Efficient Secure and Privacy Preserving Data Access Control Scheme for Multi-Authority Personal Health Record Systems in Cloud Computing

Imad El Ghoubach*
SIA/SSC laboratory
Faculty of Sciences and Technology
Sidi Mohamed Ben Abdellah University
Fez, Morocco
Email: imad.elghoubach@usmba.ac.ma

Fatiha Mrabti
SSC laboratory
Faculty of Sciences and Technology
Sidi Mohamed Ben Abdellah University
Fez, Morocco
Email: fatiha.mrabti@usmba.ac.ma

Rachid Ben Abbou
SIA laboratory
Faculty of Sciences and Technology
Sidi Mohamed Ben Abdellah University
Fez, Morocco
Email: rachid.benabbou@usmba.ac.ma

***Abstract*—The personal health record system allows the patients to learn more on their health indicator at any moment, but since the resources are stored in a semi-trusted cloud service provider, it is important to maintain data confidentiality while providing granular, scalable and flexible access control. For this reason, several schemes have been proposed. In this paper we propose a multi-authority attribute based encryption with semi-outsourced decryption that is able to provide a scalable and secure data sharing scheme while maintaining a low computational overhead.**

***Keywords*—Cloud Computing, personal health record system, Data Sharing Service, Data Privacy, Access Control, Security Analysis, Outsourced decryption, CP-ABE, P2E.**

I. Introduction

Since its emergence, the cloud computing continued attracting organizations who sow benefits in its significant computational capabilities that might not be affordable otherwise [1], [2]. Like other organization, health care systems are considering using the cloud computing in their services, especially with the emergence of the personal health record system (PRH), in which the patients are able to store, manage and share their own health information pervasively. The health care system is considering third party service providers in order to reduce the cost of maintaining specialized data centers and achieve resource sharing [3], but such implementations will require storing sensitive personal health information in a semi-trusted service provider's server. This will arise many concerns on the security and privacy of the outsourced data [4].

To ensure the security of PRH, it is essential to provide a scheme that is able to ensure the confidentiality while maintaining a fine-grained access control. Many mono-authority schemes where proposed recently [5]–[7], but since medical information may be shared with users that belongs to multiple authorities (like researchers for example), multi authority systems are required. Schemes in [8] and [9] were able to insure a secure and scalable multi-authority data sharing scheme, but since the stored data is usually accessible from various types of devices, where some of the users have limited computational power, a scheme with an efficient decryption processes is required. One of the main requirement, when designing an access control scheme, is the computational efficiency. Its important to have an efficient encryption operation, but the decryption and revocation operations are used more frequently which makes the efficiency of these operations even more important. Moreover, the users of the storage services may access the stored data using devices with limited computational capabilities (smart phones, tablets), thus making the efficiency of the decryption operation even more critical.

978-1-5090-3837-4/16/$31.00

In this paper, we introduce a multi-authority scheme with semi-outsourced decryption (MABE-SOD), we included a proxy-decryption server which will carry out most of the decryption process for the users. Moreover we provide an efficient attribute revocation scheme by outsourcing the cipher text update to the cloud server. The security of our method is ensured using two types of global IDs : the first ID generated by the Private Key Generator (PKG) will render the users unable to collude since each of their key is embedded with a different ID. The second ID is generated by each Attribute Authority, this ID is embedded with each of the users keys and it ensures that, even after the decryption process, the Proxy server will be unable to recover the encryption key. We also achieve the same level of fine-grained access control ensured by the DACC [9] scheme, this allows us to achieve an effective and privacy preserving cloud data sharing scheme with semi-outsourced decryption.

This paper is organized as follows: we present related works on cloud security in section 2. In section 3 we introduce our system and adversary models for our multi-authority construction. Section 4 provides the details of our scheme. In section 5 we provide the security and performance analysis. Finally, section 6 concludes the whole paper.

