JES
Journal of Engineering Sciences

Vol 10, Issue 11, Nov /2019
ISSN NO:0377-9254

SECURE AND LIGHTWEIGHT ACCESS CONTROL FOR PERSONAL HEALTH INFORMATION IN CLOUD BASED IOT DEVICES

B. MURALI KRISHNA[1], K. SWARNA LATHA[2]

[1]Asst. Professor, Dept. of CSE, PBR VITS, Kavali, A.P, India
[2]M.Tech, Dept. of CSE, PBR VITS, Kavali, A.P, India

Abstract – The eHealth trend has spread globally. Internet of Thing s (IoT) devices for medical service and pervasive Personal Health Information (PHI) system s play important roles in the eHealth environment. A cloud based PHI system appears promising but raises privacy and information security concerns. W e propose a cloud based fine grained health information access control framework for lightweight IoT devices with data dynamics auditing and attribute revocation functions. Only symmetric cryptography is required for IoT devices, such as wireless body sensors. A variant of ciphertext policy attribute based encryption, dual encryption and Merkle hash tree s are used to support fine grained access control, efficient dynamic data auditing, batch auditing and attribute revocation. Moreover, the proposed scheme also defines and handles the cloud reciprocity problem wherein cloud service provider s can help each other avoid fine s resulting from data loss. Security analysis and performance comparisons show that the proposed scheme is an excellent candidate for a cloud based PHI system.

Keywords – Personal Health Information, Cloud computing, Access control scheme.

I. INTRODUCTION

Internets of Thing s (IoT) devices for medical services are an emerging technology for caring for disabled or chronic patients. Combined with wearable medical sensor s and wireless communication, IoT devices can gather patient's health related parameters remotely and continuously. As a result, the electronic Health (eHealth) care business is emerging. The eHealth vision is to utilize state of the art medical technologies to prolong life expectancy significantly. Imagine physicians being able to access a tourist s Personal Health Information (PHI) regarding food allergy history from a medical record for a rapid diagnosis. A patient with chronic heart disease uses body sensors to detect irregular blood pressure and rushes to the hospital in time to survive. In these scenario s, the patient's medical record plays a key role in diagnosis therefore pervasive PHI service is essential for doctors and nurses to offer real time treatment.

One solution for an effective PHI system is to adopt cloud based storage to mitigate the burden of building and maintenance cost. However, outsourced PHI faces the challenge of security and privacy issues, f or instance, how to ensure that only the authorized requester can access the sensitive PHI or to prevent a semi trusted Cloud Service Provider (CSP) from leaking stored information. In addition, the Health Insurance Portability and Accountability Act (HIPAA) comprise a list of privacy requirements for protecting confidentiality from the data storage server. Data integrity at a semi trusted CSP is a n-other important concern. CSP s facing occasional catastrophic failures might decide to hide data errors from a patient for their own benefit. Although the data owner back s up his or her extremely important data in multiple CSPs, some CSPs might exercise mutual aid to avoid the huge cost of data loss. We call this the cloud reciprocity problem. The fact that a stored PHI would not only be accessed by medical workers but also updated by the patient requires support for data integrity verification for dynamic data operations.

One promising approach to solving such problems is to encrypt data in advance, prior to uploading to the cloud server. However, most existing PHI systems are not suitable for lightweight IoT devices because of the heavy cost of cryptographic computation. Furthermore, no scheme has dealt with the cloud reciprocity issue.

We propose a fine grained health information access control framework in the cloud for lightweight IoT devices with data dynamics auditing and attribute revocation functions. Regarding security and privacy, we use Ciphertext Policy Attribute Based Encryption (CP ABE) to perform fine grained access control on the part of a decryption key that is used to decrypt sensitive patient PHI. Basically, each Data Access Requester (has his/her own private keys associated with a set of attributes, and an essential decryption parameter TSK specifies an access policy over a defined universe of attributes. DAR can extract TSK to decrypt the encrypted PHI if and only if his/her attributes satisfy the access policy.

The main contribution s of this paper is as follows. (1) To the best of our knowledge, this is the first work suitable for lightweight IoT devices in PHI system s to achieve fine grained access control, dynamic data auditing, and user revocation simultaneously. (2) We first define and handle the problem of cloud reciprocity. (3) We propose a novel variant of proxy encryption to eliminate the involvement of HSP while DAR is accessing encrypted PHIs from a CSP and adapt the notion of CP ABE, MHT s, and dual encryption to offer the dynamic data operations, auditing, and user revocation functions. (4) Our

scheme can perform single block and batch auditing for multi patient data. (5) We prove our scheme is as secure as the existing ones.

