Annals of Telecommunications
https://doi.org/10.1007/s12243-018-00702-6

Fine-grained multi-authority access control in IoT-enabled mHealth

Qi Li[1] · Hongbo Zhu[2] · Jinbo Xiong[3] · Ruo Mo[4] · Zuobin Ying[5] · Huaqun Wang[1]

Received: 10 July 2018 / Accepted: 28 December 2018

Abstract

With the popularity of Internet of Things (IoT) and cloud computing technologies, mobile healthcare (mHealth) can offer remote, accurate, and effective medical services for patients according to their personal health records (PHRs). However, data security and efficient access of the PHR should be addressed. Attribute-based encryption (ABE) is regarded as a well-received cryptographic mechanism to simultaneously realize fine-grained access control and data confidentiality in mHealth. Nevertheless, existing works are either constructed in the single-authority setting which may be a performance bottleneck, or lack of efficient user decryption. In this paper, we propose SEMAAC, a secure and efficient multi-authority access control system for IoT-enabled mHealth. In SEMAAC, there are multiple independently worked attribute authorities (AAs). A new entity could be an AA without re-building the system. To reduce the user decryption overhead, most decryption is executed in cloud server, which whereafter returns a partial decryption ciphertext (PDC). The AAs can help the user to check if the PDC is correctly computed. Additionally, a restricted user can delegate his/her key to someone to outsource the decryption and check the returned result, without exposing the plaintext PHR file. The proposed SEMAAC is proved to be adaptively secure in the standard model. The numerical analysis and extensive experiments illustrate the efficiency and advantage of our scheme.

Keywords Mobile healthcare · Attribute-based encryption · Access control · Multiple authorities · Efficient decryption

1 Introduction

As a promising paradigm, mobile healthcare (mHealth) [1] facilitates the patients to get accurate and effective medical services by combining various technologies, e.g., Internet of Things (IoT) [2–4], Wireless Body Area Network (WBAN) [5], Cloud Computing [6, 7]. In an mHealth system, a patient can monitor and collect the real-time physiological information (such as blood pressure, body temperature) by wearable or implantable sensors. Such data is aggregated to personal health record (PHR) at smart devices and then hosted to the cloud to save the poor storage capacity of smart devices. However, once the PHR data is hosted to the cloud, it is difficult to ensure that the PHR will be accessed by authorized doctors or any other professional researchers.

To tackle this issue, a novel technique named attribute-based encryption (ABE) [8] is adopted, which provides a scalable approach to enhance fine-grained access control over encrypted data. ABE paradigms can be divided into two types: Key-policy ABE (KP-ABE) [9] and Ciphertext-policy ABE (CP-ABE) [10, 11]. In the former type, the user's decryption key is annotated by an access structure and the plaintext is encrypted under some attributes. In the latter type, the locations of attributes and access structure are reversed. In both types of ABE schemes, the decryption succeeds only if the attributes match the access structure. Soon after, multi-authority ABE (MA-ABE) [12] is introduced to capture the application where multiple attribute authorities (AAs) coexist. Each AA is responsible for managing a partial attribute universe and generating decryption keys for users. Although the traditional MA-ABE schemes [12–16] can be directly employed as a basis to build fine-grained access control schemes for mHealth applications, it is still worth considering the features of *decentralizing authorities*, *efficient decryption*, *delegation* and *adaptive security*.

Decentralizing Authorities. In most existing MA-ABE schemes, either a central authority is needed to set the

✉ Qi Li
liqics@njupt.edu.cn

Extended author information available on the last page of the article.

Published online: 09 January 2019

parameters for the AAs [12], or each AA has to cooperate with the other AAs [13] or CAs [14, 17] to initialize the system. On the one hand, if the number of AAs is set too small, once a new AA joins in the system, it is required to reconstruct the system. On the other hand, if the number of AAs is set too large, it will cause unnecessary waste of resources.

Efficient Decryption. The decryption cost of ABE schemes is usually in proportion to the scale of attribute set employed in decryption. However, the smart devices with constrained computation resource (e.g., mobile phone, smart watch) cannot handle such burdensome decryption computation. Green et al. [18] proposed an outsourcing technique to reduce the decryption cost for user by offloading heavy computation cost to the semi-trusted cloud. The user spends only one exponential operation to decrypt the partial decryption ciphertext (PDC) returned by the cloud. Later, Lai et al. [19] introduced a verifiable method to guarantee the correctness of PDC. However, there is only one AA in these two schemes.

Delegation. Suppose Alice is too busy or cannot connect the network, she needs someone to connect the cloud server and verify the correctness of PDC without exposing the sensitive PHR. Basing on [18], Li et al. [20] presented a delegation technique to allow the unauthorized user to check the PDC. To realize delegation, it has to encrypt an additional random message as well as the normal plaintext. The computation cost and the size of ciphertext are about twice of the underlying ABE scheme. Ning et al. [21] proposed an efficient auditable CP-ABE scheme. However, the scheme [21] was constructed in the single-authority setting and the delegation problem was not studied.

Adaptive security. The security of most current MA-ABE schemes is proved in the weaker selective security model [8, 9], which requires the adversary to declare the challenge access structure (in CP-ABE) or the challenge attribute set (in KP-ABE) before receiving the public parameters. On the contrary, in the adaptive security model [22–24], such restriction does not exist. Thus, higher level of security is more desirable in real applications.

To address the issues aforementioned, we introduce SEMAAC, a secure and efficient multi-authority access control system for IoT-enabled mHealth, which can simultaneously support the attractive features of decentralizing authorities, efficient decryption, delegation and adaptive security. In addition, any monotonic access structure which can be described by linear secret sharing scheme (LSSS) is supported. The formal security proof and extensive experiments demonstrate the security, practicability, and efficiency of SEMAAC. The main contributions are summarized below.

Decentralizing Authorities. In SEMAAC, each entity could be an AA by publishing the public parameters as well as the attribute universe. There is no requirement for the AA to cooperate with the others. Moreover, no CA is needed to govern the AAs.

Efficient Decryption. The cloud server receives decryption requirement from the user and returns a PDC. The user can efficiently check the correctness of the returned PDC. He then recovers the plaintext PHR by executing only one exponential operation.

Delegation. Applying a helpful delegation approach, the restrained user can ask someone else to connect the cloud server and verify the returned result, without exposing the plaintext content.

Adaptive security. The adaptive security of SEMAAC is proved under subgroup decision assumptions in the standard model.

2 Related work

2.1 Tradition ABE schemes

Chase [12] introduced the first MA-ABE scheme where the user's attributes are managed by different authorities. From then on, a series of MA-ABE schemes have been proposed in [13, 15, 16, 25]. The security of above schemes is proved in the selective model. To remove the restriction on security model, Lewko et al. [22] constructed an adaptively secure single-authority CP-ABE scheme on composite order groups as well as a KP-ABE scheme. Afterwards, Lewko et al. [23] and Liu et al. [14] studied the adaptive security of multi-authority CP-ABE in the random oracle model and the standard model, respectively.

Aiming to reduce the user's complicated decryption overhead, Green et al. [18] provided a decryption outsourcing approach for ABE, where most of the decryption operation is outsourced to a third-party, such as the cloud server. However, the correctness of PDC returned by the third-party is not guaranteed. Lai et al. [19] proposed a verifiable approach for ABE to check if PDC is correctly computed. Basing on [18], Qin et al. [26] proposed an efficient and verifiable ABE scheme, where the message is first encrypted under a symmetric key which will be encrypted by ABE. The length of ciphertext and the computation cost is almost half of that in [19]. Li et al. [20] presented a verifiable approach in CP-ABE for both authorized and unauthorized users. In the encryption phase, a message is encrypted for authorized user and another random massage is encrypted for the unauthorized user. The length of ciphertext is about twice of that in [18, 26].