II. Related Works

Sehai and Water [10] were the first to propose the concept of Attribute based encryption (ABE), in which the message is

encrypted using a set if attributes and a threshold number d. In this scheme the user is only able to decrypt the message if he has at least d matching attributes. The scheme has given way to two classes of ABE schemes; Key-Policy attribute based encryption (KP-ABE) [11] and cipher-text policy attribute based encryption (CP-ABE) [5]. The KP-ABE and CP-ABE schemes differ fundamentally in the use of the access policy. In KP-ABE the access policy is used to generate the user's key in order to describe his access rights and he is only able to recover the message if he finds; among the attributes describing the message; a subset of attributes that satisfies his access policy. By contrast the CP-ABE scheme uses the access policy to encrypt the message and ensure that only a user who has attributes that satisfies the access policy is able to recover the message.

Yu et al. [6] reduced the revocation overhead of the KP-ABE scheme by delegating the re-encryption operation to the cloud server using the techniques of proxy re-encryption and lazy re-encryption. However, the cloud server learns parts of the users' keys when performing the revocation process. [12] and [13] also combined the CP-ABE scheme with proxy re-encryption. [14] increased the flexibility of KP-ABE by using non-monotonic access structures. However the addition of the negative words causes an increase in the encryption overhead.

X. Dong et al. [7] provided an efficient encrypting, decrypting and revocation process, but the decryption process is still expensive for the users with limited computational power. Green et al. [15] outsourced the decryption process to the cloud server to reduce the user's computational overhead, but the scheme is only viable in the case of a single authority system. Ruj et al. [9] proposed Distributed access control in clouds (DACC); a Multi-authority attribute based encryption scheme that, uses LSSS matrix to provide an efficient encryption, decryption process, but the overhead of the decryption process is still important for a user with limited computational power.

III. Models

A. System Model

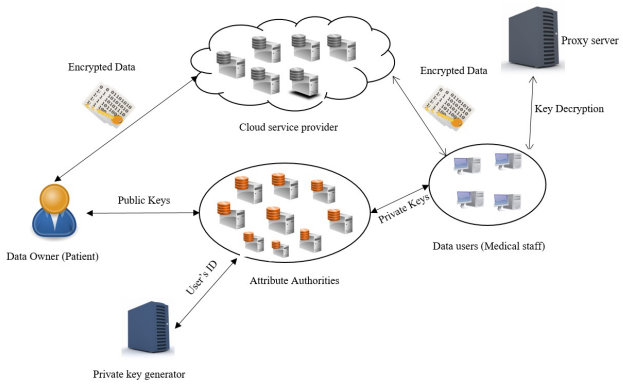

Fig. 1: System Model

The system model consists of six main actors: **Cloud Service Provider (CSP)**: delivers the required data storage space and the computation power. **Data Owner (Patient)**: stores his data in the cloud server and relies on the CSP for the maintenance. The data owner is an individual, an enterprise or an organization. **The Proxy Server** : in charge of executing most of the decryption computations. We suppose that there is no contact between the CSP and the Proxy server. **Attributes authorities (AA)**: Every AA is an independent attribute authority that is responsible for issuing, revoking and updating users attributes according to their role or identity in its domain. Every AA can manage an arbitrary number of attributes and each attribute is associated with a single authority. **Private Key Generator (PKG)**: a trusted third party, computes the private keys for the users. The PKG is only responsible for delivering the users' Identities (ID) to AA, thus it does not hold any information about the secret keys or the data stored in the cloud servers. **Data consumers or users (doctors, nurses, ...)**: who downloads the data shared by the data owner from the CSP and decrypts it using their secret keys.

B. Adversary model

In our model, we consider a semi-trusted cloud service provider that behaves appropriately most of the times, but for some benefits, the CSP might tries to find out as much secret informations as possible. This means that the CSP might collude with a small number of malicious users for the purpose of recovering data contents when it's highly beneficial. We also assume that a small number of malicious users might try to collude their keys in order to gain access to unauthorized data. Furthermore, we also consider the possibility of collusion between the CSP and the Proxy server. Moreover, since its the Multi-authority construction, we also bear in mind the case where one or more Attribute Authorities are corrupted, in which case we need to insure that even when colluding between them or with other entities of the system, they won't be able to recover any information about the data. The communication channels between different parties in our system are supposed secure under existing security protocols.