II. RESEARCH METHOD

A) System Model

Fig. 1 depicts the architecture for the proposed scheme with the details explained as follows.

1) Registered user (Patient): Each patient registers at HSP and chooses different tree secret keys for different privacy levels. The patient transmits PHIs to CSP through HSP. The patient wears wireless IoT tags or sensors to gather vital signs, which are sent to a local gateway (e.g., PDA) that is responsible for PHI data aggregation and distribution to HSP. Note that the local gateway has only limited computational capability.

2) HSP: HSP is a totally trusted entity, which is responsible for transmitting PHIs to CSP and maintains computational resources. HSP also serves as the TPA to perform the task of data auditing on behalf of patients.

3) CSP: CSPs provide data storage services for storing encrypted PHIs and are equipped with mass storage space and computational resources.

4) DAR: Cloud users (DARs) request to access stored PHIs.

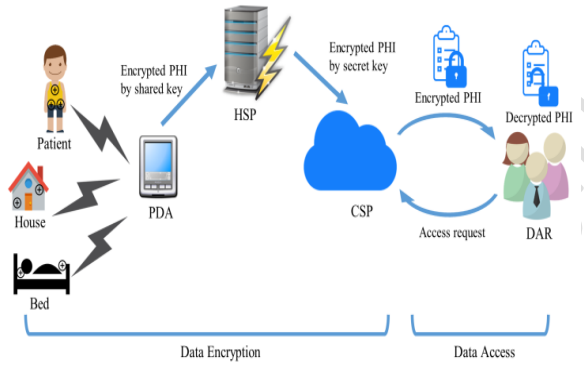

Fig. 1: System Overview.

B) Security Model

We consider the CSP to be semi trusted, i.e., honest but curious like those. This means that C SP might try to discover secret s in stored PHI if possible, but will honestly follow the protocol in general. Conversely, DAR s will try to access files beyond their privileges. To succeed, they might collude with other users. HSP is a trustable third party often regulated by the government.

C) Proposed Scheme

The main idea of the proposed scheme is to build secure and efficient cloud based PHI framework. The basic data flow of our scheme is shown in Fig. 2. In the registration phase, HSP grants Access Secret Keys ($ASK\tilde{i}$) to authorize DARs. At first, the patient s PHI data collected by IoT device s are encrypted using symmetric encryption (e.g. AES) by the shared key to HSP. HSP plays the important role in the proposed scheme of enforcing fine grained access control on PHI. First, HSP encrypts the plaintext PHI using a Data Encryption Key (DEK). Note that the DEK is an elaborate combination of Tree Secret Key (TSK) and Access Secret Key ($\tilde{}$) and cannot be derived by any single CSP or DAR. Furthermore, the TSK is encrypted using a variant of CP ABE as a message to fulfill the goal of sophisticated access control. Then, the CSP can compute the Access Decryption Key (ADK) if and only if the DAR is entitled to obtain the TSK. Finally, the DAR can decrypt the encrypted PHI by integrating $ASK\tilde{}$ and ADK. Conversely, the functions of batch auditing depend on aggregated Boneh Lynn Shacham (BLS) technology. The concept of dual encryption is adopted to adjust the CP ABE to equip the attribute revocation function. CSP constructs an MHT to perform efficient block auditing and data dynamics.

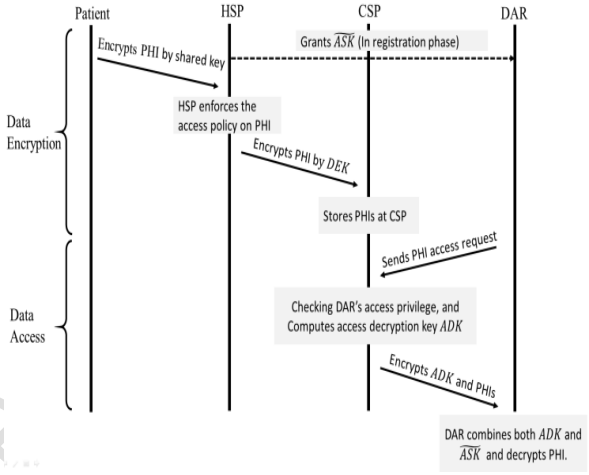

Fig. 2: Data flow of Our Framework

Let $\mathcal{U}=\{U1,...,Un\}$ be the universe of all users including patients and DARs, $\mathcal{T}=\{\lambda 1,\ldots,\lambda n\}$ be the universe of descriptive health attributes, and $\mathbb{g}=\{g1,\ldots,gn\}$ be the universe of such attribute groups. The operations related to constructing the proposed scheme are as follows:

