Decrypt
m
$rk_{A\to B}$
Delegator
SSH/SSL
proxy
$rk_{A\to B}$
Delegatee
sk_B
Create shared data ciphertext with symmetric encryption key k_A and symmetric key ciphertext with public key pk_A
Delegator
SSH/SSL
CSP
Delegatee
Create and send a ReEncryption key $rk_{A\to B}$

FIGURE 3. Model of authorized trusted update mechanism for cloud data nodes

receiver and key generation center (KGC) [13,14]. Data owners create encrypted data and upload ciphertext data to semi-trusted CSP; CSP manages and maintains multiple cloud servers to provide data storage services and access services for users; data receivers are authorized data receivers and have the right to obtain shared data outsourced by data owners to CSP; KGC is the key. Generation center is responsible for distributing key for users in cloud environment.

Because the computational efficiency of public key encryption algorithm is lower than that of symmetric encryption algorithm, it is not suitable for encrypting large data files. In the cloud data sharing protocol designed in this paper, PRE scheme is not directly used to encrypt the outsourced data of data owners, but to protect the confidentiality of user data using symmetric encryption algorithm; otherwise it will make the protocol very inefficient. Therefore, this paper uses PRE to deal with the symmetric key of symmetric encryption algorithm used in the protocol [15-17]. If there is an authorized data receiver who wants to access the shared data of the data owner, the data receiver needs to first decrypt the symmetric key with its own private key, and then use the obtained symmetric key to decrypt the shared data [18]. As mentioned earlier, assuming CSP is semi-trustworthy, meaning that CSP can faithfully execute data sharing protocols without actively damaging data confidentiality, but is curious about data cipher data stored by data owners or with data reception [13,16]. The collusion initiated an attack against the data owner.

4.2. **Algorithm implementation process.** The implementation of authorized trusted update mechanism for cloud data nodes mainly involves seven algorithmic functions.

Establishment of parameters:

$Setup(q) \to param$. Choose prime number p, group G_1, G_2 as multiplicative cyclic group, g as generator of G_1, Hash function group H_1, H_2, H_3, H_4, where H_1: $\{0,1\}^* \to G_1$, H_2: $\{0,1\}^* \to Z_p^*$, H_3: $G_2 \to \{0,1\}^l$, H_4: $\{0,1\}^* \to G_1$, open parameter $param = \{p, G_1, G_2, g, H_i(i = 1, 2, 3, 4)\}$. Define bilinear mapping e: $G_1 \times G_2 \to G_2$.

Initial key generation:

$\mathrm{KeyGen}(param) \to (sk_i, pk_i)$. Select $x_i \in Z_p^*$, then $sk_i = x_i$, $pk_i = g^{x_i}$. The node submits the identity information to the authorization management server, which can be online or offline. If it is online, the integrity and confidentiality of the information transfer need to be considered, which is beyond the scope of this article, so it is not described here. After the node is registered, the node authorization management server calls the algorithm KeyGen to generate a public and private key pair for it, and stores its public key to the authorization management server, and calls the ReKeyPara algorithm to generate the re-encryption key generation/decryption parameters of the registered node, and add it to the list.

Initial encryption:

InitEnc$(k, pk_{IoT}) \rightarrow E(k)_{IoT}$. The IoT data server generates the message M to be distributed, and symmetric encryption is performed on M. The symmetric encryption key K is called by the algorithm InitEnc to generate the ciphertext $E(k)$, and transmits it to the re-encryption proxy server.

IoT data server encrypts symmetric key K with its own public key pk_{IoT}, selects $i \in G_2$ and calculates $r = H_2(k\|i)$, then

$$\begin{aligned}
&E(k)_{IoT} = (c_1, c_2, c_3, c_4, c_5)\\
&c_1 = g^r\\
&c_2 = i \cdot e(pk_A, H_1(pk_A))^r\\
&c_3 = k \oplus H_3(i)\\
&c_4 = H_1(pk_A)\\
&c_5 = H_4(c_1\|c_2\|c_3\|c_4)
\end{aligned}$$

Re-Encryption parametric generation:

ReKeyPara$(pk_i, pk_{IoT}, sk_{IoT}, param) \rightarrow (\eta_i, \varphi_i)$. Authorization management server generates unsigned random string φ and calculates $\eta = \left(pk_i, H(pk_{IoT})^{sk_{IoT}}, r\right)$.

