International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 10 Issue 9 ǁ September 2022 ǁ PP. 105-114

A Sipmple Policy Update System for Sharing Outsourced Personal Health Records

Mr.D Surendra[1], B.Neelima[2]
[1]*Assistant Professor,Dept of CSE, Audisankara College of Engineering and Technology(AUTONOMOUS),Gudur,AP,India.*
[2]*PG Scholar,Dept of MCA, Audisankara College of Engineering andTechnology(AUTONOMOUS),Gudur,AP,India.*

Abstract
Electronic personal health records (PHRs) are used by many healthcare providers due to the high flexibility and accessibility of data outsourcing environments like the cloud computing environment. This enables individual patients to manage their own health data in such a resilient and scalable environment. However, because PHRs hold such sensitive data, security and privacy concerns are of paramount importance. Additionally, PHR owners should be allowed to set their own flexible and secure access policy for their outsourced data.Existing commercial cloud systems often offer symmetric or public key encryption as an add-on function to enable data secrecy for their tenants, in addition to the fundamental authentication capability. However, due to the significant key management overhead of symmetric encryption and the high maintenance costs associated with managing multiple copies of ciphertext for public key encryption solutions, such conventional encryption algorithms are not appropriate for a data outsourcing context. In this research, we design and construct a lightweight access policy updating system with safe and fine- grained access control for outsourced PHRs. Our suggested method is based on proxy re- encryption and ciphertext policy attribute- based encryption (CP-ABE) (PRE). To ensure complete traceability of policy changes, we also develop a policy versioning approach. Finally, we carried out the performance review to show how effective the suggested plan performs.
Keywords*: PHRs, access control, CP-ABE, policy update, proxyre-ncryption, policy versioning, performance evaluation.*

Date of Submission: 25-08-2022 Date of acceptance: 09-09-2022

I. Introduction

In a shared data environment that is outsourced, like a cloud storage system, the outsourced server must always be accessible to offer unrestricted access to shared data and services. Due to the cost savings and effective resource management offered by cloud providers, many businesses and individuals now prefer to store their important data on external servers like cloud storage. Prior to outsourcing their data to a cloud server, data owners typically encrypt their data to address privacy and security concerns. The best approach to safeguard sensitive information from unwanted access is to encrypt it. However, encryption by itself is insufficient to ensure strict security control. Another security perimeter that is usually necessary is an access control system. Attribute-based encryption (ABE)

[1] has been widely used in various works as a solution to this issue. One-to-many encryption with granular access control is offered by ABE. Additionally, it has access control and encryption features. Ciphertext-policy attribute-based encryption (CP-ABE) and key-policy attribute-based encryption are the two forms of ABE (KP-ABE). In CP- ABE, the data is encrypted using access policy and the user's decryption key is created using attributes. In KP-ABE, the encryption is carried out via a collection of characteristics while the user key is connected to the access policy. In terms of security enforcement, CP-ABE is favoured since the data owner may choose their own encryption policy. Using CP-ABE has benefits for managing group keys [3]. Decoupling abstract qualities from real keys is one of them. It lowers connection costs and offers granular data access management. It also accomplishes flexible one-to-many encryption rather than one-to- one, and it is seen as a potential solution for tackling the issues of safe and fine- grained data sharing and decentralised access control. However, when an attribute is revoked or a policy is updated, CP-ABE adds costly overheads such as ciphertext re-encryption, key re-generation, and key re-distribution. Because of the strong propagation effect to both ciphertext and user decryption key, these revocation and policy update actions must be carried out carefully. The calculation and transmission costs associated with key update are very expensive, especially when there are many users. The cost of data re-encryption and policy updates falls on the shoulders of the data owner, but the cost of communication depends on how many ciphertexts must be downloaded and re-uploaded from and to the data outsourcing environment. Such administrative burdens result in ineffective

implementation for actual data exchange scenarios. Additionally, it's conceivable that encryptors won't even be accessible when an update to the access policy is required.