AttKeyGen(CPMK , HA): The attribute key generation algorithm takes as input the HSP's master key CPMK and a set of health attributes $HA\subseteq\mathcal{T}$ with corresponding attribute group key $GK\lambda j$. It outputs a set of private keys CPSK $D=g(\alpha+r)/\beta,\forall\lambda j\in HA:Dj=gr\cdot H(\lambda j)rj,Dj'=(grj)1/GK\lambda j$ where $r1..m\in\mathbb{Z}q*$ is unique to each patient, $rj\in\mathbb{Z}q*$ for each attribute $\lambda j\in HA$, and $GK\lambda j$ is the corresponding attribute group keys.

TSKGen($PKPIDi,\rho j,w$): This function takes as input the patient PID i 's public key $PKPIDi=gxi$, random numbers $\rho 1..\rho m$, and the CSP storing key w. Next, HSP generates the tree secret keys $TSKi=(gxi)\rho 1/w$ ……$(gxi)\rho m/w$ for privacy levels 1 to m.

TSKEncryption(CPPK, TSK_i , A_i): This algorithm takes as input the public CP ABE parameter CPPK , a tree secret key TSK i for privacy level i as a message, and an access structure A i . HSP also prepares the selected attribute group keys $GK\lambda j$ that appear in A_i. The algorithm chooses a polynomial q x for each node x in the access structure A_i for privacy levels i . These polynomials are selected in a top down

manner, starting from the root node R. For each node x in A $_i$, the algorithm sets the degree dx of the polynomial qx to be one less than the threshold value kx of that node, i.e., dx = kx-1. For root node R, it picks a random $s \in \mathbb{Z}q*$ and sets q_R (0) = s. Then, it sets d $_R$ to other points of the polynomial q x randomly to define it completely. For any other node x, it sets q_x and selects d x other points randomly to define q x completely. Let Y be the set of leaf nodes in the access tree. The algorithm finally encrypts TSK i as a message and outputs

ET $Ai, \tilde{C}=TSKi \cdot e(g,g)\alpha s, C=\eta s, \forall y \in Y: Cy=g\, qy(0), Cy' = (H(\lambda y)\, qy(0))GK\lambda y$

TSKDecryption(ET, CPSK): This algorithm takes as input the ciphertext ET that contains an access structure A i and a private key CPSK $\in HA$. The decryption can be performed iff HA satisfies A$_i$. The algorithm performs recu r- sively. We first define a recursive algorithm Decrypt Node ET, CPSK, x)), where x is a node from tree access structure A$_i$, and it outputs a group element of $\mathbb{G}$ or $\perp$. If the node x is a leaf node, the function works as follows. If $\lambda x \in HA$, then

$$DecryptNode(ET, CPSK, x) = \frac{e(D_x, C_x)}{e(D'_x, C'_x)} = \frac{e\left(g^r \cdot H(\lambda_x)^{r_x}, g^{q_x(0)}\right)}{e\left(\left(g^{r_x}\right)^{\frac{1}{GK_{\lambda_x}}}, \left(H(\lambda_x)^{q_x(0)}\right)^{GK_{\lambda_x}}\right)} = e(g,g)^{rq_x(0)}$$

