Robust, Revocable and Adaptively Secure Attribute-Based Access Control with Outsourced Decryption

Anis Bkakria

IRT SystemX, Paris, France

Abstract. Attribute based encryption (ABE) is a cryptographic technique allowing fine-grained access control by enabling one-to-many encryption. Existing ABE constructions suffer from at least one of the following limitations. First, single point of failure on security meaning that, once an authority is compromised, an adversary can either easily break the confidentiality of the encrypted data or effortlessly prevent legitimate users from accessing data; second, the lack of user and/or attribute revocation mechanism achieving forward secrecy; third, a heavy computation workload is placed on data user; last but not least, the lack of adaptive security in standard models.
In this paper, we propose the first single-point-of-failure free multi-authority ciphertext-policy ABE that simultaneously (1) ensures robustness for both decryption key issuing and access revocation while achieving forward secrecy; (2) enables outsourced decryption to reduce the decryption overhead for data users that have limited computational resources; and (3) achieves adaptive (full) security in standard models. The provided theoretical complexity comparison shows that our construction introduces linear storage and computation overheads that occurs only once during its setup phase, which we believe to be a reasonable price to pay to achieve all previous features.

Keywords: Attribute-Based Encryption, Threshold Cryptography, Adaptive Security

1 Introduction

Cloud computing enables the on-demand provision of various resources, such as computing power and storage over the Internet, freeing companies from maintaining IT infrastructure and managing data centers so they can focus on managing resources to focus on their core business. In addition, cloud computing enables users to take advantage of a variety of powerful resources on a pay-as-you-go basis. Nevertheless, security and privacy issues have become the main obstacle to the wider adoption of cloud computing. According to Techfunnel's top five cloud computing predictions for 2020 [Whi20], security tops the list of the biggest cloud challenges. Hence, users/companies are reluctant to subscribe to a cloud service to outsource their important data.

In a non fully trusted cloud environment, preserving data confidentiality, making appropriate decisions about data access, and enforcing fine-grained access policies are major challenges. Hence, many cryptography-based system models and techniques have been proposed to enable efficient and secure cloud access control. Among the previous, ABE [SW05]. The latter allows to ensure simultaneously confidentiality preservation and fine-grained access control. ABE has succeeded in attracting considerable research efforts [KA+18,ADSLK19,ZDX+20] which allowing to add additional cryptographically functional features, such as access revocation, accountability, and robustness to the basic construction. Unfortunately, all the proposed ABE constructions suffer from at least one of the following limitations. First, the lack of robustness meaning that once the authority responsible of issuing decryption keys to users is compromised, an adversary can either easily break the confidentiality of the encrypted data or effortlessly prevent legitimate users from accessing data. Second, the lack of access revocation making the concerned approaches inflexible. Third, most of the proposed ABE constructions require heavy computation workload to be performed by data user at access time. Last but not least, the lack of security in standard models. We provide a full comparison with related literature in Section 2.

In this paper, we propose a new multi-authority ciphertext-policy ABE (CP-ABE) scheme with some interesting features. First, it ensures robustness for both decryption key issuing and access revocation processes. That is, an adversary needs to compromise several authorities to be able either to break the confidentiality of the outsourced data or to prevent authorized users from accessing outsourced data. Second, our construction enable attribute revocation while achieving forward secrecy. Third, it allows to outsource most part of the decryption process to the cloud server while ensuring that the latter will learn nothing about the partially decrypted data. Fourth, our construction achieves adaptive security in standard models. The construction we propose in this paper is – to our knowledge – the first to provide all previously mentioned features. Finally, we conduct theoretical comparison with similar constructions to show that our construction introduces linear storage and computation overheads that occurs only once during its setup phase.

The paper is organized as follows. Section 2 reviews related work and provides a comprehensive comparison with our construction. Sections 3 and 4 present the assumptions and the adversary model we are considering to achieve provable security. Section 5 formalizes our primitive. Then, in Section 6, we provide the security results. In Sections 7, we discuss the complexity of our construction. Finally, Section 8 concludes.