In this research, we discover a practical method for updating CP-ABE access rules without requiring a new encryption procedure on the part of the data owner. Regarding the idea of PHRs being shared, the data owner, such as a patient, can share their data with anyone they like on a selective basis. We use symmetric encryption to encrypt data since it offers superior encryption performance, while the symmetric key is encrypted using the CP-ABE technique, in order to provide efficient encryption and increased efficiency of data access and policy updating. The cost for updating the policy only impacts the encrypted symmetric key since we utilise the CP-ABE technique to encrypt the symmetric key. Therefore, it is not necessary to re-encrypt all ciphertexts.This considerably lowers the proxy side calculation cost. Technically, the ciphertext re-encryption, which is theprincipal expense of policy updating, is handled by the proxy re-encryption (PRE) protocol.

The following is a summary of our contributions.

1. In a multi-authority data outsourcing scenario, we provide an access control approach for PHRs with lightweight policy updates. With our cryptographic design and newly introduced PRE technique, the re-encryption operation is offloaded to the proxy when the policy is amended, leaving the data owner to handle minor computation. Based on two-step encryption, the cost for both the data owner side and the proxy side is optimised.
2. We provide a policy versioning mechanism that enables all update events to be accurately documented and allows for the reconstruction of earlier iterationsof any policy for full analysis at any time.
3. To parallelize all crypto processes in the PRE system, we use parallel programming. According to our concept, the system will effectively re-encrypt all ciphertexts affected by a change in policy when the policy is revised.
4. To prove that our suggested approach is secure and effective for actual implementation, we offer security and performance studies.

The remainder of this essay is structured as follows. Section II discusses historical context and associated research. Section III provides background information on CP-ABE. The system we suggest is shown in Section IV. The method of policy versioning is presented in Section IV.The security analysis is presented in Section VI. The experiment and assessment are provided in Section
VII. The paper is concluded in Section VIII.

II. Literature Survey

One of the main overheads in CP-ABE that is degrading scalability and efficiency is policy updating. A data owner updates the policy by adding, changing, or removing logical gates (AND, OR, or M of N) or attribute values from the policies. Here, the data owner must first retrieve the policies that are often kept on site before updating the data. All ciphertexts that have been encrypted by the impacted policy must be re-encrypted after it has been modified. The ciphertexts will be transferred to the cloud after being re- encrypted.These procedures are often carried out by the data owners, who are also affected by the processing and communication overhead. In a PHR management situation, patients may need to alter the access policy used to encrypt their medical data in order to permit access to their medical treatments by other doctors at various institutions.

In general, there are two ways to provide policy updating in a CP-ABE setting: ciphertext update and proxy re-encryption (PRE).

A. CIPHERTEXTUPDATE

This approach requires the data owners to create update keys or tokens and submit them to the external server hosting the cyphertexts. The computation of the impacted characteristics yields the update keys, which are then used to update the associated ciphertext components. The access matrix and mapping functions are immediately impacted when there is a change in policy in most techniques adopting this mechanism, which rely on the linear secret sharing scheme (LSSS) [5, [10].

Belguith et al[4] .'s efficient access policy updating technique, for instance, uses small size ciphertext. Although this method is built on KP-ABE, which restricts data owners' capacity to define their own access policies. Additionally, the data owner is required to create the entire updated ciphertext. The burden of calculation shifts to the data owner as a result.

Li et al. suggested an effective policy update and file update for the CP-ABE setup in [5]. The key update produced by the data owner is used to update the ciphertext components. It lowers the client's storage and communication expenses. Additionally, it is shown that the proposed system is safe while assuming a decision q-parallel bilinear Diffie- Hellman exponent. However, in addition to creating the update keys based on the LSSS concept, the data owner must also possess the encrypted parts of the current

ciphertext.