While $\lambda x \notin HA$, we define $Decrypt(ET, CPSK, x)=\perp$. When the node x is a non-leaf node, the algorithm $Decrypt(ET, CPSK, z)$ is called by all its child nodes z and outputs F z . Let S x be an arbitrary k x sized set of child nodes z such that $Fz \neq \perp$. If no such set exists, then the node was not satisfied and the function returns $\perp$. Otherwise, we compute

$$F_x = \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)}, where^{i=index(z),}_{S'_x=\{index(z): z \in S_x\}} = \prod_{z \in S_x} (e(g,g)^{rq_z(0)})^{\Delta_{i,S'_x}(0)} = \prod_{z \in S_x} (e(g,g)^{rq_{p(z)}(index(z))})^{\Delta_{i,S'_x}(0)} = \prod_{z \in S_x} (e(g,g)^{rq_x(i) \cdot \Delta_{i,S'_x}(0)} = e(g,g)^{rq_x(0)},$$

where $\Delta i, Sx'$ (0) is the Lagrange coefficients [10] and return the result. The decryption algorithm begins by calling the function on the roo t node R of the tree access structure. If the tree is satisfied by HA, we set $V=DecryptNode(ET, CPSK, R)=e(g,g)rs$, and the algorithm decrypts the ET by computing $\tilde{C}/(e(C,D)/V)=\tilde{C}/(e(\eta s, g\alpha+r\beta)/e(g,g)rs)=TSKi$.

PrivCheck (; ET): The CSP checks whether the DSR's HA (health attributes) satisfy the access tree structure by executing this function. If the DAR possesses the set of health attributes that satisfy U $_i$'s access tree structure A$_j$, the CSP can retrieve the $TSKi$ by calling TSKDecryption(ET, CPSK).

UpReEncrypt(ET, s′ λi): The re encryption algorithm is a randomized algorithm that takes as input the ciphertext ET , including an access structure Ai, random number $s' \in \mathbb{Z}q*$, and the changing attribute λi. It re encrypts ET as follows. The algorithm picks a new random $GK'\lambda i$ for the changing attribute λi and finally outputs

$$ET' = \{A_i, \tilde{C} = TSK_i \cdot e(g,g)^{\alpha(s+s')}, C = \eta^{s+s'}, C_i = g^{q_i(0)+s'}, C'_i = \left(H(\lambda_i)^{q_i(0)+s'}\right)^{GK_{\lambda_i}}, \forall y \in Y \backslash \{i\}: C_y = g^{q_y(0)+s'}, C'_y = \left(H(\lambda_y)^{q_y(0)+s'}\right)^{GK_{\lambda_y}}\}.$$

D) PHI Access Control Procedure

i. HSP Enforcement of Access Policy on PHI

In the beginning, the patient's PDA $U1$ and HSP employ the Diff-Hellman Key exchange protocol [4] to construct the shared key $KPH= PKHSPx1=ghx1=ghx1=PKU1\ h.$

Then, the patient encrypts PHI "pi" as $Epi=(pi)$ using the shared key and computes the message authentication code on PHI as $HMACEpi =h(Epi||PIDU1||KPH)$. Finally, the patient transmits $<Epi, HMACEpi>$ to HSP.

S2. After receiving $<Epi, HMACEpi>$, HSP first checks the correctness of $HMACEpi$ using the shared key $KPH=PKU1\ h.$ If the result is positive, HSP decrypts Epi using the shared key KPH to retrieve the PHI pi

S3. According to the assigned privacy level, e.g., privacy level 1, HSP encrypts "pi" using the corresponding tree secret key $TSK1=(gx1)\rho 1/w$ and $ASK\tilde{I}$ to enforce the fine-grained access control policy on PHI. First, HSP computes the data encryption key $DEK =(PKU1,\ g\rho 1w) \cdot e(gwh\ ,\ g1t1)=e(g,g)\rho 1x1w+wht1$, where $t1$ is the random number for privacy level 1. Then, HSP encrypts the PHI ' pi' as $Pi=pi\times DEK=pi \cdot e(g,g)\rho 1x1w+wht1$, and takes **TSKEncryption(*CPPK, TSK1, A1*)** to yield the corresponding $ET1$. HSP also puts the corresponding $ET1$, which is used to extract $TSK1$, and pro-vides a signature as $SigPi=(ET1||Pi||PHIID)h$. Finally, HSP transmits the $<ET1,,IID,SigPi>$ to CSP. HSP repeatedly performs this step for the other PHIs of a block.

S4. After transmitting all the PHIs of a block, as shown in Fig. 6, HSP constructs an MHT, whose leaf nodes are an ordered set of hashes of $h(Pi)$, as shown in Fig. 3, produces a digital signature on the root $\mathcal{R}=h(ha||hb)$ as $SigMHT\_R=H(R)h$ and sends the MHT root signature $<SigMHT\_R>$ to CSP. Moreover, HSP can delete $\{Pi, SigPi, SigMHT\_R\}$ to miti-gate the storage burden and achieve the blockless and stateless verification goal.