2.2 ABE-based access control systems

Various techniques [27–35] have been adopted to achieve data security and privacy preserving in real applications. As a promising cryptography, ABE is regarded as an important basis to construct fine-grained access control systems. In [36], Yu et al. demonstrated how to construct fine-grained access control systems on basis of ABE in cloud computing. Li et al. [37] introduced a multi-authority PHR access control scheme for cloud storage which supports outsourced decryption and adaptively security. However, the correctness of returned PDC cannot be verified. Hahn et al. [1] presented a AND-gate data sharing scheme in mobile health networks. Yang et al. [38] studied a lightweight access control system for the healthcare application in IoT. Li et al. [39] studied the outsourcing techniques of both decryption and key-issuing. Wang et al. [40] and Li et al. [41] considered directly revocation and accountability in cloud, respectively. Ning et al. [21] proposed an auditable outsourced and selectively secure CP-ABE scheme in cloud computing. Zhang et al. [24] constructed an adaptively secure access control system in smart health without optimizing the decryption algorithm. In these schemes [1, 21, 24, 36, 38], only one authority is supported.

In Table 1, we compare some characteristics between current MA-ABE systems and ours. From the feature comparison, our SEMAAC is the only one which simultaneously achieves efficient decryption, delegation and adaptive security in the multi-authority setting.

3 Preliminaries

3.1 Linear secret sharing schemes (LSSS)

Definition 1 LSSS [42, 43]: Let $\mathbb{P}$, p and $M \in \mathbb{Z}_p^{\ell \times n}$ denote a set of participants, a chosen prime and a matrix, respectively. For all $i = 1, \ldots, \ell$, a function ρ maps the ith row of M to a participant. ($i.e.$ $\rho \in \mathcal{F}([\ell] \rightarrow \mathbb{P})$). A secret sharing scheme Π is linear if

1. The shares for each participant make a vector over $\mathbb{Z}_p$.
2. There exists a share-generating matrix M which can make shares for Π. In order to generate the shares of a secret $s \in \mathbb{Z}_p$, we choose the column vector $\overrightarrow{v} = (s, d_2, \ldots, d_n)^{\top}$, where $d_2, \ldots, d_n$ are randomly chosen from $\mathbb{Z}_p$, then $M\overrightarrow{v}$ is the ℓ shares of s according to Π. The share $(M\overrightarrow{v})_i$ belongs to the participant $\rho(i)$.

As shown in [42], LSSS has the linear reconstruction property: Let (M, ρ) denote the access structure $\mathbb{M}$. S refers to an authorized set. Let $I = \{i : \rho(i) \in S\}$ be the index set of rows which are linked with the attributes in S. There must be some computable constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ which satisfy that if the shares $\{\lambda_i = (M\overrightarrow{v})_i\}$ of s are valid, then we have $\sum_{i \in I} \omega_i \lambda_i = s$. However, if S is not authorized, there are no such constants.

3.2 Composite order bilinear group

A group generator $\mathcal{G}$ takes in a security parameter λ and outputs the terms $(\mathbb{G}, \mathbb{G}_1, p_1, p_2, p_3, e)$, where p_1, p_2, p_3 are three different primes, the order of cyclic groups $\mathbb{G}$ and

Table 1 A comparison of multi-authority, security model, and type

Systems	Type	MA	AS	SM	DO	Verifiability	Delegation
[12, 13]	KP	Yes	Threshold	Selective	No	No	No
[23]	CP	Yes	LSSS	Adaptive	No	No	No
[14]	CP	Yes	LSSS	Adaptive	No	No	No
[15]	KP	Yes	LSSS	Selective	No	No	No
[18]	KP,CP	No	LSSS	Selective	Yes	No	No
[19]	CP	No	LSSS	Selective	Yes	Yes	No
[20]	CP	No	LSSS	Selective	Yes	Yes	Yes
[37]	CP	Yes	LSSS	Adaptive	Yes	No	No
[38]	CP	No	LSSS	Adaptive	Yes	Yes	Yes
[21]	CP	No	LSSS	Selective	Yes	Yes	No
SEMAAC	KP	Yes	LSSS	Adaptive	Yes	Yes	Yes

MA multiple authorities, *AS* access structure, *SM* security model, *DO* decryption outsourcing

$\mathbb{G}_1$ is $N = p_1p_2p_3$, and the map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ has such properties:

1. Bilinearity: $\forall \omega, \theta \in \mathbb{G}$ and $a, b \in \mathbb{Z}_N$, we have $e(\omega^a, \theta^b) = e(\omega, \theta)^{ab}$.
2. Non-degeneracy:$\exists\ \hbar \in \mathbb{G}$ such that the order of $e(\hbar, \hbar)$ in $\mathbb{G}_1$ is N.

3.3 Complexity assumption

Let $\mathbb{GD}$ denote the terms $(\mathbb{G}, \mathbb{G}_1, N = p_1p_2p_3, e)$ generated by a group generator $\mathcal{G}$. Denote $\mathbb{G}_{p_i}$ as the subgroup of order p_i in $\mathbb{G}$. Suppose group $\mathbb{G}_{p_1}$, $\mathbb{G}_{p_2}$ and $\mathbb{G}_{p_3}$ are generated by g_1, g_2 and g_3, respectively. Then, each element $\partial \in \mathbb{G}$ can be written as $g_1^x g_2^y g_3^z$ for some exponents $x, y, z \in \mathbb{Z}_N$. For instance, g_1^x refers to the "$\mathbb{G}_{p_1}$ part of ∂".

Assumption 1 [22]. *Given $\mathcal{G}$, we can define the following distributions:*

$$g \xleftarrow{R} \mathbb{G}_{p_1}, \aleph_3 \xleftarrow{R} \mathbb{G}_{p_3},$$
$$D = (\mathbb{GD}, g, \aleph_3),$$
$$\partial_1 \xleftarrow{R} \mathbb{G}_{p_1p_2}, \partial_2 \xleftarrow{R} \mathbb{G}_{p_1}.$$

The advantage with which an algorithm can distinguish ∂_1 from ∂_2 is defined as $Adv1_{\mathcal{G},\mathcal{A}} = |Pr[\mathcal{A}(D, \partial_1) = 1] - Pr[\mathcal{A}(D, \partial_2) = 1]|$.

Definition 2 We say that $\mathcal{G}$ satisfies Assumption 1 if the advantage $Adv1_{\mathcal{G},\mathcal{A}}$ is negligible for any polynomial time attacker.

Assumption 2 [22]. *Given $\mathcal{G}$, we can define the following distributions:*

$$g, \aleph_1 \xleftarrow{R} \mathbb{G}_{p_1}, \aleph_2, \Re_2 \xleftarrow{R} \mathbb{G}_{p_2}, \aleph_3, \Re_3 \xleftarrow{R} \mathbb{G}_{p_3},$$
$$D = (\mathbb{GD}, g, \aleph_1\aleph_2, \aleph_3, \Re_2\Re_3),$$
$$\partial_1 \xleftarrow{R} \mathbb{G}, \partial_2 \xleftarrow{R} \mathbb{G}_{p_1p_3}.$$

The advantage with which an algorithm can distinguish ∂_1 from ∂_2 is defined as $Adv2_{\mathcal{G},\mathcal{A}} = |Pr[\mathcal{A}(D, \partial_1) = 1] - Pr[\mathcal{A}(D, \partial_2) = 1]|$.

Definition 3 We say that $\mathcal{G}$ satisfies Assumption 2 if the advantage $Adv2_{\mathcal{G},\mathcal{A}}$ is negligible for any polynomial time attacker.