To manage policy updates in the cloud server, Kan Yang et al. presented a ciphertext updating mechanism in . They looked at the cost of changing policies and presented linear secret sharing algorithms for adding and deleting characteristics from the AND, OR, and threshold gates of ABE policies (LSSS). With this strategy, data owners must create update keys based on generic order groups, and the complexity increases linearly with the number of policy attribute numbers. Therefore, using a resource-constrained device to perform bilinear computations is not appropriate.

A matrix-based policy updating method that he built using the CP-ABE scheme's fundamental encryption algorithm. According to the plan, the data owner must deal with a ciphertext update algorithm that compares a variety of characteristics between the old and the new policy. The policy size affects the computation and communication costs. The idea of a multi- keyword search over CP-ABE with dynamic policy was updated. This method updates the policy using the existing policy's encryption data without selecting a new secret value. After that, the access policy is updated using the Update Keygenalgorithm. The update key computation, which entails transforming the LSSS matrix and mapping function and comparing an old and new policy at the data owner side, is expensive if the policy contains a large number of attributes, even though this scheme is effective for ciphertext updates resulting from policy updates.

B. PROXYRE-ENCRYPTION (PRE)

Proxy re-encryption was initially presented by Mambo and Okamoto [6]. (PRE). The suggested method uses the idea of a delegator to re-encrypt the ciphertext that the originator originally transmitted. In this approach, neither the original data nor the decryption keys are revealed to the delegator. This idea was later adopted by several works [7]–[10]. because it delegated the proxy's expensive cryptographic operations to them. Some designs split the PRE server to handle solely the re-encryption process, and they employ a cloud server to communicate and calculate the secret component needed tofacilitate user revocation .

The authors of suggested the PRE approach to facilitate key updating. The user and the proxy must communicate in order for the proxy to get the value of the encrypted ciphertext along with the symmetric key, and for the proxy to be able to decipher some of the ciphertext. The user must next decode another portion of ciphertext that the proxy has returned to them. Users still have to deal with two troublesome procedures for decryption, and the communication cost between the user and the proxy is the major burden if there are several decryption events. This is true even when some of the processing for decryption is partially outsourced to the proxy.

we introduced a method called VL- PRE to help make it possible for the policy update to be carried out in the cloud in a productive and financially advantageous way. The primary technique is an optimization approach for PRE and re- encryption key creation. Although this method spares data owners from having to deal with complex cryptographic operations, the ciphertext re-encryption process at the cloud side is still based on the CP-ABE. Users that need to access the updated ciphertext may have performance issues if there is a significant amount of re- encryption jobs.

However, while using both proxy re- encryption and ciphertext updating, none of the aforementioned efforts have concentrated on the practicality perspective, as seen by three shortcomings. First off, the cost of re-encryption depends on a number of updated policy features that are inappropriate for mobile devices used by PHR owners, especially when the policy is large and is changed often. Second, no works have clearly explored the confidence of a delegated system, such as an outsourced server or a delegated proxy server. Existing works ignore the trust of key updates or any secret components communicated by the data owner and the delegator.Existing works ignore the trust of key updates or any secret components communicated by the data owner and the delegator. Finally, there are no methods that take full account of policy update traceability.

The practical and efficient application of secure outsourced CP-ABE policy updating with complete traceability is the focus of this article. To facilitate quick policy updates, we mix CP-ABE with a proxy re-encryption technique in this case. For effective data re-encryption, the suggested system is equipped with parallel processing techniques. The data owner can adaptably amend the policies kept in the outsourced data storage under our proposed method. Additionally, CP- cryptographic ABE's details are apparent to users, enhancing the tool's usability. We provide the policy versioning approach as another key component of our suggested access control system in order to provide robust accountability of policy modification history.

III. Background

The formal CP-ABE concept and associated definitions utilised in our suggested system are described in this section.

CIPHERTEXTPOLICYATT RIBUTE- BASEDENCRYPTION(CP-ABE)

Essentially, bilinear maps serve as the foundation for the ABE structure. The formal definition of bilinear maps is described here.