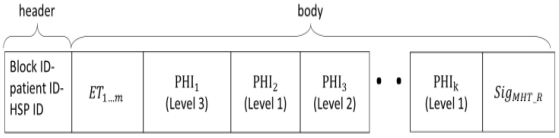

Fig. 3 Example of a PHI block

ii. Storing PHIs at CSP

S1. Receiving <$ET1$,,IID,$SigPi$>, CSP first checks the correctness of signature $SigPi$ by ($SigPi$,g)?= $e(H(ET1||Pi||PHIID)$,$PKHSP)$, is verified as follows. $e(SigP0$,$g)$?= $e(H(ET1||Pi||PHIID)$,$PKHSP)$ = $e(H(ET1||Pi||PHIID)$,$gh)$ = $e(H(ET1||Pi||PHIID)$,$PKHSP)$

S2. If the signature is valid, CSP saves {$ET1$,,IID,$SigPi$} and computes $h(Pi)$ for the construction of the corresponding MHT. CSP repeatedly executes this step until all encrypted PHIs of a block are successfully received and verified.

S3. Based on $h(Pi)$, CSP can build the corresponding MHT to extract the root $\bar{\mathcal{R}}$ and confirm the validity of the MHT signature $SigMHT\_R$ by $e(SigMHT\_R$,$g)$?= $e(H(\bar{R})$,$PKHSP)$. As long as $SigMHT\_R$ is correct, CSP also stores {$SigMHTR$} at its cloud storage server, according to the format of Fig.3.

iii. DARAuthorization and Access Procedure

S1. To access the encrypted PHI stored in the CSP, a DAR must send a request to the CSP. Then, CSP checks whether the DAR's $HA\hat{}$ (health attributes) satisfy the access tree structure by asking DAR to perform **PrivCheck(**$HA\hat{}$, ET**)**. If this is positive, CSP can obtain $TSK1=(gx1)\rho 1/w$ and compute the access decryption key $ADK1=e(TSK1,PKDAR)=e(g,g)x1\rho 1dw$. Next, CSP signs Pi, timestamp Ti, and $ADK1$ as $SigCSP=(Pi||Ti||ADK1)c$ and transmits <Pi,Ti,$ADK1$,$SigCSP$> to DAR.

S2. After checking the validity of the digital signature $SigCSP$, the DAR combines both $ADK1$ and $AS\tilde{K}1$, obtained from HSP, to decrypt the encrypted PHI Pi, as follows.

$PiADK1-d\cdot AS\tilde{K}1=pi\cdot(g,g)\rho 1x1w+wht1e(g,g)\rho 1x1$
$dwd.e(g,g)wht1=pi.$

III. RESULT AND ANALYSIS

A. *Functionality Comparisons*

We compare our scheme with the state of the art ,including ESPAC[11],SPS[16], Li etal. and Narayan. Those are based on patient-centric PHI sharing and use attribute-based ciphertext policy with privacy leveling to grant access control remotely. **Table 1** depicts the features of our scheme. Compared to other schemes, our scheme increases several functions as follows. (1) Batch auditing: HSP can perform multiple auditing tasks simultaneously.(2) Correctness assurance of data dynamics: our scheme supports the efficient correction assurance of data dynamic operations, such as PHI modification, insertion, and deletion. (3) Patient pseudonym: our scheme further enhances the privacy of patient real identity by using a pseudonym to avoid real name exposure during PHI upload-ing. (4) Resistance to cloud reciprocity problem: in our scheme, CSPs cannot share the stored PHI in each other to evade undesirable fines resulting from data loss.