Assumption 3 [22]. *Given $\mathcal{G}$, we can define the following distributions:*

$$\sigma, s \xleftarrow{R} \mathbb{Z}_N, g \xleftarrow{R} \mathbb{G}_{p_1}, \aleph_3 \xleftarrow{R} \mathbb{G}_{p_3},$$
$$\aleph_2, \Re_2, \Im_2 \xleftarrow{R} \mathbb{G}_{p_2},$$
$$D = (\mathbb{GD}, g, g^{\sigma}\aleph_2, \aleph_3, g^s\Re_2, \Im_2),$$
$$\partial_1 = e(g, g)^{\sigma s}, \partial_2 \xleftarrow{R} \mathbb{G}_1.$$

The advantage with which an algorithm can distinguish ∂_1 from ∂_2 is defined as $Adv3_{\mathcal{G},\mathcal{A}} = |Pr[\mathcal{A}(D, \partial_1) = 1] - Pr[\mathcal{A}(D, \partial_2) = 1]|$.

Definition 4 We say that $\mathcal{G}$ satisfies Assumption 3 if the advantage $Adv3_{\mathcal{G},\mathcal{A}}$ is negligible for any polynomial time attacker.

4 System architecture, formal definition, and security models

4.1 System architecture

As in Fig. 1, the following four entities, multiple AAs, PHR Owner (PO), PHR User (PU), and Cloud Service Provider (CSP), are involved in SEMAAC.

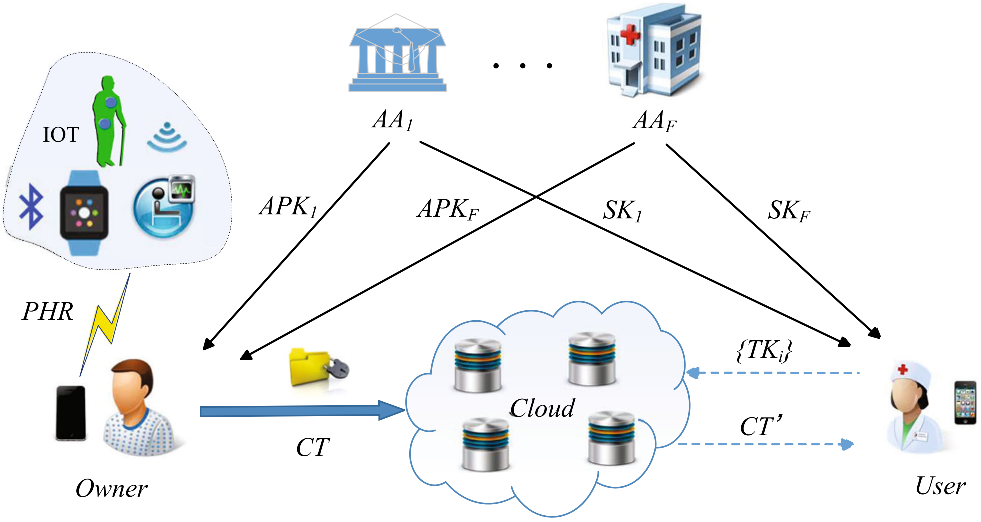


Fig. 1 Architecture of SEMAAC

- **Multiple AAs**: Multiple AAs can independently publish their public parameters and corresponding partial attribute universe. They are also in charge of generating keys and providing verification service for PU.
- **PO**: PO collects the health information by various wearable or implantable sensors, aggregates the PHR file by smart devices, and encrypts the PHR file under some attributes before uploading it to CSP. PO can be an elder or a patient, even a hospital or a sanitation department which manages PHRs for its patients.
- **CSP**: CSP offers the storage service for PO. It also answers the outsourced decryption request from PU and responses by the partial decryption ciphertext.
- **PU**: PU can be a medical researcher or a doctor who is associates with some access policies. PU tells CSP which ciphertext he wants to access and calls CSP to execute the outsourced decryption operation. Then, he will upload the partial decryption ciphertext to the corresponding AAs and call for the verification service.

We now briefly introduce the proposed SEMAAC system.

- **System Initialization**: In the initialization phase, each AA can independently publish the partial attribute universe and generate its public parameters and the corresponding master secret key.
- **PHR Outsourcing**: Before outsourcing the PHR to CSP, PO has to choose some attributes to encrypt the PHR file.
- **Authorization**: The AA_i could grant the access rights to PU according to his access policy over the attributes which are governed by AA_i. Generally speaking, a PU may be issued multiple access policies from different AAs.
- **PHR Access**: By uploading the transformed private key ($\{TK_i\}$), PU could choose a PHR ciphertext and call CSP to pre-decrypt it. However, only if the attributes in ciphertext match the access policies in $\{TK_i\}$, the decryption would success. Besides, the PU can check if the decryption is correctly executed.

In SEMAAC, the AAs are assumed to be fully trusted. The CSP is honest-but-curious, which honestly executes its tasks but attempts to recover the content of encrypted PHR files. The malicious PU may collude with others to access the file that none of them is authorized.

4.2 Formal definition

SEMAAC consists of the following ten algorithms:

GlobalSetup $(\lambda) \rightarrow (GLPK)$: The GlobalSetup algorithm takes in a security parameter λ and outputs the global system public parameters $GLPK$.

UserSetup $(GID, GLPK) \rightarrow (PK_{GID}, DK_{GID})$: A user is associated with a unique fixed global identifier (GID) when he joins in the system. The UserSetup algorithm takes in GID and $GLPK$. It then outputs the user's public key PK_{GID} and its decryption key DK_{GID}.

AASetup $(k, U_k, GLPK) \rightarrow (APK_k, AMK_k)$: The AASetup algorithm takes in the index k of AA_k, its partial attribute universe U_k and $GLPK$. It outputs its public parameter APK_k and the corresponding master secret key AMK_k of AA_k. We denote $\mathbb{K} = \{1, 2, \ldots, K\}$ and $U = \bigcup_{k=1}^{K} U_k$ as the index set of total AAs and the system attribute universe, respectively. Especially, for all $i \neq j \in \mathbb{K}$, we have $U_i \bigcap U_j = \emptyset$.

Encrypt $(S, GLPK, \bigcup APK_k) \rightarrow (CT, SEK)$: The Encrypt algorithm takes in a set of attributes S, $GLPK$ and the set of relevant public parameters $\bigcup APK_k$. It then outputs the symmetric encryption key (SEK) and a ciphertext CT.

AAKeyGen $(k, GID, \mathbb{M}_{GID,k}, GLPK, PK_{GID}, AMK_k) \rightarrow (UASK_{\mathbb{M}_{GID,k}})$: The AAKeyGen algorithm takes in the index k of AA_k, GID, an access structure $\mathbb{M}_{GID,k}$, $GLPK$, PK_{GID}, and AMK_k. It then outputs the user's partial private key $UASK_{\mathbb{M}_{GID,k}}$. We denote the user's private key as $SK_{GID} = \bigcup UASK_{\mathbb{M}_{GID,k}}$.

TKKeyGen $(SK_{GID}, DK_{GID}, GLPK) \rightarrow (TK_{GID})$: The TKKeyGen algorithm takes in SK_{GID}, DK_{GID}, and $GLPK$. It outputs the transformation key TK_{GID}.

Pre-decrypt $(GLPK, TK_{GID}, CT) \rightarrow (PDC)$: The Pre-decrypt algorithm takes in $GLPK$, TK_{GID} and CT. If S in CT matches the access structures in TK_{GID}, it outputs a PDC. Otherwise, it outputs $\perp$.

Decrypt $(PDC, DK_{GID}) \rightarrow (SEK)$: The Decrypt algorithm takes in PDC and DK_{GID}. If **Verify** $(PK_{GID}, \bigcup AMK_k, CT, PDC) \rightarrow 1$, it outputs SEK. Otherwise, it aborts.

Verify $(PK_{GID}, \bigcup AMK_k, CT, PDC) \rightarrow 1$ or 0: The Verify algorithm takes is PK_{GID}, $\bigcup AMK_k$, CT and PDC. If PDC is correctly computed, it outputs 1. Otherwise, it outputs 0.