Bilinear Maps

Let e be a bilinear map, G0 G0 G1, and G0 G0 G1 be two multiplicative cyclic groups of prime order p. Let g serve as a G0 generator. Let H: 0, 1 G0 be the hash function used by the random oracle security model.

The following characteristics apply to the bilinear map e:

Bilinearity: for all $u, v \in G_1$and $a, b \in Z_p$,$e(u^a$, $v^b)=e(u,v)^{ab}$

Non-degeneracy:$e(g,g)/=1$.

Definition 1: Let a {set P1, P2, . .}. , Pnbegiven. A col-: lectionA2{P1,P2,...,Pn}ismonotoneIf ∀*B,Cif*

A monotone collection is what an access structure is. Non-empty subsets of the set A of P1, P2, . . . , Pn , i.e.

A 2{P1,P2,...,Pn}/

Definition 2: Access Tree T [2]: Assume T is an access structure represented by a tree. With the help of its offspring and a threshold value, each non-leaf node in the tree represents a threshold gate. If kx is a node's threshold value and numx is anode's number of children, then$0<k_x \leq num_x$.

The threshold gate is an OR gate when kx is 1, and an AND gate when kx is numx. An attribute and a threshold value, kx 1, characterise each leaf node x in the tree. The Koffin threshold gate is also permitted in T; in this instance, it has the form kx k, where k is the threshold value chosen for the Koffin gate. T is referred to as an access control policy, or ACP, in our approach.

The following are the four main algorithms that make up the conventional CP-ABE. Setup. The implicit security parameter is the only input the setup procedure requires. The master key MK and the public parameters PK are output.

Generation Key (MK, S). The method requires the master key MK and a list of characteristics S that identify the key as inputs. A user decryption key

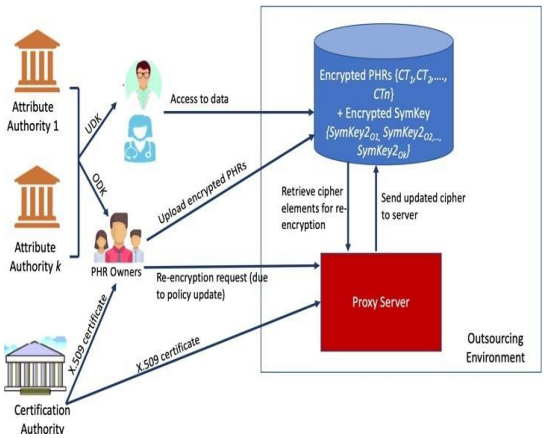

Fig.1. System model of out sourced PHRs sharing.

(UDK) is produced. Technically, bilinear maps provide the foundation for key generation.

Encrypt (PK, M, A). The public parameters PK, the message M, and the access structure A (also known as the access tree T) over the universe of attributes are all inputs to the encryption method. The algorithm creates a ciphertext CT after encrypting M.

Decrypt (PK, CT, SK). The method requires three inputs: a user, a ciphertext, and public parameters PK.

Proposed Approach
The system model and thecryptographic building block of our suggested approach are presented in this section.

a. System Model, part A
According to our concept, PHR owners upload encrypted data files, such as patient profiles and treatment records, to a cloud server. Users, such as doctors, can access the shared file if they have the necessary skills (a decryption key thatcomplies with the access control policy).
As illustrated in Figure 1, attribute authorities offer PHR owners and users a collection of attributes in the form of a user decryption key. Our architecture allows for numerous authorities to provide users the traits. For instance, a patient could receive keys from several organisations, such as hospitals or an insurance provider. In a setting where data outsourcing is common.

b. Construction of Systems
The cryptographic methods we offer are based on an extension of the original in our scheme.CP-ABE [1]. There are two encryption components in our system build. Data is first encrypted using symmetric AES encryption. Second, CP- ABE is used to encrypt the symmetric key. The outsourced server has these two encrypted results. Here, we define the notations used to describe our cryptographic methods and are displayed in Table 1.
The five main stages of our concept are System Setup, Key Generation, Encryption, Decryption, and Re-encryption. A list of the notations utilisedin our model is shown in Table.