C. Health Care Security Requirements

In a health care system, the privacy of the patient data is a major issue, since any modification to the patient's information may lead to a mis-diagnosis, which have serious repercussion on the patient's health. Moreover, since the data is shared with different parties of the system (Medical doctors, pharmacist, nurses) an efficient and scalable access control is required.

The main goal of our work is to provide an efficient and secure data sharing scheme, that is able to maintain a fine-grained access control, data confidentiality and an efficient revocation process. Furthermore, we aim to reduce the computational overhead of the decryption operation since the users may access the data from low end devices (Tablets, smart phones, ...).

IV. The Design of MABE-SOD

A. Overview

Similar to [5], [9] and [16], in our method each file is described by a set of attributes which is used to construct an access tree and assigned it to the file. Each user has a set of attributes that describes his access privileges (assigned to him by the data owner) and a unique ID. For each attribute assigned to the user a key is created, a user is only able to decrypt the file if he has a set of attributes that satisfies the file's access policy.

Our method is characterized by four principle algorithms : **setup, encryption, key generation, Decryption**. In the setup phase the AA chooses the universe of attributes that will be used to describe the privileges in its domain and generate the system parameters for each attribute. The data owner receives the public keys of each concerned authority, then chooses a set of attributes and constructs an access policy to describes the access privileges and finally uses the encryption algorithm to encrypt his data. In the key-generation operation, each AA will choose a set of attribute that describes the user's role in its domain, this set will be used with the ID received from the PKG to generate the user's secret key. Finally, the user uses his secret key to decipher the data with the help of the proxy server, using the decryption algorithm; the file will only be decrypted if the user has a set of attributes that satisfies the file access tree.

B. Format of the access policy

The access policy can be expressed by a binary access tree A with logic gates as intermediate nodes and attributes as leaf nodes [7], [9], [17]. Using the scheme in [18] any access tree A can be converted to a Linear Secret Sharing Scheme (LSSS) matrix $\mathbf{M}$ with ρ as the permutation function mapping each attribute in A to its corresponding row in $\mathbf{M}$ .In Our scheme, a message $\mathbf{Ma}$ is encrypted with a LSSS access structure $(\mathbf{M}, \rho)$, only a user who has a subset of attributes that maps to a set of rows with $(\mathbf{1}\,\mathbf{0}\,\ldots\,\mathbf{0})$ is in the span of these rows, can decrypt the message correctly.

C. The proposed scheme in detail

Let S_A and S_U denote the set of attributes authorities and the set of users in the system respectively. Let G_1 and G_2 be the multiplicative groups with the same prime order p and $e : G_1 \times G_1 \rightarrow G_2$ be the bilinear map. Let g_1 be the generator of G_1. Let $H : \{0,1\}* \rightarrow G_1$ be a hash function.

1) Attribute Authorities Setup: Each authority $k \in S_A$ runs the setup algorithm using $W_{A,k}$ (the set of attributes that AA_k manages). For each attribute $i \in W_{A,k}$ the AA_k generate two random numbers $\alpha_{i,k}, \beta_{i,k} \in Z_p$, thus the secret key is:

$$Sk_k = \{\alpha_{i,k}, \beta_{i,k}, i \in W_{A,k}\}, \tag{1}$$

then publishes the public key:

$$Pk_k = \{g_1^{\alpha_{i,k}}, g_1^{\beta_{i,k}}, i \in W_{A,k}\}. \tag{2}$$

2) Encryptions: To encrypt a Message $\mathbf{M_a}$, the owner begins by defining a set of attributes I from the attributes belonging to the involved AAs. Using those attributes, he defines an access policy Matrix $\mathbf{M}$ then chooses two random vectors v and w such as the first entry of v is s and the first entry of w is 0, then computes $\lambda_x = v \cdot \mathbf{M_x}$ and $w_x = w \cdot \mathbf{M_x}$. Then for each leaf node x, the owner chooses a random number $r_x \in Z_p$ and computes:

$$\begin{aligned} D_{x,1} &= g_1^{\lambda_{\rho(x)}} g_1^{\alpha_{\rho(x)} r_x}, \\ D_{x,2} &= g_1^{r_x}, \\ D_{x,3} &= g_1^{\beta_{\rho(x)} r_x} g_1^{w_{\rho(x)}}, \end{aligned} \tag{3}$$

then encrypt the Message $\mathbf{M_a}$ as follows:

$$\mathbf{M_e} = Enc^{sym}_{e(g_1,g_1)^s}(\mathbf{M_a}).$$

Finally, the data owner uploads the encryption file $D = \{\forall x, \{D_{x,1}, D_{x,2}, D_{x,3}\}; (\mathbf{M}, \rho), \mathbf{Me}\}$ to the cloud server.

3) Keygen by the AAs: For every user in S_u, each authority $A_k \in S_A$ chooses a set of attributes I_u to assign to the user (according to his role in its domain), then acquires his ID from the PKG. Finally, it computes the user's secret key as follows:

$$\begin{aligned} Sk_{u,k} = &\{\{g_1^{\frac{\alpha_{i,k}}{Z_k}} H(ID_u)^{\frac{\beta_{i,k}}{Z_k}}, \forall i \in I_u\}, H_k = H(ID)^{\frac{1}{Z_k}}, \\ &Z_{u,k} = g_1^{\frac{1}{Z_k}}, ID_{u,k} = Z_k\} \end{aligned} \tag{4}$$

with $ID_{u,k}$ denotes the ID generated by AA_k for the user i.

Each $Sk_{u,k}$ is then encrypted using the user's public key and delivered to the user via the cloud server. In this manner, only the user can decrypt the key using his private key. Finally, after retrieving all the keys from the authorities, the user's secret key becomes : $Sk_u = \{Sk_{u,k}, \forall k \in S_A\}$.

4) Decryption: The user receives a cipher-text D:

$$D = \{\forall x, \{D_{x,1}, D_{x,2}, D_{x,3}\}; (\mathbf{M}, \rho), \mathbf{Me}\},$$

chooses a set of attributes Att_u that satisfies the data access policy, then for each attribute $att_i \in Att_u$, the user sends the corresponding key:

$$Sk_{att_i} = \{g_1^{\frac{\alpha_{i,k}}{Z_i}} H(ID_u)^{\frac{\beta_{i,k}}{Z_i}}, H = H(ID)^{\frac{1}{Z_i}}, Z_u = g^{\frac{1}{Z_i}}\}$$

and the corresponding $CT_i^{'} = \{D_{att_i,1}, D_{att_i,2}, D_{att_i,3}\}$ to the proxy server which will, for each attribute $att_i \in Att_u$,

computes:

$$
\begin{aligned}
c^{'}_{att_i} &= \frac{e(D_{att_i,1}, Z_u)e(H, D_{att_i,3})}{e(sk_{x,u}, D_{att_i,2})}, \\
&= \frac{e(g_1,g_1)^{\frac{\lambda_{att_i}}{Z_i}} e(g_1,g_1)^{\frac{\alpha_{att_i} r_{att_i}}{Z_i}}}{e(g_1,g_1)^{\frac{\alpha_{att_i} r_{att_i}}{Z_i}}} \times \\
&\quad \frac{e(H(ID),g_1)^{\frac{w_{att_i}}{Z_i}} e(H(ID),g_1)^{\frac{r_{att_i}\beta_{att_i}}{Z_i}}}{e(H(ID),g_1)^{\frac{r_{att_i}\beta_{att_i}}{Z_i}}}, \\
&= e(g_1,g_1)^{\frac{\lambda_{att_i}}{Z_i}} e(H(ID),g_1)^{\frac{w_{att_i}}{Z_i}}, \\
c^{'}_{att_i} &= (e(H(ID),g_1)^{w_{att_i}} e(g_1,g_1)^{\lambda_{att_i}})^{\frac{1}{Z_{att_i}}}. \qquad (5)
\end{aligned}
$$