Table 1. Comparison of Patient-Centric Access Control Schemes

Function \ Scheme	ESPAC [11]	SPS[16]	[14]	[20]	[29]	[30]	Our Scheme
Suitable for IoT sensors	Yes	No	No	No	No	No	Yes
Third-party auditability	No	Yes	No	No	No	No	Yes
Batch auditing	No	No	No	No	No	No	Yes
Correctness assurance of data dynamics	No	No	No	No	No	No	Yes
Attribute revocation	No	No	Yes	Yes	No	Yes	Yes
Patient pseudonym	No	No	No	No	No	No	Yes
Resistance to cloud reciprocity problem	No	No	No	No	No	No	Yes

B. *Performance Efficiency*

Here we analyze the performance efficiency of data encryption and data access in our scheme. Let Pair be the computation cost of a single pairing, Exp$\mathbb{G}$ the computation cost of an expo-nent operation in $\mathbb{G}$, Exp$\mathbb{G}$T the time for an exponent operation in $\mathbb{G}$T, Mult the computation cost of multiplications, and Hash the computation cost of a hash function into the group $\mathbb{G}$.In addition, Encsym and decsym are the computation costs of sym-metric-key encryption and decryption, respectively, as well as $hash$ is the hash function. Generally, the costs of pairing and exponentiation operations dominate the major computation time. A Type A curve in the pairing-based cryptography (PBC) library is used on a 3.0 GHz processor PC to provide groups in which a bilinear map is defined, and the implementation uses a 160-bit elliptic curve group based on the super singular curve $y2=x3+x$over a 512-bit finite field. The execution time for each operation is Pair=2.9 ms, Exp$\mathbb{G}$=1.0 ms, and Exp$\mathbb{G}$T=0.2 ms respectively. The execution time of single pairing is ten times more than the execution of scalar multiplication . Similarly to other schemes, we ignore the simple arithmetic computation costs containing multiplication, addition,and hash operations in the graphic comparison. Moreover, we assume k=10, the number of PHIs in one block.

First, we analyze the time costs of the **TSKEncryption**() and **TSKDecryption**() functions, which are used to protect those $TSKi$ and to obtain the tree secret keys $TSKi$, respectively. We assume At=10, the number of attributes appearing in the ac-cess tree and Au=15, the number of attributes associated with a user's private key. The several possible settings for data encryption are shown in Fig. 4.The computation cost sof **TSKEncryption**() and **TSKDecryption**() are (2At+1) Exp $\mathbb{G}$+1$*$ Exp $\mathbb{G}$T$\cong$21.2ms and (2Au+1) Pair + logAt$*$ Exp$\mathbb{G}$T$\cong$90.1ms, respectively.

For data encryption, shown in Table 2, first, the patient must generate shared key $KPH=ghx1$, encrypt PHI "pi" as $Epi=(pi)$, and sign the message authentication code$HMACEpi$ =$h(Epi||PIDU1||KPH)$. The corresponding computational cost is$Exp\mathbb{G}$,ym,and $hash$, respectively. After receiving<Epi, $HMACEpi$ >, HSP must verify HMA , generate share key $KPH=ghx1$, and decrypt Epi. The corresponding computational cost is $hash$,,and

$decsym$,respectively. Second, HSP must generate the data encryption key=$e(PKU1,\ g\rho 1w)\cdot e(gwh\ ,\ g1t1)=e(g,g)\rho 1x1w+wht1$, encrypt the PHI $Pi=pi\times DEK=pi\cdot e(g,g)\rho 1x1w+wht1$,sign $SigPi=H(ET1||Pi||PHIID)h$, and produce a digital signature on the root $\mathcal{R}$ as $SigMHT\_R=H(\mathcal{R})h$. The corresponding computational cost is $3-Exp\mathbb{G}$+2-$Pair$,+$Pair$,and $(k+1)(Hash+Exp\mathbb{G}T)$,respectively. Finally, CSP stores PHIs and checks $SigPi$ and $SigMHT\_R$. The corresponding computational cost is (k+1)($Hash$+2-$Pair$).

Table 2. Execution Time of Data Encryption In Our System

Data Encryption(Our Scheme)					
Procedure / Phase	key generation	encryption	signs the signature	decryption	verify the signature
Patient → HSP	$Exp_{\mathbb{G}}$	Enc_{sym}	$hash$	/	/
HSP	$Exp_{\mathbb{G}}$	/	/	dec_{sym}	$hash$
HSP → CSP	$3-Exp_{\mathbb{G}}$+2-$Pair$	$Mult+Pair$ $TSKEncryption()$	$(k+1)(Hash+Exp_{\mathbb{G}})$	/	/
CSP	/	/	/	/	$(k+1)\,(Hash$+2-$Pair)$