Proxy re-encrypted key generation:

ReKeyGen$(\eta_i, \varphi_i) \rightarrow rk_{IoTi}i$. Generate a proxy re-encrypted key $rk_{IoTi} = \big(pk_i, pk_i^r, H_1(pk_i\|\varphi_i) \cdot H_1(pk_{IoT})^{sk_{IoT}}, g^{-r}\big)$ for node i.

Proxy re-encryption:

ReEnc$(E(K)_{IoT}, rk_{IoTi}) \rightarrow E(K)_{IoTi}$. Re-encrypted proxy server to initial ciphertext $E(K)_{IoT}$.

Authorization management server updates the corresponding entry in the parameter list, the authorization will need to revoke the corresponding parameter, delete.

After re-encrypting, the private key sk_i can be generated by node i: decrypted ciphertext $E(K)_{IoTi} = (c_1', c_2', c_3', c_4', c_5')$.

If $e(c_1, H_4(c_1\|c_2\|c_3\|c_4)) = e(g, c_5)$ calculates as follows, otherwise the feedback information integrity is wrong.

$$\begin{aligned}
c_1' &= c_1\\
c_2' &= c_2 \cdot e\left(pk_i^r g^{-r}, H_1(pk_{IoT})^{-sk_{IoT}}\right) \cdot e\left(pk_i^r, H_1(pk_i\|\varphi_i.) \cdot H_1(pk_{IoT})^{-sk_{IoT}}\right)\\
&= i \cdot e\left(pk_i^r, H_1(pk_i\|\varphi_i)\right)\\
c_3' &= c_3\\
c_4' &= H_1(pk_i)\\
c_5' &= H_4\left(c_1'\|c_2'\|c_3'\|c_4'\right)^r
\end{aligned}$$

Decryption:

ReDec$(E(K)_{IoTi}, sk_i, \varphi_i) \rightarrow K$. Node i decrypts and re-encrypts the symmetric key ciphertext and obtains the symmetric key K. If $e(c_1', H_4(c_1'\|c_2'\|c_3'\|c_4')) = e(g, c_5')$ calculates as follows, otherwise the feedback information integrity is wrong:

Calculate $i = c_2'/e(c_1', H_1(pk_i\|\varphi_i))^{sk_i}$;

Calculate $K = c_3' \oplus H_3(i)$;

Calculate $r = H_2(K\|i)$, if $c_1' = g^{-r}$ and $c_2' = i \cdot e(pk_i, H_1(pk_i\|\varphi_i))^r$, output K.

5. Performance Evaluation.

5.1. Experiment setup.

The proposed schemes are compared with those in [7-9]. The execution time of the four schemes mentioned above will be evaluated by Miracl cryptographic library function. Miracl is a cryptographic algorithm library based on C language. Generally, in terms of computational overhead, some computational operations such as multiplication or addition on a group, conventional hash functions and exclusive or operations are neglected. Therefore, this paper only considers the number of "large integer" operations that each certificateless proxy re-encryption scheme needs to perform. This section evaluates the performance of data owners, CSPs, and data receivers in the cloud data sharing protocols mentioned in this article. The experimental hardware environment is shown in Table 1.

Table 1. Experimental environment

CPU	Memory	Operating system
Inter(R) Core(TM) 2 Quad 2.40GHz	3GB RAM DDR2	Windows10

In data sharing protocols, data owners need to implement algorithms including SetSecretValue, SetPrivateKey, SetPublicKey, ReEncryptKey, Encrypt and Decrypt2. In this section, only three algorithms, ReEncryptKey, Encrypt and Decrypt2, are considered to evaluate the computing overhead of data owners. Data owner implements Encrypt algorithm to encrypt symmetric key, Decrypt2 algorithm to decrypt symmetric key ciphertext, and ReEncryptKey algorithm to calculate a re-encryption authorization for each data receiver.

5.2. Experimental results and analysis.

Figure 4 depicts the computational overhead of encryption and decryption for data owners. The horizontal axis represents the number of executions of encryption and decryption algorithms, and the vertical axis represents the computational time of encryption and decryption algorithms. In Figure 4, the encryption and decryption efficiency of the data owner is better than that of the cloud data sharing protocol based on several contrast schemes under the cloud data sharing protocol in this paper. Compared with the cloud data sharing protocol, the computing cost of decryption is higher than that of encryption. In addition, Figure 4 does not reflect the computational overhead required by the data owner to execute the ReEncryptKey algorithm, because the time of the ReEncryptKey algorithm in this scheme is 2.19 times as long as that in the comparison scheme. Therefore, the cloud data sharing protocol based on this scheme is better than others in terms of data owner computing overhead.