Then sends $C^{'} = \{c^{'}_{att_i}, att_i \in Att\}$ to the user. The user will then compute:

$$
\begin{aligned}
\prod_{att_i} (c^{'}_{att_i})^{Z_{att_i}} &= \prod_{att_i} \left((e(H(ID),g_1)^{w_{att_i}} e(g_1,g_1)^{\lambda_{att_i}})^{\frac{1}{Z_{att_i}}} \right)^{Z_{att_i}}, \\
&= \prod_{att_i} e(H(ID),g_1)^{w_{att_i}} e(g_1,g_1)^{\lambda_{att_i}}, \\
&= e(H(ID),g_1)^{\sum_i w_{att_i}} e(g_1,g_1)^{\sum_i \lambda_{att_i}}, \\
&= e(H(ID),g_1)^0 e(g_1,g_1)^s, \\
\prod_{att_i} (c^{'}_{att_i})^{Z_{att_i}} &= e(g_1,g_1)^s. \qquad (6)
\end{aligned}
$$

Then recovers $\mathbf{M_a} = Dec^{sym}_{e(g_1,g_1)^s} = \mathbf{M_e}$.

D. Revocation

1) Attribute revocation: Revoking a user's access rights means that the owner needs to change the attributes associated with his keys, for KP-ABE based schemes [6], [11] it would mean that we need to change the access structures of all the users affected by the attribute revocation. In our case, the revocation process will follow three main steps:

First, the authority that will revokes an attribute i, will generate a new $\alpha_i, \beta_i \in Z_p$ and updates his private and public key, then generates a new attribute updated key:

$$g_1^{\frac{\alpha_i}{Z_u}} H(ID_u)^{\frac{\beta_i}{Z_u}},$$

for each non-revoked user U and encrypt the updated key parts using the user public key, then sends the keys to their respective users. The authority then sends the cipher text-update key:

$$CK = c\{t_1, t_2\},$$

(with $t_1 = \alpha_{old} - \alpha_{new}$, and $t_2 = \beta_{old} - \beta_{new}$) and send them to the server. Finally, for each file containing the revoked attribute, the server will then compute:

$$
\begin{aligned}
D_{new,1} &= \frac{D_1}{(D_1)^{t1}} = \frac{g_1^{\lambda_{\rho(x)}} g_1^{\alpha_{old} r_x}}{g_1^{(\alpha_{old}-\alpha_{new}) r_x}}, \\
&= g_1^{\lambda_{\rho(x)}} g_1^{\alpha_{new} r_x}, \qquad (7)
\end{aligned}
$$

and:

$$
\begin{aligned}
D_{new,3} &= \frac{D_3}{(D_3)^{t2}} = \frac{g_1^{\beta_{old} r_x} g_1^{w_{\rho(x)}}}{g_1^{(\beta_{old}-\beta_{new}) r_x}}, \\
&= g_1^{\beta_{new} r_x} g_1^{w_{\rho(x)}}. \qquad (8)
\end{aligned}
$$

2) File access revocation: When the owner needs to revoke the access right of a set of users to a data file, he first chooses a new encryption key s_{new}, then defines a new access policy matrix $\mathbf{M}'$ and computes:

$$
\begin{aligned}
D_{x,1} &= g_1^{\lambda_{\rho(x)}} g_1^{\alpha_{\rho(x)} r_x}, \\
D_{x,2} &= g_1^{r_x}, \\
D_{x,3} &= g_1^{\beta_{\rho(x)} r_x} g_1^{w_{\rho(x)}}.
\end{aligned}
$$

for each attribute in the access structure. Finally, he generates a proxy re-encryption key s_{proxy} and sends $D = \{\forall x, \{D_{x,1}, D_{x,2}, D_{x,3}\}; (\mathbf{M}', \rho'), s_{proxy}\}$, to the cloud server.

The cloud server then uses the proxy re-encryption key s_{proxy} to re-encrypt the data then updates the file's access structure.

V. Analysis

A. Security analysis

1) Fine-grained access control: Each user U_u receives a set of attributes describing the privileges assigned to him by the multiple authorities. Since the data is encrypted under an access policy, only a user who has the required attributes can find $\Sigma_x \lambda_x = s$ and $\Sigma_x w_x = 0$ and recover the encryption key $e(g_1, g_1)^s$. Moreover, our scheme only discloses the decryption keys to the authorized users, making the cloud server and the unauthorized users unable to decrypt the data. For this reason, our method can help the owner achieve the fine-grained access control of the cloud data.