2 Related Work

Revocable ABE. Several researchers have been devoted to build ABE constructions allowing access revocation. Liang et al. [LCLS09] introduce a provably selectively secure CP-ABE construction that

enables access revocation through proxy re-encryption. Using the latter technique, Luo et al. [LHC10] designed a selectively secure and small attribute universe based CP-ABE supporting policy updating. Always relying on proxy re-encryption, Yu et al. [YWRL10] proposed an AND-gate policy based ABE construction enabling attribute and user revocations. The proposed construction is proved to be selectively secure under the decisional bilinear Diffie-Hellman (DBDH) assumption. To allow the enforcement of non-monotonic access structures, Lewko et al. [LSW10] propose a selectively secure in the standard model ABE construction that enables attribute revocations. Hur and Noh [HN11] relies on a stateless group key distribution method based on binary trees to define a CP-ABE solution enabling efficient attribute revocation. The authors claimed that the proposed scheme achieves backward secrecy and forward secrecy without providing formal security analysis. Yang et al. [YJR13a] propose a CP-ABE construction enabling attribute and user revocation based on a ciphertext re-encryption mechanism performed by a third-party honest-but-curious server. The proposed construction is proved to be selectively secure under the q-type assumption. In [YJR$^+$13b], Yang et al. propose a multi-authority CP-ABE supporting revocation process. The latter is performed mainly by attribute authorities which are responsible of computing an updated decryption key for each non-revoked user. Relying on binary trees, Cui et al. [CDLQ16] proposes a CP-ABE scheme that enables attribute revocation where most of the related computations are delegated to an untrusted server. Similarly to [HN11], the authors of [CDLQ16] claimed that their construction is both backward and forward secure without providing any formal proofs. Liu et al. [LW16] proposed a large universe CP-ABE construction enabling simultaneously user revocation and accountability. The authors shows that the proposed construction is selectively secure in standard models. Li et al. [LML$^+$16] propose a new multiauthority CP-ABE scheme enabling attribute revocation while being adaptively secure in the setting of bilinear groups with composite order. Relying on an untrusted server, Qin et al. [QZZC19] designed an adaptively secure CP-ABE scheme enabling attribute revocation. In the proposed scheme, the untrusted server is used to help non-revoked users to transform ciphertexts. Very recently, Xiong et al. [XHY$^+$21] proposed an adaptively secure CP-ABE scheme allowing attribute revocation. It uses monotonic span program [KW93] as an access structure to reduce the number of pairing and exponentiation operations for encryption and decryption.

Outsourced decryption-based ABE. To mitigate the burden of decryption for data user, Yang et al. [YJR$^+$13b] proposes a construction that outsource most part of the computation to the cloud server. The same idea was later used in [CDLQ16,YCTH18,YCTH18,QZZC19,XHY$^+$21].

A common weakness of the previously mentioned revocable CP-ABE is that they all include a single point of failure of security. That is, as soon as an attribute authority is compromised by an adversary, the latter can easily break the confidentiality of the outsourced data by issuing valid secret decryption keys.

Robust ABE. To mitigate the single-point-of-failure weakness, Li et al. [LXXH15] propose a multi-authority CP-ABE called TMACS. In contrast to previously mentioned approaches, the set of attribute authorities are collaboratively managing the whole set of attributes and no one of them can have full control of any specific attribute. The construction relies on a (t, n) threshold secret sharing protocol (Section 3.2) to require the collaboration of at least t attribute authorities to issue a valid decryption key, which allow to prove that the proposed construction is selectively secure even when $t-1$ authorities are compromised by an adversary. Unfortunately, neither the access revocation, nor the outsourced decryption has been addressed in this work.

Table 1 presents a comprehensive feature comparison of the related CP-ABE schemes. According to it, the construction we propose in this paper is the only one that achieves simultaneously robustness, access revocation, outsourced decryption, and adaptive security.

Approaches	**Robustness**	**Revocation**	**Security Model**	**Outsourced Decryption**
[SW05],[OSW07],[GJPS08], [NYO08],[LCLX09],[EMN$^+$09], [Wat11],[HSMY12],[HSM$^+$14]	✗	✗	Selective	✗
[LCLS09],[LHC10],[YWRL10], [LSW10],[HN11],[YJR13a], [LYH$^+$17]	✗	✓	Selective	✗
[YJR$^+$13b],[CDLQ16],[YCTH18]	✗	✓	Selective	✓
[LXXH15]	✓	✗	Selective	✗
[BSW07],[LW11],[TKN21]	✗	✗	Fully	✗
[LW16],[LML$^+$16]	✗	✓	Fully	✗
[QZZC19],[XHY$^+$21]	✗	✓	Fully	✓
This work	✓	✓	Fully	✓

We used **BS** and **FS** to denote backward and forward secrecy respectively.