Delegate $(PK_{GID}, TK_{GID}, GID') \rightarrow (PK_{GID}, TK_{GID})$: The Delegate algorithm takes in PK_{GID}, TK_{GID} and a new GID', it outputs the delegated key $(PK_{GID'}, TK_{GID'})$.

4.3 Security game for SEMAAC

The same as the corruption model in [12, 23], the adversary is assumed to corrupt the AAs only statically. We now introduce the CPA security model of SEMAAC by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$:

Setup. $\mathcal{A}$ first declares a index set $\mathbb{K}_c$ which refers to the set of corrupted AAs. $\mathbb{K}_{uc} = \mathbb{K} \backslash \mathbb{K}_c$ refers to the index

set of uncorrupted AAs. $\mathcal{B}$ runs **GlobalSetup** to generate $GLPK$. Especially, in **AASetup** phase, for $k \in \mathbb{K}_c$, $\mathcal{B}$ generates APK_k and AMK_k. It then gives them to $\mathcal{A}$. For $k \in \mathbb{K}_{uc}$, only APK_k is given to $\mathcal{A}$.
Phase 1. $\mathcal{B}$ initializes an empty set H and an empty table L. $\mathcal{A}$ can make the secret key queries in the following way:

(1) Private key query. Once receiving the private key query on $\mathbb{M}_{GID} = \bigcup \mathbb{M}_{GID,k}$, $\mathcal{B}$ first runs **UserSetup** to obtain (PK_{GID}, DK_{GID}). It then runs **AAKeyGen** to get the user's private key SK_{GID}. Finally, it sets $H = H \cup \mathbb{M}_{GID}$ and gives SK_{GID} to $\mathcal{A}$.
(2) Transformation key query. Once receiving the transformation key query on the access structure $\mathbb{M}_{GID}$ of a user with GID, $\mathcal{B}$ first checks if the entry $(\mathbb{M}_{GID}, SK_{GID}, TK_{GID}, DK_{GID})$ in L. If so, it returns TK_{GID} to $\mathcal{A}$. Otherwise, it runs **AAKeyGen** and **TKKeyGen**. It then stores $(\mathbb{M}_{GID}, SK_{GID}, TK_{GID}, DK_{GID})$ in T and sends TK_{GID} to $\mathcal{A}$.

$\mathcal{A}$ can answer the private key queries to the corrupted AAs itself, since it knows the APK_k and AMK_k for $k \in \mathbb{K}_c$.
Challenge Phase. $\mathcal{A}$ submits a challenge attribute set S^*. The only restriction is that S^* cannot satisfy any $\mathbb{M}_{GID} \in H$. $\mathcal{B}$ runs **Encrypt** to get (CT^*, SEK^*). $\mathcal{B}$ then random picks $\mu \in \{0, 1\}$. If $\mu = 0$, (CT^*, SEK^*) is sent to $\mathcal{A}$. Otherwise, (CT^*, R^*) is transmitted to $\mathcal{A}$, where R^* is randomly chosen in the encapsulated key space.
Phase 2. $\mathcal{A}$ acts as in Phase 1. The only restriction is that, for all $k \in \mathbb{K}_{uc}$, at least a set $S_k^* \subseteq S^*$ cannot match $\mathbb{M}_{GID,k}$.
Guess. $\mathcal{A}$ guesses μ' on μ.

The advantage of $\mathcal{A}$ in the above game is $Pr[\mu' = \mu] - 1/2$.

Definition 5 SEMAAC is secure against the chosen plaintext attack if all PPT adversaries have at most a negligible advantage in the security game.

4.4 Verifiability

We now introduce the verifiability model of SEMAAC by the following security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$.

Setup. $\mathcal{A}$ and $\mathcal{B}$ act as in the CPA security game.
Phase 1. $\mathcal{A}$ acts as in the CPA security game.
Challenge Phase. Once receiving a challenge attribute set S^* from $\mathcal{A}$, $\mathcal{B}$ runs **Encrypt** to obtain (CT^*, SEK^*) and then gives CT^* to $\mathcal{A}$.
Phase 2. Same as **Phase 1**.
Output. $\mathcal{A}$ outputs an access structure $\mathbb{M}_{GID}^*$ and two partial decryption ciphertext PDC_1^* and PDC_2^* for CT^*. Assume that the tuple $(\mathbb{M}_{GID}, SK_{GID}, TK_{GID}, DK_{GID})$ can be found in L. If not, $\mathcal{B}$ produces by itself. $\mathcal{A}$ wins if the entry (PDC_1^*, PDC_2^*) can pass **Verify**, and **Decrypt** $(PDC_1^*, DK_{GID}) \neq$ **Decrypt** (PDC_2^*, DK_{GID}).

Definition 6 SEMAAC is verifiable if all PPT adversaries have at most a negligible advantage in the above game.

5 Details of SEMAAC

Different from [44], we take the user's GID as the secret to be shared during key generation. Then, the private keys from different AAs are bound with the same GID. Thus, the collusion-resistance is enhanced. In addition, similar to the key encapsulation mechanism (KEM) [21, 26], the PHR file in SEMAAC is encrypted under a symmetric encryption key (SEK) which is further encrypted under some attributes. We now give the detailed construction of SEMAAC.

System Initialization The **GlobalSetup** algorithm generates the global system public parameters, basing on which the PU's GID-related keys are created by running the **UserSetup** algorithm and AAs' public parameters and corresponding master secret keys are produced by running the **AASetup** algorithm.

- **GlobalSetup**: This algorithm calls the group generator and outputs a bilinear group $(N, \mathbb{G}, \mathbb{G}_1, e)$, where $\mathbb{G}$ and $\mathbb{G}_1$ are multiplicative cyclic groups of composite order $N = p_1 p_2 p_3$ (3 different primes). X_3 refers to a generator of $\mathbb{G}_{p_3}$. Randomly pick $g, h \in \mathbb{G}_{p_1}$. The global system public parameter $GLPK$ is: $GLPK = (N, \mathbb{G}, \mathbb{G}_1, e, g, h, X_3)$.
- **UserSetup**: Each PU will be associated with a $GID = gid \in \mathbb{Z}_N$. Randomly select $z_{GID} \in \mathbb{Z}_N$ and compute $PK_{GID} = g^{z_{GID}}$. The user decryption key is set to be $DK_{GID} = z_{GID}$.
- **AASetup**: For each AA_k, randomly pick $\sigma_k, \gamma_k \in \mathbb{Z}_N$. Then, calculate $APK_{k,1} = e(g, g)^{\sigma_k}$ and $APK_{k,2} = g^{\gamma_k}$. For each attribute $ATT_{k,j} \in U_k$, Randomly select $t_{k,j} \in \mathbb{Z}_N$ and calculate $T_{k,j} = g^{t_{k,j}}$. Finally, AA_k publishes its public parameter as $APK_k = (APK_{k,1}, APK_{k,2}, T_{k,j} \forall j)$. The master secret key is $AMK_k = (\sigma_k, \gamma_k)$.

PHR Outsourcing By running the **Encrypt** algorithm, PO can specify some attributes to encrypt the PHR file before offloading it to CSP.

- **Encrypt**: Set $S = \bigcup S_k$ where $S_k \subseteq U_k$ and $S_k \neq \emptyset$. Denote I_C as the index set of S_k. Randomly pick $s \in \mathbb{Z}_N$ and compute: $C_0 = (\prod_{k \in I_C} g^{\gamma_k})^s$, $C_1 = g^s$. For each $ATT_{k,j} \in S$, compute $C_{k,j} = T_{k,j}^s$. The ciphertext is $CT = (C_0, C_1, \{C_{k,j} | ATT_{k,j} \in S\})$.

  The symmetric encryption key in KEM mechanism is $SEK = (\prod_{k \in I_C} e(g,g)^{\sigma_k})^s$. The ciphertext of PHR file encrypted under SEK is defined as EN_{PHR}. Finally, EN_{PHR} and CT are uploading to CSP.