2) Collusion resistance: We achieve fully collusion by linking the users key with his ID, which means that even if multiples unauthorized users try to collude, they wont be able to cancel the element $e(g_1, H(ID))^{w_x}$ and recover the encryption key. Since the decryption is partially ensured by the proxy server, we modified the result of each attribute operation using the ID generated by the AA_s, which makes the proxy server unable to recover the decryption key even if he collude with multiple unauthorized users. Moreover, since the cloud server does not hold any additional information than the unauthorized users and the proxy server, the collusion between those parties and the cloud will not disclose any information on the encryption key.

In case of several corrupt AA, and being given that they do not control the attributes necessary for the decryption, then they will not be able to recover the decryption key. Thus, the proposed system maintains a fine-grained access control and fully collusion resistant.

3) User access privilege confidentiality: In MABE-SOD we disclose parts of the users key to the proxy server when conducting the decryption operation, but since the keys are altered using the IDs provided by the different authorities the proxy server wont gain any information by using this key parts. Moreover, we do not disclose any information about the users keys to any parties other than the user and the proxy

TABLE I: Comprehensive MABE-SOD with DACC scheme

Scheme	Authority	Computation		Revocation Message	Revocation Security		Revocation Controller	Cipher-tex Updater
		Encryption	Decryption		Backward	Forward		
DACC [9]	Multiple	$O(I_c)$	$O(I_u)$	$O(n_{c,x} * n_{non,x})$	Yes	No	Owner	Owner
MABE-SOD	Multiple	$O(I_c)$	$O(I_u)$	$O(n_{non,x})$	Yes	Yes	AA	Server

server, which protects the users privacy against the different parties of the system.

4) Backward And Forward Security: In MABE-SOD, we are able to achieve backward security by updating the non-revoked users' secret keys, which prevents the revoked users from decrypting the new cipher-texts that are encrypted by the new public attribute keys. Moreover, we are able to achieve Forward Security by updating the cipher-text which makes sure that the newly joined users can still access the data published before they join the system, when their attributes satisfy the access policy associated with the cipher-texts.

B. Comprehensive analysis

Let I_u be the total number of attributes of a user and I_c be the total number of attributes in the cipher text. For a revoked attribute x, let $non_{c,x}$ be the number of the non-revoked user who hold the revoked attributes and $n_{c,x}$ the number of cipher-text containing the revoked attribute.

From Table 1 we can notice that MABE-SOD can achieve a lower revocation overhead on the data owner, since the attribute revocation is conducted by the attributes authorities and the re-encryption operation is conducted by the server. The owner is only responsible of generating a proxy re-encryption key and a new access policy for the revoked data, we can also see that, in contrast to DACC, our scheme is able to achieve both Forward and backward security. Moreover, in even though both the complexity of the encryption and the decryption operations, MABE-SOD and DACC have the same complexity, the majority of the decryption overhead of our scheme is outsourced to the proxy server, this largely lowers the user's decryption overhead.

C. Performance Analysis

In the experimental evaluation, we change the size of the access structures by increasing the number of used attributes; this enables us to test the speed of the encryption, key generation and decryption operation. The whole experiment system is implemented by Java on a Windows 7 machine with Core 2 duo running at 2.9 GHz.

The figure 3 shows that we were able to achieve a similar encryption efficiency to the one achieved by DACC since we use the same basic operation. Although we can see a slight increase in the encryption overhead of MABE-SOD. That increase is negligible when compared to the decrease of the decryption overhead achieved by our scheme as shown in figure 2.