Phase1: System Setup
The AA or data owner will perform the following six algorithms during this step.
Establish Attribute Authority (k) PKk, SKk, and PKx.k. This algorithm accepts an attribute authority as input. ID (k). The procedure selects a bilinear group G0 with a generator of prime order p. The next step is to select two random Zp. The public key is calculated as follows:
$PK_k=$ ^{n} G , $g,^{h}=g^{\beta}$,f = $g^{\frac{1}{}}$ $e(g,g)^{a}$
0 β

Type equation here.

Note that the secret key SKk is (, g) and that f is just used for delegation. For every attribute that the Ak has issued, the algorithm additionally publishes the public attribute keys (PKx,k).
2. ENCO- DKoid,kODKoid,k (ODKoid,k,SymKey1oid). In order to encrypt the PHR

owner decryption key ODKoid,k with AES encryption, the procedure uses symmetric key 1.

ENC_{AES} ($ODK_{oid,k,}SymKey1_{Oid}$)
$=ENCODK_{Oid,k}$

Generaterandomsecret
The PHR owner completes this phase byrunning the subsequent algorithm.
First, Gen R(r1, r2,... rn) R

The algorithm selects a collection of random seeds, rs, at random as input and produces a 256-bit random number, R, also known as a random string, with its position, Rp. Then it gives back R and Rp.
(2) Add R to a symmetric key that is shared. Substitute R(SymKey1Oid) RSk The proxy server then stores a secret RSk that is randomly generated.
Second phase: key creation

The AA oversees this stage. The user decryption key is created using the UserKeyGen algorithm (CP-ABE decryption key). The algorithm's specifics are described in the sections below.UDKuid,k, Suid,k, SKk, UserKeyGen The KeyGen algorithm accepts a list of characteristics (Suid,k) as input.

Policy Versioning

A change in access policies is a crucial problem since it directly links access rules and permission enforcement. Every ciphertext encrypted using an earlier version of the policy in CP-ABE has to be re-encrypted with the revised policy. When the update detail is insufficient for precise tracing, all update occurrences are typically captured in the log file. For instance, in a situation involving the outsourcing of health-care data, suppose that historical treatment records from the previous two years were encrypted using Policy A and preserved on the external server. Lineage of policy modifications and their current decryption keys are absolutely necessary if these medical data are to be inspected.

We offer a policy versioning approach that incorporates the policy linage preservation and policy retrieval mechanism in order to enable the detailed traceability of policy changes. The directed acyclic graph seen in Figure 3 can be used to explain the lineage of policy updates.

According to the policy lineage, historical policies were often either independently formed or derived from earlier policies. All update records are routinely stored in the database, as shown in Table 1, to provide the specifics of update history.

State 1 Thread of update policy string and query all enc keys, that were encrypted by the updated policy.
Input policy _id, new _policy_string

UPDATE policy _string =new
_Policy_String WHERE pid =ACP_{pid}

List <File>$EncSymKey2_{oid_}$list =SELECTall file WHERE ACPpid= pid
SEND $EncSymKey2_{oid}$_list to ReENCfunction.
Output none.

State 2 Re-encryption queue managementfunction and load balance handling of re- encryption thread based on file size

Input $EncSymKey2_{oid}$_list

FOREACH CT AS $EncSymKey2_{oid}$_list

Active_ thread = FIND re- encryption thread that contains list of summary filesize in queue
PUSH CT to active _ thread. queueEND FOREACH
Output none.

State 3 File Re-encryption

Input none

VARIABLE queue_ of _reEnc _file WHILE THREAD IS NOT CLOSED
IF queue_ of _reEnc _file is not empty

CT =pull first file fromqueue_of_reEnc_file
Run ReENC($PK_{k,}$rs, RSK, ACP',
$EncSymKey2_{oid,k}$)

END IF
END WHILE

Users may see the history from DAG and provide the policy version using our developed system to obtain any past policies. The chosen policy is then taken out of the policy versioning table. The complete traceability of policy updates is a benefit of this policy versioning record format.