In the cloud data sharing protocol based on PRE, CSP only needs to execute ReEncrypt algorithm and re-encrypt the shared data of the data owner with the re-encrypted key. Figure 5 depicts the CSP computing overhead of the proposed scheme under cloud data sharing protocol compared with that of [7-9]. The horizontal axis represents the number of data receivers, and the vertical axis represents the computing overhead of CSP. With the increasing number of data owners, the computational overhead of CSP for re-encrypting all data receivers increases linearly. The performance of the ReEncrypt algorithm of this scheme is obviously better than that of the contrast scheme. Therefore, in the cloud data sharing environment based on this scheme, the computational overhead of CSP is less.

In the cloud data sharing environment, the data receiver decrypts the re-encrypted shared data using Decrypt1 algorithm. By analyzing the scheme and mechanism comparison scheme proposed in this paper, we can see that Decrypt1 algorithm of proxy

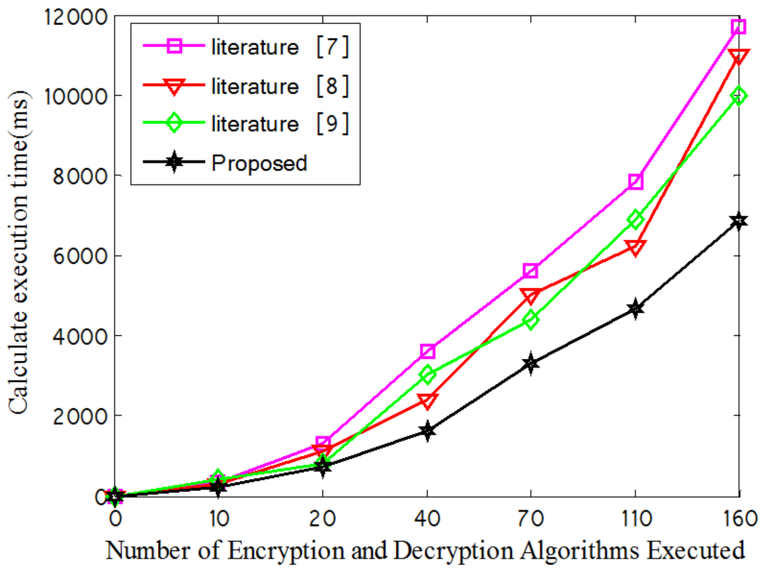

FIGURE 4. Computational overhead comparison of encryption/decryption for data owners

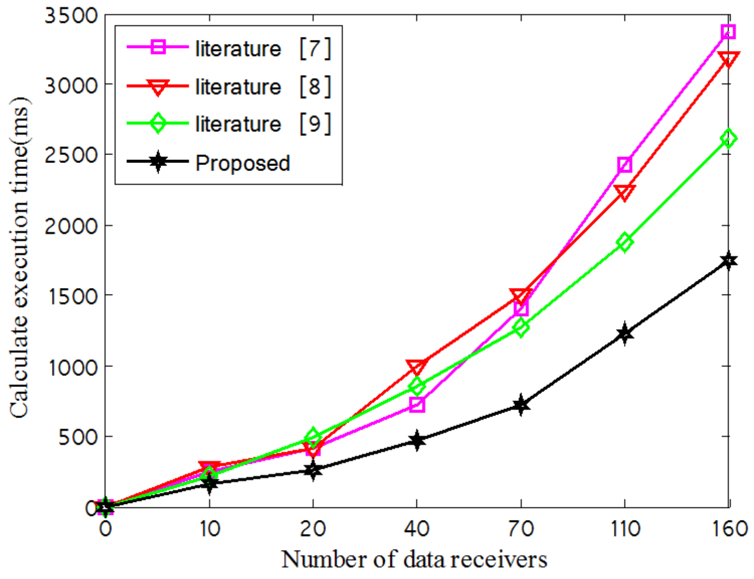

FIGURE 5. Computing overhead comparison of CSP

re-encryption scheme has the same computing overhead, which is 4.29 times the computing time of a basic power operation. Therefore, the cloud data sharing protocol based on this scheme has the same advantages and disadvantages as the cloud data sharing protocol based on the comparison scheme in terms of computing overhead of data receivers.