Authorization Each AA runs the **AAKeyGen** algorithm and issues user's partial private key according to PU's access policy.

- **AAKeyGen**: Let $\mathbb{M}_{GID,k} = (M_k, \rho)$ be an access policy issued by AA_k, where M_k is an $\ell_k \times n_k$ LSSS matrix. AA_k randomly selects $\delta_2, \ldots, \delta_{n_k}$ and $\sigma_k, \kappa_2, \ldots, \kappa_{n_k}$ from $\mathbb{Z}_N$. It then sets two vectors $\overrightarrow{\delta_k} = (\sigma_k, \delta_2, \ldots, \delta_{n_k})^\top$ and $\overrightarrow{\kappa_k} = (gid, \kappa_2, \ldots, \kappa_{n_k})^\top$. It then calculates $\overrightarrow{\lambda_k} = (\lambda_1, \lambda_2, \ldots, \lambda_{\ell_k})^\top = M_{GID,k}\overrightarrow{\delta_k}$ and $\overrightarrow{\xi_k} = (\xi_1, \xi_2, \ldots, \xi_{\ell_k})^\top = M_{GID,k}\overrightarrow{\kappa_k}$. For each $x_k \in [\ell_k]$, randomly choose $W_{x,k}, V_{x,k} \in \mathbb{G}_{p_3}$ and $r_{x,k} \in \mathbb{Z}_N$. Compute

$$D_{x,k,1} = {PK_{GID}}^{\lambda_{x,k}} h^{\xi_{x,k}\gamma_k} T_{\rho(x,k)}^{r_{x,k}}$$
$$W_{x,k} = g^{z_{GID}\lambda_{x,k}} h^{\xi_{x,k}\gamma_k} T_{\rho(x,k)}^{r_{x,k}} W_{x,k},$$
$$D_{x,k,2} = g^{r_{x,k}} V_{x,k}.$$

  The PU's partial private key from AA_k is $UASK_{\mathbb{M}_{GID,k}} = \{D_{x,k,1}, D_{x,k,2}\}_{x \in \mathbb{M}_{GID,k}}$. The PU's total private key from relevant AAs is set as $SK_{GID} = \bigcup UASK_{\mathbb{M}_{GID,k}}$.

PHR Access In order to reduce the local decryption overhead, PU first runs the **TKKeyGen** algorithm to transform his private key before requesting the oursourced decryption service. CSP then runs the **Pre-decrypt** algorithm and returns a partial decryption ciphertext (PDC). Finally, PU can recover SEK by running the **Decrypt** algorithm and further decrypt the PHR ciphertext if the **Verify** algorithm indicates PDC is correct.

- **TKKeyGen**: Compute $D_3 = h^{gid}$ and set $TK_{GID} = (SK_{GID}, D_3)$.
- **Pre-decrypt**: This algorithm aborts if S in CT does not match the access structures in TK_{GID}. Otherwise, for each S_k, calculate constants $\{c_{k,x} \in \mathbb{Z}_N\}_{x \in I_k}$ such that $\sum_{x \in I_k} c_{k,x} A_{k,x} = (1, 0, \ldots, 0)$, where $M_{k,x}$ denotes the xth row of M_k and $I_k = \{x : \rho(x) \in S_k\}$. Then, compute

$$PDC = \frac{\prod_{k \in I_C} \prod_{x \in I_k} (\frac{e(C_1, D_{x,k,1})}{e(C_{\rho_k(x)}, D_{x,k,2})})^{c_{k,x}}}{e(C_0, D_3)} = (\prod_{k \in I_C} e(g,g)^{z_{GID}\sigma_k})^s$$

- **Decrypt**: If **Verify** $(PK_{GID}, \bigcup AMK_k, CT, PDC) \rightarrow 1$, the user can recover SEK by computing: $SEK = PDC^{1/DK_{GID}} = (\prod_{k \in I_C} e(g,g)^{\sigma_k})^s$. Otherwise, it aborts.
- **Verify**: For each AA_k, this algorithm computes $CV_k = e({PK_{GID}}^{\sigma_k}, C_1) = e(g,g)^{z_{GID}\sigma_k s}$. It then checks whether the equation $\prod_{k \in I_C} CV_k = PDC$ holds. If so, it outputs 1 to indicate that PDC is correctly computed. Otherwise, it outputs 0 to indicate that PDC is wrong.

Delagation Suppose a PU is in trouble to call for the pre-decryption service and verification, he may delegate his partial privilege to someone else without compromising the PHR confidentiality by the following **Delegate** algorithm.

- **Delegate**: Let I_{SK} be the index set of AA_k which is in charge of administrating the attributes in SK_{GID}. For the user GID', PU first chooses $c \in \mathbb{Z}_N$. It calculates $PK_{GID'} = {PK_{GID}}^c = g^{gid \cdot c}$ and $D_3' = {D_3}^c$ For each $k \in I_{DE} \subseteq I_{SK}$ and each $x \in M_k$, it computes $D'_{x,k,1} = {D_{x,k,1}}^c$ and $D'_{x,k,2} = {D_{x,k,2}}^c$. The delegated private key is $SK_{GID'} = \bigcup_{k \in I_{DE}} UASK'_{\mathbb{M}_{GID,k}}$, where $UASK'_{\mathbb{M}_{GID,k}} = \{D'_{x,k,1}, D'_{x,k,2}\}_{x \in \mathbb{M}_{GID,k}}$.

  The delegated transformation key is $TK_{GID'} = (SK_{GID'}, D_3')$.

  The delegated user public key is $PK_{GID'} = g^{gid \cdot c}$.

6 Security proof

The adaptive security of SEMAAC is reduced to the adaptively secure KP-ABE scheme [22]. We denote SEMAAC and the KP-ABE scheme [22] as $\sum_{SEMAAC}$ and $\sum_{KP}$, respectively.

Lemma 1 *If Assumption 1, 2 and 3 hold, the scheme $\sum_{KP}$ is adaptively secure in the standard model.*

Proof The details of formal security proof can be found in [22]. □

Lemma 2 *Suppose $\sum_{KP}$ is adaptively secure, then our $\sum_{SEMAAC}$ is secure in the security scheme defined in* Section 4.

Proof Suppose that an adversary $\mathcal{A}$ can break $\sum_{SEMAAC}$ with a non-negligible advantage, then a simulator $\mathcal{B}$ can be built to break $\sum_{KP}$ with a non-negligible advantage. □

Setup. $\sum_{KP}$ gives simulator $\mathcal{B}$ the public parameter $PK_{KP} = (N, e, \mathbb{G}, \mathbb{G}_1, g, e(g, g)^{\sigma}, T_i = g^{t_i} \forall i \in U_{KP}, X_3)$. $\mathcal{B}$ sets the system attribute universe in $\sum_{SEMAAC}$ as $U = U_{KP} = \bigcup U_k$, where U_k is associated with the corresponding AA_k.

Without loss of generality, $\mathcal{A}$ is assumed to corrupt all AAs but AA_1.

$\mathcal{B}$ randomly chooses $a, \sigma_2, \cdots, \sigma_K, \gamma_1, \cdots, \gamma_K \in \mathbb{Z}_N$. For each $k \in \{2, 3, \cdots, K\}$, $\mathcal{B}$ sets $APK_{k,1} = e(g, g)^{\sigma_k}$ and $APK_{k,2} = g^{\gamma_k}$. For AA_1, $\mathcal{B}$ sets $APK_{1,1} = e(g, g)^{\sigma}$ and $APK_{1,2} = g^{\gamma_1}$. Finally, $\mathcal{B}$ outputs $GLPK = (N, \mathbb{G}, \mathbb{G}_1, e, g, h = g^a, X_3)$.