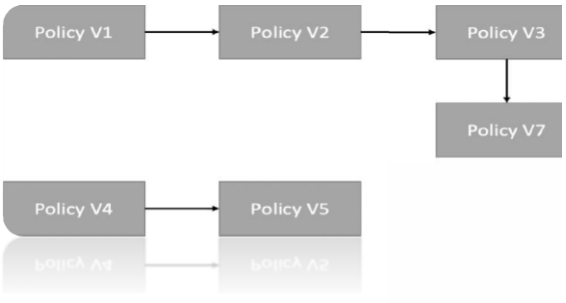

FIG3. Policyupdateslineage.

Data owners or auditors may examine modifications to policy structures and related files. If the past cases are required, any historical rules can be modified to allow file encryption or decryption scenarios. If some users' keys weren't changed, for instance, they could still need to use their old keys to decrypt files that were encrypted under earlier regulations.
By using the re-encryption algorithm, any previous policy versions may be recreated with our suggested policy versioning in order to re-encrypt the target file. In essence, the system auditing and policy lineage research are served by the suggested versioning approach. To protect the policy's content's confidentiality, thepolicyversioningtableise ncryptedwithdataowner'sX.509publickeya ndthenitcanbestoredintheoutsourcingserver

Performance Analysis
This section divides the performance analysis into two sections: an examinationof computational effectiveness and an examination of experimental results.

Policy ID	Ver. No.	Policy tree	Start date	End date	FileNo.
P01	1	Doctor or (NursedLevel >11)	01-07-2020	02-07-2020	F01,F02
P01	2	Doctor or (NursedLevel >10)	02-07-2020	03-07-2020	F01,F02
P01	3	Doctor or (NursedLevel > 9)	03-07-2020	04-07-2020	F01,F02,F04
…	…	…	…	…	…
P01	8	Doctor or	07	-	F01,F0

		(Nursean dLevel >10) or Mr.smith	- 07 - 20 20		2,F04
P02	1	Nurseand Level >5	01 - 07 - 20 20	06 - 07 - 20 20	F03
P02	2	Nurseand Level >6	07 - 07 - 20 20	-	F03,F05
…	…	…	…	…	…

Table 1: - provides example of policyversioning table where all historical changes of policy structure of all versions

EFFICIENCYANALYSIS

In this part, we contrast our scheme's functionality with that of Li et al[5] .'s and Ying et al[16] .'s schemes. We define the following notations in order to streamline the representation of computing cost for each scheme.
Let*p*betheelementsize inthe$G1$,$G2$, Z_p. G_0:ExponentiationoperationingroupG_0 G_1:ExponentiationingroupG_1
R_d = Random decryption over the messageor cipher text
N_c = : number of attributes associated withthe ciphertext or encrypted key.
Compared to [5,] our plan offers lower communication costs for policy updates. This is so that the proxy server only receives the random element and the new access control policy, which only accepts attributes that belong to Zip. The update key generation and matrix mapping element with a new access policy to be sent for cipher text updating incur communication costs in [5]. Regarding the cost of computation, scheme [5] demands that the data owner and cloud server conduct ciphertext update and update update key generation, respectively.

scheme	Policy update communication		
		USER	Proxy
5	$3\|Z_P\|$	$3Z_P$	$3G_P+Z_p$
ours	$R_d+\|Z_p\|$	N\A	$N_C\|P\|$