In cloud data sharing environment, data owners need to communicate with CSP, including shared data ciphertext, symmetric key ciphertext and re-encrypted key. When calculating the communication overhead of data owners, the communication overhead of shared data ciphertext will not be considered. The second layer ciphertext length of this scheme is 2208 bits, the length of re-encryption key is 160 bits, while the scheme of [7] is 3812 bits and 160 bits, the scheme of [8] is 3422 bits and 160 bits, and the scheme of [9] is 3392 bits and 160 bits, respectively. Therefore, in the cloud data sharing protocol based on this scheme, the data owner's one-time communication overhead is 2368 bits, while

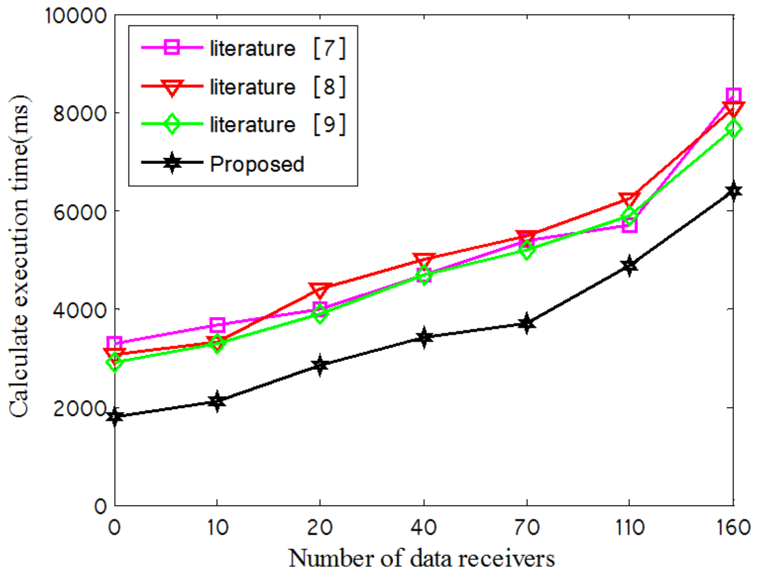

FIGURE 6. Computing overhead comparison of CSP

[7-9] schemes are 3962, 3582 and 3556 bits, respectively. Note that the data owner only uploads symmetric key ciphertext to CSP once, and for each data receiver, the data owner needs to send a re-encrypted key to CSP. Figure 6 depicts the data owner communication overhead comparison between the proposed scheme and the contrast scheme under cloud data sharing protocol.

The results show that with the increasing number of data owners, the total amount of communication overhead consumed by data owners also increases. Moreover, the communication overhead of the data owner in this scheme is less than that of the comparison scheme. Therefore, the cloud data sharing protocol based on this scheme has more performance advantages in terms of communication overhead for data owners. In addition, the computing overhead of the data owner determines the storage space required by the cloud server managed by CSP. The communication overhead of data receiver comes from downloading shared data ciphertext from CSP-managed cloud server and re-encrypted symmetric key ciphertext. When calculating the communication overhead of data receiver, the shared data ciphertext is also not included. In terms of data receiver communication overhead, there is no significant difference between the cloud data sharing protocol based on this scheme and the cloud data sharing protocol based on the comparison scheme.

In summary, the cloud data sharing protocol based on this scheme is superior to the cloud data sharing protocols based on [7-9], both in terms of computing and communication costs.

6. Conclusion.

Based on the proxy re-encryption scheme, this paper designs a trusted update mechanism of node authorization using proxy re-encryption under the cloud computing platform of the IoT. The protocol satisfies the security of adaptive selection ciphertext, ensures the security of data storage and sharing, and also has unidirectionality, universality, non-interaction, non-transferability and anti-collusion attack. This paper also analyzes the computing and communication costs of data owners, CSP and data receivers under the data sharing protocol.