For each $k \in \{2, 3, \cdots, K\}$, $APK_k = (APK_{k,1}, APK_{k,2}, T_{k,j} \forall j)$ and $AMK_k = (\sigma_k, \gamma_k)$ are sent to $\mathcal{A}$. For AA_1, only APK_1 is transmitted to $\mathcal{A}$.

Phase 1. We note that $\mathcal{A}$ can answer the private key queries corresponding to the corrupted AAs itself. Therefore, we only answer the key query to AA_1. $\mathcal{B}$ initializes an empty set H and an empty table L. $\mathcal{B}$ answers the key queries of AA_1 from $\mathcal{A}$ as follows:

(1) Private key query. Once receiving the private key query of $\mathbb{M}_{GID,1}$, $\mathcal{B}$ sends $\mathbb{M}_{GID,1}$ to $\sum_{KP}$ and gets the private key (D_x^1, D_x^2), where $D_x^1 = g^{\lambda_x} T_{\rho(x)}^{r_x} W_x$ and $D_x^2 = g^{r_x} V_x$. $\mathcal{B}$ runs **UserSetup** to obtain $(PK_{GID} = g^{z_{GID}}, DK_{GID} = z_{GID})$ and sets $\{\xi_{x,k}\}$ as in the real scheme. It then computes $D_{x,k,1} = (D_x^1)^{z_{GID}} \cdot h^{\xi_{x,k}\gamma_k} = g^{z_{GID}\lambda_{x,k}} h^{\xi_{x,k}\gamma_k} T_{\rho(x,k)}^{r_{x,k}} W_{x,k}$, $D_{x,k,2} = (D_x^2)^{z_{GID}} g^{r_{x,k}} V_{x,k}$. Finally, it sets $H = H \cup \mathbb{M}_{GID,1}$ and sends $SK_{GID,1} = \{D_{x,k,1}, D_{x,k,2}\}_{x \in \mathbb{M}_{GID,1}}$ to $\mathcal{A}$.

(2) Transformation key query. $\mathcal{B}$ scans the entry $(\mathbb{M}_{GID,1}, SK_{GID,1}, TK_{GID,1}, DK_{GID})$ in L. If so, it returns such entry $(\mathbb{M}_{GID,1}, SK_{GID,1}, TK_{GID,1}, DK_{GID})$. Otherwise, it sets $SK_{GID,1}$ and DK_{GID} as in the private key query. It then runs **TKKeyGen** to obtain $TK_{GID,1} = (SK_{GID,1}, K_3 = h^{gid})$. Finally, it stores the entry $(\mathbb{M}_{GID,1}, SK_{GID,1}, TK_{GID,1}, DK_{GID})$ in L and sends $TK_{GID,1}$ to $\mathcal{A}$.

Challenge Phase. Once receiving a challenge attribute set S^* from $\mathcal{A}$, $\mathcal{B}$ chooses two equal length messages M_0 and M_1. It then submits (S^*, M_0, M_1) to $\sum_{KP}$ and gets the ciphertext $(C = M_b e(g, g)^{\sigma s}, C_{KP} = g^s, C_i = T_i^s \forall i \in S^*)$. $\mathcal{B}$ sets S_k^* and I_C as in the real scheme and calculates

$$C_0 = (C_{KP})^{\sum_{k \in I_C} \gamma_k} = \left(\prod_{k \in I_C} g^{\gamma_k}\right)^s.$$

$$C_1 = C_{KP}$$

For each $ATT_{k,j} \in S$, $\mathcal{B}$ sets $C_{k,j} = T_{k,j}^s$.

$\mathcal{B}$ randomly flips a coin $\mu_{\mathcal{B}} \in \{0, 1\}$ and sets $SEK^* = C/M_{\mu_{\mathcal{B}}}(\prod_{k \in I_C - 1} e(g^{\sigma_k}, C_{KP})) = \prod_{k \in I_C} e(g, g)^{\sigma_k})^s$. We assume that $S_1 \in S^*$ and $S_1 \neq \emptyset$.

$\mathcal{B}$ gives $CT^* = (C_0, C_1, \{C_{k,j} | ATT_{k,j} \in S\})$ and SEK^* to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ repeats the queries as in Phase 1.

Guess. $\mathcal{A}$ guesses $\mu' \in \{0, 1\}$. If $\mu' = 1$, it means that SEK^* is a random key in the adversary's view, $\mathcal{B}$ then outputs its guess $1 - \mu_{\mathcal{B}}$. If $\mu' = 0$, it means that SEK^* is a legal key in the adversary's view, $\mathcal{B}$ then outputs its guess $\mu_{\mathcal{B}}$.

Since that the distributions of $GLPK$, $\{APK_k\}$, SK_{GID}, DK_{GID}, CT^*, and SEK^* are identical to the real system, if $\mathcal{A}$ can break $\sum_{SEMAAC}$ with a non-negligible advantage, $\mathcal{B}$ can break $\sum_{KP}$ with the same advantage.

Theorem 1 *If Assumptions 1, 2, and 3 are intractable, then our SEMAAC is adaptively secure.*

Proof The security proof follows Lemma 1 and Lemma 2. □

6.1 Verifiability

Theorem 2 *For all PPT adversaries, the advantage is at most negligible in the verifiability security game of SEMAAC.*

Proof Given an adversary $\mathcal{A}$ which breaks the verifiability of SEMAAC, then we can construct a simulator $\mathcal{B}$ to interact with $\mathcal{A}$ as follows:

Setup. $\mathcal{A}$ first declares a index set $\mathbb{F}_c$. $\mathcal{B}$ runs **GlobalSetup** and publishes $GLPK = (N, \mathbb{G}, \mathbb{G}_1, e, g, h, X_3)$. It then runs **AASetup** to produce the key pairs $\{(APK_k, AMK_k)\}$. For $k \in \mathbb{K}_c$, it sends (APK_k, AMK_k) to $\mathcal{A}$. For $k \in \mathbb{K}_{uc}$, only APK_k is sent to $\mathcal{A}$. Similar to the CPA game, we assume that only AA_1 is not corrupted.

Phase 1. $\mathcal{A}$ queries the keys as in the CPA security game. $\mathcal{B}$ can answer the key queries by itself since that it knows $\{AMK_k\}$.

Challenge Phase. $\mathcal{A}$ submits a challenge attribute set S^*. $\mathcal{B}$ runs **Encrypt** and gets (CT^*, SEK^*). It then sends CT^* to $\mathcal{A}$.

Phase 2. Same as **Phase 1**.

Output. $\mathcal{A}$ outputs two PDCs: PDC_1^* and PDC_2^* for CT^*. $\mathcal{A}$ wins the game if the entry (PDC_1^*, PDC_2^*) satisfies the following conditions:

(1) **Verify** $(PK_{GID}, \bigcup AMK_k, CT^*, PDC_1^*) \to 1$;

Table 2 Size and feature comparison

	Public parameters	Ciphertext	Private key	DO	Verifiability	Delegation
[22]	$(\lvert U\rvert+1)\lvert\mathbb{G}\rvert+1\lvert\mathbb{G}_1\rvert$	$(\lvert S_C\rvert+1)\lvert\mathbb{G}\rvert+1\lvert\mathbb{G}_1\rvert$	$2\lvert S_D\rvert\lvert\mathbb{G}\rvert$	No	No	No
[37]	$(\lvert U\rvert+F_U+2)\lvert\mathbb{G}\rvert+F\lvert\mathbb{G}_1\rvert$	$(2\lvert S_C\rvert+1)\lvert\mathbb{G}\rvert+1\lvert\mathbb{G}_1\rvert$	$(\lvert S_D\rvert+F_D)\lvert\mathbb{G}\rvert$	Yes	No	No
SEMAAC	$(\lvert U\rvert+F_U+3)\lvert\mathbb{G}\rvert+F\lvert\mathbb{G}_1\rvert$	$(\lvert S_C\rvert+2)\lvert\mathbb{G}\rvert$	$2\lvert S_D\rvert\lvert\mathbb{G}\rvert$	Yes	Yes	Yes

(2) **Verify** $(PK_{GID}, \bigcup AMK_k, CT^*, PDC_2^*) \rightarrow 1$;

(3) **Decrypt** $(PDC_1^*, DK_{GID}) \neq$ **Decrypt** (PDC_2^*, DK_{GID}).