Table 2: Policy communication values

IV. Experimental Analysis

We set up the system simulation using a proxy server as a simulated outsourcing environment to assess the effectiveness of our suggested method. The Java Pairing-Based Cryptography library and the cp-abe tools are used to simulate the system
||
(JPBC). The test was run on an Intel(R)
Xeon(R)-CPU E5620 running at 2.40GHz. By comparing the processing times for encryption, decryption, and re-encryption of our proposed re-encryption process performed by PRE with multi-thread processing and without PRE, we assess the effectiveness of our scheme. In order to compare scheme [5] with our scheme, we employed JPBC to mimic the cryptographic design of scheme [5. We modify a variety of policy-related

factors in the simulation to compare the computing efficiency of encryption and decryption.simply changing a few of the policy's defining characteristics. It was put to the test with a file size of 50 KB. Figures 4a and 4b demonstrate how our technique offers shorter encryption time and decryption than Li et alscheme. .'s Our method gradually outperforms Li et alscheme, .'s particularly as the number of characteristics rises.
Despite the fact that our technique necessitates two encryption processes, CP- symmetric ABE's key encryption is employed for the second encryption stage. The encryption and decryption times are not significantly impacted by the overall processing time due to the relatively modest 128-bit symmetric key size. This clearly demonstrates the advantages of our suggested cryptography techniques.

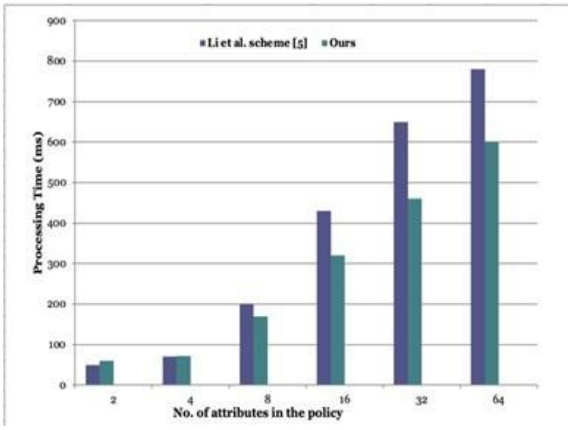

FIG 4. Comparison of encryption time b. Comparison of decryption time.

For our system [5], we analyse proxy re- encryption cost in addition to update key generation and ciphertext update costs to estimate the cost of the policy change. The outcomes of our suggested system, the Li et al. scheme, and the policy update time (ms) are shown in Figure 5. An increasing quantity of ciphertexts that need to be re- encrypted is used to estimate processing time. We applied the five qualities to the simulation. we used the 5-attributes policyto re-encrypt files having 20-KB size in average.

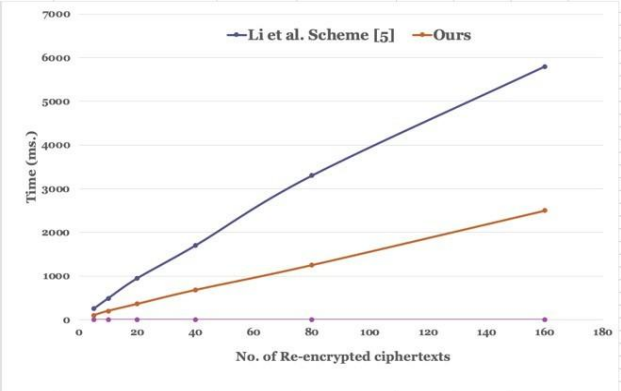

FIG 5. Comparison of re- encryption performance.

As can be seen from Figure 5, our proposed PRE with multithread processing takes less re-encryption time than the Li et al. scheme in a significant manner KeyGen update and ciphertext update are included in the cost of updating the policy. In our technique, the proxy merely uses the encrypted symmetric key to re-encrypt it. This benefit becomes much more apparent when there are several ciphertexts that need to be re-encrypted. The results of the studies support the notion that our suggested PRE scheme is effective in assisting PHRs owner to update the policy. Our suggested technique should be able to sustain a large number of ciphertext re- encryptions brought on by policy modifications in the PHRs outsourcing scenario, as the results obtained demonstrate its effectiveness. On order to take use of the scalability and other advantages of the cloud, it is therefore promising to run our re-encryption proxy in VM platforms or Linux containersin the actual cloud environment.resource resilience, and high availability.