Table 3. Comparison of Data Access Efficiency with Other Schemes

Data Access							
Procedure / Scheme	Phase	key generation	encryption	signs the signature	decryption	verify the signature	revocation
Our Scheme	CSP → DAR	$Pair(ADK)$; $Exp_{\mathbb{G}}$(share key)	Enc_{sym}	$Hash+Exp_{\mathbb{G}}$	$TSKDecryption()$	/	/
	DAR	$Exp_{\mathbb{G}}$	/	/	$dec_{sym}(ADK)$; $Exp_{\mathbb{G}_T}+2-Pair$+2-$Mult$(PHI)	$Hash$+2-$Pair$	$A_u * Exp_{\mathbb{G}}$ (attribute revocation for DAR)
SPS[16]	HSP → CSP	$Pair+Exp_{\mathbb{G}_T}(\acute{K})$; $Exp_{\mathbb{G}}$(share key)	$Mult+Pair(\acute{K})$; $Mult+Pair+Exp_{\mathbb{G}_T}$ (PHI)	$Hash+Exp_{\mathbb{G}}$	$Decrypt_2()$	/	/
	CSP	$Exp_{\mathbb{G}}$	/	/	$Mult+Pair$	$Hash+Exp_{\mathbb{G}}$ +2-$Pair$	/
	CSP → DAR	$Exp_{\mathbb{G}}$	$Mult+Pair(\acute{K})$; $Mult+Pair$(PHI)	$Hash+Exp_{\mathbb{G}_T}$	/	/	/
	DAR	$Exp_{\mathbb{G}}$	/	/	$Mult+Pair(\acute{K})$; $Exp_{\mathbb{G}_T}+Pair$+1-$Mult$(PHI)	$Hash+Exp_{\mathbb{G}}$ +2-$Pair$	/
[14]	PUD user	/	/	/	$\sim(2k+1)Pair$	/	$\|\gamma'\|Exp_{\mathbb{G}}$ (user revocation)
ESPAC[11]	Data requester	/	/	/	$(2k+1)Pair+\log t * Exp_{\mathbb{G}_T}$	/	/

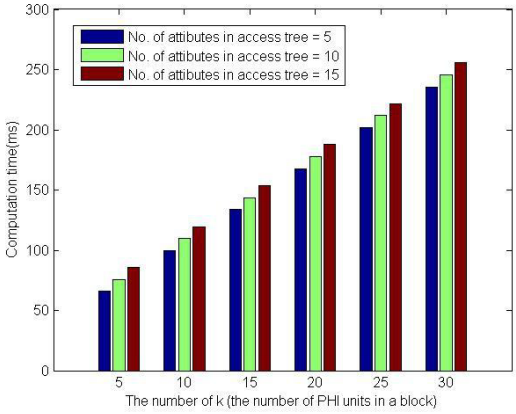

Fig. 4 Computation time vs. number of PHI units in a block

For data access, we compare with previous schemes in terms of key generation, encryption, signing the signature, decryption, verifying the signature, and revocation. Note that ESPAC is based entirely on CP-ABE without a role of HSP, hence some functionalities cannot be achieved. Therefore, in the graphic comparison, we focus only on the comparison with SPS. In SPS, SPS requires HSP to re-encrypt PHI for each data access. However, by the elaborately proposed proxy encryption, our scheme can transmit PHI directly to DAR from CSP. Therefore, SPS has four extra $Exp\mathbb{G}$ operations, three extra $Exp\mathbb{G}$T operations, and seven extra pairing operations. Hence, the data access time of our scheme is less than the data access time of SPS by approximately24.9ms. Regarding attribute revocation, our scheme takes $Au*Exp\mathbb{G}$ to perform attribute revocation for DAR, where Au is the number of attributes associated with private keys of a user. Takes $|\gamma'|Exp\mathbb{G}$ to perform user revocation, whereγ' is a minimal number of attributes to revoke a user. The computational costs of our schemes are nearly the same. Furthermore, SPS requires132.7 ms to execute each data ac- cess, but our scheme requires107.8 ms. Therefore, as the number of DARs increases, the times of data access increases, and our advantages will also increase linearly as illustrated in Fig. 5. Regarding the efficiency of batch auditing, we implement a timed batch auditing test, where the x-axis represents the number of auditing tasks, and y-axis stands for the average per task auditing time. From Fig. 6,we realize that batch auditing can reduce the TPA's computational cost more than 80 percent of per-task auditing time, as compared with individual auditing. In addition, regarding a single block audit, our scheme is also faster than SPS[16]by about 9ms.