So far, for proxy re-encryption, this paper argues that there are still several problems worthy of further study: 1) Construct a proxy re-encryption scheme that satisfies the “temporary” requirement. Due to the change of user rights or the existence of privacy such as key leakage, it is inevitable to consider the revocation of proxy re-encrypted key or decryption authorization. At present, all the PRE schemes supporting revocation introduce a valid time slice, that is, decryption authorization is valid only within that time slice. How to revoke the trustee's decryption authorization safely and efficiently will be the focus of future research; 2) Construct a provably secure anti-key leakage proxy re-encryption scheme. In cryptosystem, the problem of key leakage is becoming more and more serious. However, key evolution technology, key isolation technology and intrusion tolerance technology are all effective solutions to the problem of key leakage. Therefore, how to construct a provably secure anti-key leak proxy re-encryption scheme will be the follow-up work of this paper, especially the provably secure key isolation proxy re-encryption scheme.

REFERENCES

[1] M. Al Rawajbeh, Performance evaluation of a computer network in a cloud computing environment, *ICIC Express Letters*, vol.13, no.8, pp.719-727, 2019.

[2] J. Zhang and M. T. Wang, Research on communication scheduling algorithm for smart home in Internet of Things under cloud computing, *Journal of Advanced Computational Intelligent and Intelligent Informatics*, vol.23, no.1, pp.124-128, 2019.

[3] L. Jin and C. H. Yan, Research on key technologies of massive videos management under the background of cloud platform, *Journal of Advanced Computational Intelligent and Intelligent Informatics*, vol.23, no.1, pp.72-77, 2019.

[4] J. Sun, B. Zhu, Q. Jing et al., Confidentiality-preserving publicly verifiable computation, *International Journal of Foundations of Computer Science*, vol.28, no.6, pp.799-818, 2017.

[5] W. Rankothge, F. Le, A. Russo et al., Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms, *IEEE Trans. Network & Service Management*, vol.14, no.2, pp.343-356, 2017.

[6] J. Sun, Y. Hu and L. Y. Zhang, CCA-secure unidirectional proxy broadcast re-encryption in the standard model, *Journal of Computational Information Systems*, vol.8, no.14, pp.5909-5916, 2012.

[7] J. Zhang, Z. Zhang and Y. Chen, PRE: Stronger security notions and efficient construction with non-interactive opening, *Theoretical Computer Science*, vol.542, pp.1-16, 2014.

[8] M. Chiregi and N. J. Navimipour, A comprehensive study of the trust evaluation mechanisms in the cloud computing, *Journal of Service Science Research*, vol.9, no.1, pp.1-30, 2017.

[9] B. Yu, J. Wright, S. Nepal et al., IoTChain: Establishing trust in the Internet of Things ecosystem using blockchain, *IEEE Cloud Computing*, vol.5, no.4, pp.12-23, 2018.

[10] Z. Zhao and S. Lei, Attribute-based access control with dynamic trust in a hybrid cloud computing environment, *International Conference on Cryptography*, 2017.

[11] F. Tang, H. Li and J. Chang, Multi-hop unidirectional proxy re-encryption from multilinear maps, *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, vol.98, no.2, pp.762-766, 2015.

[12] S. Fugkeaw and H. Sato, Achieving scalable and optimized attribute revocation in cloud computing, *IEICE Trans. Information & Systems*, vol.100, no.5, pp.973-983, 2017.

[13] S. H. Kim and I. Y. Lee, Secure multicast using proxy re-encryption in IoT environment, *KSII Trans. Internet & Information Systems*, vol.12, no.2, pp.946-959, 2017.

[14] Y. Chen, Y. Hu, M. Zhu et al., Attribute-based keyword search with proxy re-encryption in the cloud, *IEICE Trans. Communications*, pp.1798-1808, 2018.

[15] M. Su, L. Wang, A. Fu et al., Proxy re-encryption based multi-factor access control for ciphertext in cloud, *Journal of Shanghai Jiaotong University*, vol.23, no.5, pp.76-80, 2018.

[16] Y. Yang and M. Ma, Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds, *IEEE Trans. Information Forensics & Security*, vol.11, no.4, pp.746-759, 2017.

[17] M. A. Ferrag and A. Ahmim, ESSPR: An efficient secure routing scheme based on searchable encryption with vehicle proxy re-encryption for vehicular peer-to-peer social network, *Telecommunication Systems*, vol.66, no.2, pp.1-23, 2017.

[18] H. Hong and Z. Sun, Sharing your privileges securely: A key-insulated attribute based proxy re-encryption scheme for IoT, *World Wide Web*, vol.21, no.4, pp.1-13, 2017.