V. Conclusion

We have put forth a policy updating approach based on proxy re-encryption and policy outsourcing. Our plan completely offloads the expense of updating policies to the external server. The re-encryption method also fully utilises multi-thread processing to allow high scalability and enhance system performance. We created a GUI tool for CP-ABE policy update implementation for the experiment. Data owners can use our system to upload data and policies to the external storage in an encrypted fashion. Administrators or data owners do not need to access the local database to obtain policies or communicate with the external server during the re-encryption process. Our web-based application allows for policy updates at any time and from any location.As a result, this offers transparent access control for the management of policy update management and file storage. We also suggested the policy versioning approach to make it possible to rebuild past policies effectively for thorough audits. Finally, we showed how well the file re-encryption worked. The outcomes demonstrated that the multi- thread re-encryption method worked better than the one without one. In our upcoming work, we will conduct in-depth tests to verify the cloud-based proxy using a bigger amount of data and access controls in a genuine cloud context.

REFERENCES

[1]. A.SahaiandB.Waters,‘‘Fuzzyidentity- basedencryption,’’inProc.24thAnnu.Int.Co nf.Appl.Cryptograph.Technique(EUROCR YPT)(Lec-ture Notes in ComputerScience). Berlin, Germany: Springer, May2015,pp. 457–473.

[2]. J.Bethencourt,A.Sahai,andB.Waters,‘‘Ciph ertext-policyattribute-basedencryption,’’ in Proc. IEEE Symp. Secur. Privacy, Oakland, CA, USA,May 2007, pp. 321–334.

[3]. L.Cheung,J.Cooley,R.Khazan,andC.Newp ort,‘‘Collusion- resistantgroupkeymanagementusingattribut e-basedencryption,’’Cryptol.ePrintArch.,Tec h.Rep.2007/161.[Online].Available:https:// eprint.iacr.org/2007/161.pdf

[4]. S.Belguith,N.Kaaniche,andG.Russello,‘‘PU-ABE:Lightweightattribute-basedencryptionsupportingaccesspolicyup dateforcloudassisted IoT,’’ in Proc. IEEE 11th Int. Conf. Cloud Comput. (CLOUD),Jul. 2018, pp. 924–927.

[5]. J. Li, S. Wang, Y. Li, H. Wang, H.Wang, H. Wang, J. Chen, and Z. You,‘‘Anefficientattribute- basedencryptionschemewithpolicyupdatea ndfileupdate in cloud computing,’’ IEEE Trans. Ind. Informat., vol. 15, no. 12,pp.6500–6509, Dec. 2019.

[6]. M. Mambo and E. Okamoto, ‘‘Proxy cryptosystems: Delegation of thepowertodecryptciphertexts,’’IEICETran s.,vol.E80-A,no.1,pp. 54–63,1997.

[7]. K. Liang, W. Susilo, and J. K. Liu, ‘‘Privacy-preserving ciphertext multi-sharing control for big data storage,’’ IEEE Trans. Inf. Forensics Security,vol. 10, no.8, pp. 1578–1589,Aug. 2015.

[8]. S.FugkeawandH.Sato,‘‘Embeddinglightwe ightproxyre-encryptionfor efficientattribute revocation in cloud computing,’’ J. High Perform.Comput. Netw., vol. 9,no. 4, pp. 299–309,2016.

[9]. Y.Kawai,‘‘Outsourcingthere- encryptionkeygeneration:Flexibleciphertex t-policy attribute-based proxy re- encryption,’’ in Proc. Int. Conf.Inf. Secur. Pract. Exper. (ISPEC), Beijing, China,2015, pp. 301–315.

[10]. X.Liang,Z.Cao,H.Lin,andJ.Shao,‘‘Attribut ebasedproxyre-encryptionwith delegating capabilities,’’ in Proc. 4th Int. Symp. Inf., Comput., Com-mun. Secur. (ASIACCS), 2009, pp. 276–286