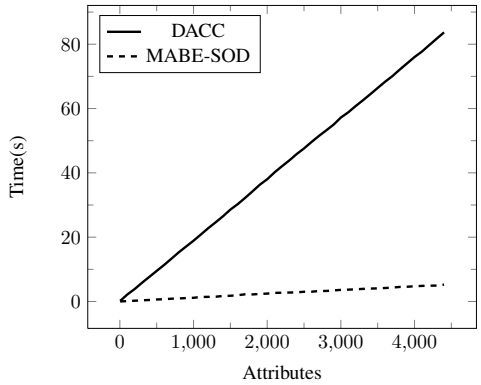

Fig. 2: Decryption overhead in DACC and MABE-SOD schemes

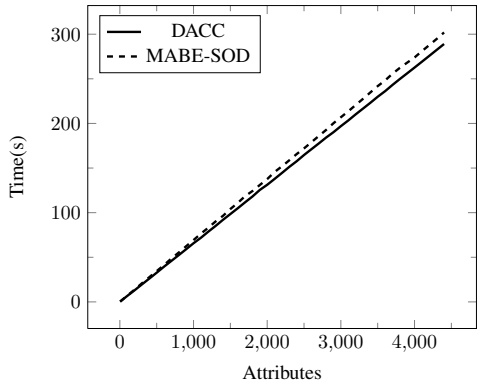

Fig. 3: Encryption overhead in DACC and MABE-SOD schemes

In both our and DACC schemes, the decryption overhead is linear to the number of attributes. But in our case, for each attribute, the user is only needed to perform one exponentiation $(c'_{att_i})^{Z_{att_i}}$ which has noticeably reduced the decryption overhead over the users.

VI. Conclusion

In this paper, we presented an efficient secure and privacy preserving data access control scheme for multi-authority personal health record systems. We were able to achieve a lower decryption overhead than DACC by outsourced most of the computation to the proxy server, we also proposed an efficient revocation process that is able to achieve both forward and backward security. In the future works we will try to include a delegation and data integrity verification in our scheme.

REFERENCES

[1] N. Sultan, "Cloud computing for education: A new dawn?" *International Journal of Information Management*, vol. 30, no. 2, pp. 109–116, 2010.

[2] J. Mounet, "Le livre blanc du cloud Computing, Tout ce que vous devez savoir sur linformatique dans le nuage, syntech informatique, paris, 2010. www. syntec-informatique. fr."

[3] J. Liu, X. Huang, and J. K. Liu, "Secure sharing of personal health records in cloud computing: ciphertext-policy attribute-based signcryption," *Future Generation Computer Systems*, vol. 52, pp. 67–76, 2015.

[4] N. Sultan, "Making use of cloud computing for healthcare provision: Opportunities and challenges," *International Journal of Information Management*, vol. 34, no. 2, pp. 177–184, 2014.

[5] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE symposium on security and privacy (SP'07)*. IEEE, 2007, pp. 321–334.

[6] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Infocom, 2010 proceedings IEEE*. Ieee, 2010, pp. 1–9.

[7] X. Dong, J. Yu, Y. Luo, Y. Chen, G. Xue, and M. Li, "Achieving an effective, scalable and privacy-preserving data sharing service in cloud computing," *computers & security*, vol. 42, pp. 151–164, 2014.

[8] M. Li, S. Yu, K. Ren, and W. Lou, "Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings," pp. 89–106, 2010.

[9] S. Ruj, A. Nayak, and I. Stojmenovic, "Dacc: Distributed access control in clouds," pp. 91–98, 2011.

[10] A. Sahai and B. Waters, "Fuzzy identity-based encryption," Springer, pp. 457–473, 2005.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*. Acm, 2006, pp. 89–98.

[12] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 735–737.

[13] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. ACM, 2010, pp. 261–270.

[14] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 2007, pp. 195–203.

[15] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts." in *USENIX Security Symposium*, vol. 2011, no. 3, 2011.

[16] X. Dong, J. Yu, Y. Luo, Y. Chen, G. Xue, and M. Li, "P2e: Privacy-preserving and effective cloud data sharing service," in *2013 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2013, pp. 689–694.

[17] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *International Workshop on Public Key Cryptography*, 2011, pp. 53–70.

[18] A. Beimel and . , *Secure schemes for secret sharing and key distribution*. Technion-Israel Institute of technology, Faculty of computer science, 1996.