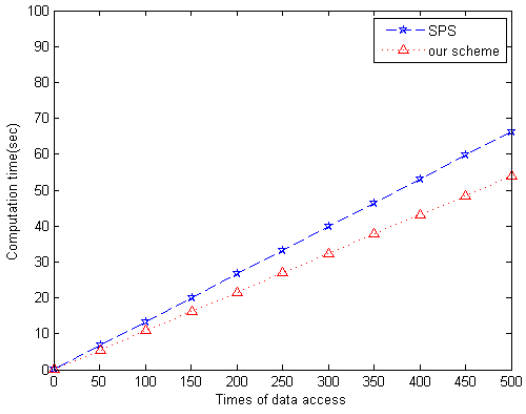

Fig. 5 Comparison in the data access time between our scheme and SPS

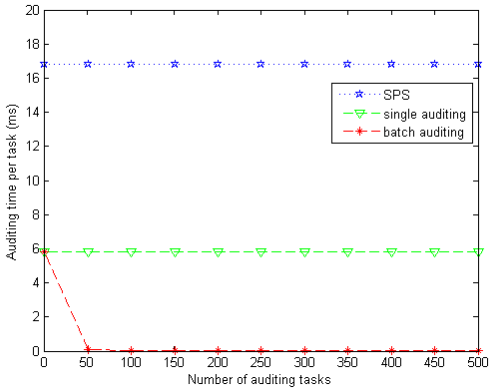

Fig. 6 Comparison in auditing time between batch, single auditing and SPS

IV. CONCLUSION

In this paper, we proposed a lightweight access control framework suitable for healthcare IoT devices to achieve the functions of fine grained access control, efficient revocation, and dynamic data verification. To the best of our knowledge, this work is the first framework with complete functions for cloud based PHI systems. Lightweight IoT devices use only symmetric encryption to upload PHIs to HSP. Both efficient single block and batch auditing are fully supported. Moreover, this paper first defines and handles a potential security issue the cloud reciprocity problem. The security analysis and performance evaluation results show that our scheme is a promising framework for cloud based PHI systems.

REFERENCES

1. S. Amendola, R. Lodato, S. Manzari, C. Occhiuzzi, and G. Marrocco, "RFID Technology for IoT Based Personal Healthcare in Smart Space," IEEE Internet of Things Journal , v ol . 1, n o . 2, pp. 144 152, 2014.
2. V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute based encryption for fine grained access control of encrypted data," Proc. ACM Computer and Communications, 2006.
3. R. C. Merkle, "Protocols for public key cryptosystems," Proc. of IEEE Symp o- sium on Security and Privacy , pp. 122 133, 1980.
4. W. Diffie, and M. Hellman, "New directions in cryptography," IEEE Transaction on Information Theory , v ol. 6, n o . pp. 644 654, 1976.
5. D. Boneh, B. Lynn, and H. Shacham, "Short Signature from the Weil Pairing, Pairing," Proc. Adv. ASIACRYPT'01 , pp. 514 532, 2001.
6. M. Bellare, R. Guerin, and P. Rogaway, "XOR MACs: New methods for message authentication using finite pseudorandom functions," Proc. Adv. CRYPTO'95 , pp. 15-28, 1995.
7. R. Rivest, "The RC5 encryption algorithm," P roc. Fast Soft. Encryption , pp. 86-96, 1995.
8. Y. Tsiounis, and M. Yung, "On the security of ElGamal based encryption," Proc. PKC , pp. 117-134, 1998.

AUTHORS

B. Murali Krishna has received his B.Tech in CSE and M.Tech degree in Computer science from JNTU, Hyderabad in 2006 and ANU, GUNTUR in 2010 respectively. He is dedicated to teaching field from the last 11 years. He has guided 15 P.G and 20 U.G student batches. His research areas included AI and Cloud Computing. At present he is working as Assistant Professor in PBR VITS, Kavali, SPSR Nellore, Andhra Pradesh, India.

K.SWARNA LATHA has received his MCA degree in PBR VITS affiliated to JNTU, Anantapur in 2017 and Pursuing M.Tech degree in Computer science in PBR VITS affiliated to JNTU, Anantapur in 2019.