The condition (1) means that $\prod_{k\in I_C} CV_k = PDC_1^*$ and the condition (2) means that $\prod_{k\in I_C} CV_k = PDC_2^*$. From the condition (1) and (2), we have $\prod_{k\in I_C} CV_k = PDC_1^* = PDC_2^*$. However, the condition (3) implies that $PDC_1^* \neq PDC_2^*$. Thus, $\mathcal{A}$ can win the game with at most a negligible advantage.

We note that even at most $K-1$ AAs are corrupted, the **Verify** algorithm can still guarantee the correctness of the partial decryption ciphertext as long as AA_1 rightly outputs $CV_k = e(PK_{GID}{}^{\sigma_k}, C_1)$.

7 Performance comparison

7.1 Numerical analysis

In Table 2, we compare the parameter size and feature of SEMAAC with two adaptively secure schemes: LOSTW [22] and LMLLXC [37]. We denote $|\mathbb{G}|$ and $|\mathbb{G}_1|$ as an element in $\mathbb{G}$ and $\mathbb{G}_1$, respectively. S_C and S_D denote the attribute sets used in encryption and decryption, respectively. F_U and F_D refer to the number of authorities involved in the system and the private key, respectively. The size of public parameters, ciphertext, and private keys is calculated in terms of the number of group elements.

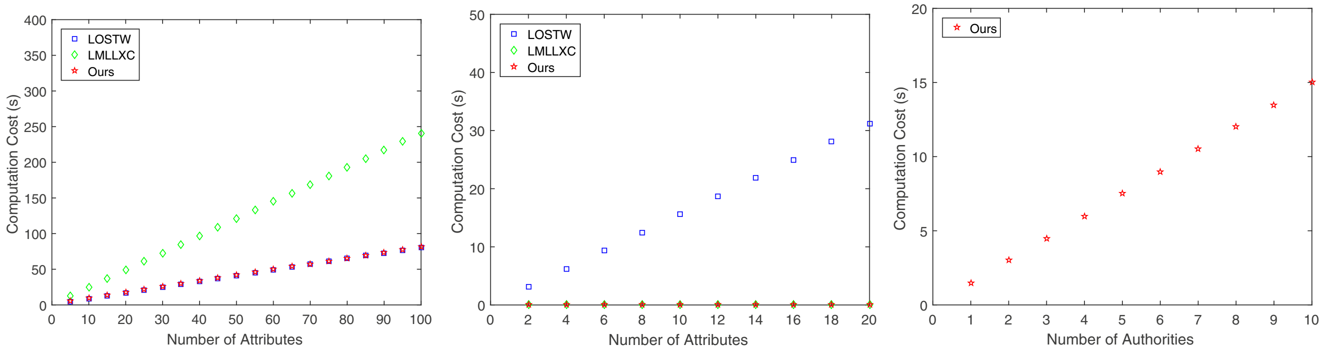
From Table 2, we can see that the size of public parameters, ciphertext, and private keys of SEMAAC is almost the same as that of the underlying KP-ABE scheme [22]. The scheme [37] and ours support the property of outsourced decryption. However, the correctness of partial decryption ciphertext in [37] cannot be guaranteed. Moreover, our work enables the restricted user to delegate someone to interact with the cloud server and check the returned result without exposing the original PHR content. Thus, SEMAAC is more efficient and considerable in mHealth applications.

7.2 Implementation result

We implement SEMAAC, the underline adaptively secure schemes (LOSTW) [22] and decryption outsourced scheme (LMLLXC) [37] by using the JPBC library [45]. All the experiments are conducted over a Type A1 elliptic curve group whose order is a product of 3 517-bit primes. The experiments are run on a laptop with Intel Pentium(R) CPU @ 1.70GHz and 6.00 GB RAM running 32-bit Windows 7.

Figure 2 gives the computation time of the following main algorithms: encryption, decryption, and verify. The experiment result of each algorithm is the average of 20 trials.

Figure 2a presents the encryption time of PO versus the number of attributes. The encryption time of our scheme is as well as the underlying single-authority LOSTW scheme, which requires fewer computation time than the LMLLXC scheme.



(a) Encryption (b) Decryption (c) Verify

Fig. 2 Comparisons of computation cost

Figure 2b shows the user decryption time versus the number of attributes. Since most of the decryption overhead of the LMLLXC scheme and ours is offloaded to the cloud, the user only needs to execute one exponential operation to recover the data encryption key or SEK.

Figure 2c illustrates the verification time of returned PDC versus the number of authorities. The verification time of each PDC increases linearly with the number of related authorities, regardless of how many attributes are involved.

In summary, our scheme is superior to the existing adaptively secure schemes since that it can simultaneously support decentralizing AAs, efficient decryption, delegation, and adaptive security.

8 Conclusion

In this paper, we constructed a secure and efficient multi-authority access control system for IoT-enabled mHealth applications, named SEMAAC, which simultaneously achieved the features of decentralizing authorities, efficient decryption, delegation, and adaptive security. In SEMAAC, there is no requirement to re-initialize the system while a new AA joins in. PO can offload most of the decryption overhead to CSP and check if CSP has honestly computed PDC by interacting with the corresponding AAs. Specially, if a PU cannot directly request the partial decryption ciphertext from CSP, he/she may delegate an agent to interact with the cloud server and verify the result without exposing the original PHR file. We prove the adaptive security of SEMAAC in the standard model. The numerical analysis and implementation results show the efficiency of SEMAAC.

Acknowledgements We thank the reviewers for the helpful comments.

Funding information This research is supported by the National Natural Science Foundation of China under grant no. 61502248, 61872192, 61427801, u1405255, China Postdoctoral Science Foundation (Grant no. 2018M632350), Natural Science Foundation of Jiangsu Province (No. BK20181394) and Qing Lan Project.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

1. Hahn C, Kwon H, Hur J (2016) Efficient attribute-based secure data sharing with hidden policies and traceability in mobile health networks. Mob Inf Syst 2016:13
2. Xu LD, He W, Li S (2014) Internet of things in industries: a survey. IEEE Trans Ind Inf 10(4):2233–2243
3. Wu D, Shi H, Wang H, Wang R, Fang H (2018) A feature-based learning system for internet of things applications. IEEE Internet Things J 1–1. https://doi.org/10.1109/JIOT.2018.2884485
4. Xiong J, Ren J, Chen L et al (2018) Enhancing privacy and availability for data clustering in intelligent electrical service of iot. IEEE Internet Things J 1–10. https://doi.org/10.1109/JIOT.2018.2842773
5. Al-Janabi S, Al-Shourbaji I, Shojafar M, Shamshirband S (2017) Survey of main challenges (security and privacy) in wireless body area networks for healthcare applications. Egyptian Inf J 18(2):113–122
6. Zhang Y, Deng RH, Liu X, Zheng D (2018) Blockchain based efficient and robust fair payment for outsourcing services in cloud computing. Inf Sci 462:262–277
7. Yang YL, Liu R, Chen YL, Li T, Tang Y (2018) Normal cloud model-based algorithm for multi-attribute trusted cloud service selection. IEEE Access 7:37644–37652
8. Sahai A, Waters B (2005) Fuzzy identity-based encryption. In: Cramer R (ed) Advances in cryptology – EUROCRYPT 2005, Lecture Notes in Computer Science, vol 3494. Springer, Berlin, pp 457–473
9. Goyal V, Pandey O, Sahai A, Waters B (2006) Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on computer and communications security, CCS '06. ACM, New York, pp 89–98
10. Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute-based encryption. In: IEEE symposium on security and privacy, 2007. SP '07, pp 321–334
11. Zhang Y, Chen X, Li J, Wong DS, Li H, You I (2017) Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing. Inf Sci 379:42–61
12. Chase M (2007) Multi-authority attribute based encryption. In: Vadhan S (ed) Theory of cryptography. Lecture Notes in Computer Science, vol 4392. Springer, Berlin, pp 515–534
13. Chase M, Chow SS (2009) Improving privacy and security in multi-authority attribute-based encryption. In: Proceedings of the 16th ACM conference on computer and communications security, CCS '09. ACM, New York, pp 121–130
14. Liu Z, Cao Z, Huang Q, Wong D, Yuen T (2011) Fully secure multi-authority ciphertext-policy attribute-based encryption without random oracles. In: Atluri V, Diaz C (eds) Computer security – ESORICS 2011. Lecture Notes in Computer Science, vol 6879. Springer, Berlin, pp 278–297
15. Li Q, Ma J, Li R, Xiong J, Liu X (2015) Large universe decentralized key-policy attribute-based encryption. Secur Commun Netw 8(3):501–509
16. Li Q, Ma J, Li R, Xiong J, Liu X (2015) Provably secure unbounded multi-authority ciphertext-policy attribute-based encryption. Secur Commun Netw 8(18):4098–4109
17. Xue K, Xue Y, Hong J, Li W, Yue H, Wei DSL, Hong P (2017) Raac: robust and auditable access control with multiple attribute authorities for public cloud storage. IEEE Trans Inf Forensics Secur 12(4):953–967
18. Green M, Hohenberger S, Waters B (2011) Outsourcing the decryption of abe ciphertexts. In: Proceedings of the 20th USENIX conference on security, SEC'11. USENIX Association, Berkeley, pp 34–34
19. Lai J, Deng R, Guan C, Weng J (2013) Attribute-based encryption with verifiable outsourced decryption. IEEE Trans Inf Forensics Secur 8(8):1343–1354
20. Li J, Wang Y, Zhang Y, Han J (2017) Full verifiability for outsourced decryption in attribute based encryption. IEEE Trans Serv Comput PP(99):1–1
21. Ning J, Cao Z, Dong X, Liang K, Ma H, Wei L (2018) Auditable σ -time outsourced attribute-based encryption for access control in cloud computing. IEEE Trans Inf Forensics Secur 13(1):94–105

22. Lewko A, Okamoto T, Sahai A, Takashima K, Waters B (2010) Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert H (ed) Advances in cryptology – EUROCRYPT 2010. Lecture Notes in Computer Science, vol 6110. Springer, Berlin, pp 62–91
23. Lewko A, Waters B (2011) Decentralizing attribute-based encryption. Advances in Cryptology – EUROCRYPT 2011. Lecture Notes in Computer Science, vol 6632. In: Paterson K (ed). Springer, Berlin, pp 568–588
24. Zhang Y, Zheng D, Deng RH (2018) Security and privacy in smart health: efficient policy-hiding attribute-based access control. IEEE Internet Things J 5(3):2130–2145
25. Li J, Huang Q, Chen X, Chow SSM, Wong DS, Xie D (2011) Multi-authority ciphertext-policy attribute-based encryption with accountability. In: Proceedings of the 6th ACM symposium on information, computer and communications security, ASIACCS '11. ACM, New York, pp 386–390
26. Qin B, Deng RH, Liu S, Ma S (2015) Attribute-based encryption with efficient verifiable outsourced decryption. IEEE Trans Inf Forensics Secur 10(7):1384–1393
27. Gao C, Lv S, Wei Y, Wang Z, Liu Z, Cheng X (2018) M-sse: an effective searchable symmetric encryption with enhanced security for mobile devices. IEEE Access 1–1
28. Wang X, Zhang Y, Zhu H, Jiang L (2018) An identity-based signcryption on lattice without trapdoor. J Univ Comput Sci 1–1
29. Li J, Li J, Chen X, Jia C, Lou W (2015) Identity-based encryption with outsourced revocation in cloud computing. IEEE Trans Comput 64(2):425–437
30. Gao C, Cheng Q, He P, Susilo W, Li J (2018) Privacy-preserving naive bayes classifiers secure against the substitution-then-comparison attack. Inf Sci 444:72–88
31. Yu Z, Gao CZ, Jing Z, Gupta BB, Cai Q (2018) A practical public key encryption scheme based on learning parity with noise. IEEE Access 6:31918–31923
32. Li J, Li YK, Chen X, Lee PPC, Lou W (2015) A hybrid cloud approach for secure authorized deduplication. IEEE Trans Parallel Distrib Syst 26(5):1206–1216
33. Yang L, Han Z, Huang Z, Ma J (2018) A remotely keyed file encryption scheme under mobile cloud computing. J Netw Comput Appl 106:90–99
34. Wang H, He D, Han J (2017) Vod-adac: anonymous distributed fine-grained access control protocol with verifiable outsourced decryption in public cloud. IEEE Trans Serv Comput PP(99):1–1
35. Wang H, He D, Yu J, Wang Z (2018) Incentive and unconditionally anonymous identity-based public provable data possession. IEEE Trans Serv Comput. https://doi.org/10.1109/TSC.2016.2633260
36. Yu S, Wang C, Ren K, Lou W (2010) Achieving secure, scalable, and fine-grained data access control in cloud computing. In: INFOCOM, 2010 Proceedings IEEE, pp 1–9
37. Li Q, Ma J, Li R, Liu X, Xiong J, Chen D (2016) Secure, efficient and revocable multi-authority access control system in cloud storage. Comput Secur 59:45–59
38. Yang Y, Liu X, Deng RH (2017) Lightweight break-glass access control system for healthcare internet-of-things. IEEE Trans Ind Inf 14(8):3610–3617
39. Li J, Huang X, Li J, Chen X, Xiang Y (2014) Securely outsourcing attribute-based encryption with checkability. IEEE Trans Parallel Distrib Syst 25(8):2201–2210
40. Wang H, Zheng Z, Wu L, Li P (2017) New directly revocable attribute-based encryption scheme and its application in cloud storage environment. Clust Comput 20(3):2385–2392
41. Li J, Chen X, Chow SS, Huang Q, Wong DS, Liu Z (2018) Multi-authority fine-grained access control with accountability and its application in cloud. J Netw Comput Appl 112:89–96
42. Beimel A (1996) Secure schemes for secret sharing and key distribution. DSc dissertation
43. Waters B (2011) Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. Springer, Berlin, pp 53–70
44. Rahulamathavan Y, Veluru S, Han J, Li F, Rajarajan M, Lu R (2016) User collusion avoidance scheme for privacy-preserving decentralized key-policy attribute-based encryption. IEEE Trans Comput 65(9):2939–2946
45. De Caro A, Iovino V (2011) jpbc: Java pairing based cryptography. In: Proceedings of the 16th IEEE symposium on computers and communications, ISCC 2011, Kerkyra, Corfu, Greece, June 28–July 1, pp 850–855

Affiliations

Qi Li[1] · **Hongbo Zhu**[2] · **Jinbo Xiong**[3] · **Ruo Mo**[4] · **Zuobin Ying**[5] · **Huaqun Wang**[1]

Hongbo Zhu
zhb@njupt.edu.cn

Jinbo Xiong
jinbo810@163.com

Ruo Mo
593430655@qq.com

Zuobin Ying
yingzb@ahu.edu.cn

Huaqun Wang
whq@njupt.edu.cn

[1] School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, 210023, China

[2] Jiangsu Innovative Coordination Center of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, 210003, China

[3] College of Mathematics and Informatics, Fujian Normal University, Fuzhou, 350117, China

[4] School of Cyber Engineering, Xidian University, Xi'an, 710071, China

[5] School of Computer Science and Technology, Anhui University, Hefei, 230601, China