Table 1: Feature Comparaison Of CP-ABE Constructions

3 Preliminaries

In this section, we give background information on bilinear maps and the security assumption we are considering. Then, we give a brief description of the trusted third party free secret sharing method proposed by Pedersen in [Ped91].

3.1 Bilinear Maps

Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be two multiplicative cyclic group of prime order p. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map having the following properties:

- Symmetric bilinearity: for all $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, we have $e(g_1^a, g_2^b) = e(g_1^b, g_2^a) = e(g_1, g_2)^{a \cdot b}$.
- Non-degeneracy: $e(g_1, g_2) \neq 1$.
- The group operations in $\mathbb{G}_1$, $\mathbb{G}_2$ and $e(\cdot, \cdot)$ are efficiently computable.

The security of the proposed constructions hold as long as $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently computable homomorphism exists between $\mathbb{G}_1$ and $\mathbb{G}_2$ in either directions. In the sequel, the we refer to the tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e(\cdot, \cdot))$ as a bilinear environment.

Definition 1 (independence [Boy08]). *Let p be some large prime, r, s, t, and c be positive integers. Let $R = \langle r_1, \cdots, r_r \rangle \in \mathbb{F}_p[X_1, \cdots, X_c]^r$, $S = \langle s_1, \cdots, s_s \rangle \in \mathbb{F}_p[X_1, \cdots, X_c]^s$, and $T = \langle t_1, \cdots, t_r \rangle \in \mathbb{F}_p[X_1, \cdots, X_c]^t$ be three tuples of multivariate polynomials over the field $\mathbb{F}_p$. We say that polynomial $f \in \mathbb{F}_p[X_1, \cdots, X_c]$ is dependant on the triple $\langle R, S, T \rangle$ if there exists $r \cdot s + t$ constants $\{\vartheta_{i,j}^{(a)}\}_{i,j=1}^{i=r,j=s}$, $\{\vartheta_k^{(b)}\}_{k=1}^{t}$ such that*

$$f = \sum_{i,j} \vartheta_{i,j}^{(a)} \cdot r_i \cdot s_j + \sum_k \vartheta_k^{(b)} \cdot t_k$$

We say that f is independent of $\langle R, S, T \rangle$ if f is not dependent on $\langle R, S, T \rangle$.

Definition 2 (GDHE assumption [Boy08]). *Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e(\cdot, \cdot))$ be a bilinear environment and r, s, t, and c be positive integers. Let $R = \langle r_1, \cdots, r_r \rangle \in \mathbb{F}_p[X_1, \cdots, X_c]^r$, $S = \langle s_1, \cdots, s_s \rangle \in \mathbb{F}_p[X_1, \cdots, X_c]^s$, and $T = \langle t_1, \cdots, t_r \rangle \in \mathbb{F}_p[X_1, \cdots, X_c]^t$ be three tuples of multivariate polynomials over the field $\mathbb{F}_p$. The GDHE assumption states that, given the vector*

$$H(x_1, \cdots, x_n) = (g_1^{R(x_1, \cdots, x_n)}, g_2^{S(x_1, \cdots, x_n)}, e(g_1, g_2)^{T(x_1, \cdots, x_n)}) \in \mathbb{G}_1^s \times \mathbb{G}_2^s \times \mathbb{G}_T^s$$

it hard to decide whether $U = e(g_1, g_2)_t^{f(x_1, \cdots, x_c)}$ or U is random if f is independent of (R, S, T).

3.2 Trusted Third Party Free Threshold Secret Sharing

In a secret sharing scheme, a secret is distributed among several participants organized in an access structure listing all groups that can access the secret. The objective is to provide information specific to each participant so that only a specific group of participants can reconstruct the secret. Several practical secret sharing schemes have been proposed [BI92,ISN89,Ped91,Sha79]. In this work, we use the trusted third party free threshold secret sharing construction proposed in [Ped91], which we briefly describe as following.

Consider a system involving a set $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$ of n participants and a threshold t ($t \leq n$). Let us suppose that to each participant $P_i \in \mathcal{P}$ is associated a unique scalar $z_i \in \mathbb{Z}$ ($\forall P_i, \forall P_j \in \mathcal{P} : P_i \neq P_j \Leftrightarrow z_i \neq z_j$) representing the public identifier of the participant in the system. First, each participant P_i selects a random scalar $s_i \in \mathbb{Z}_p$ that will represent his/her sub-secret and generates a random polynomial $f_i(x)$ of degree $t-1$ such that $f_i(0) = s_i$. The sum of sub-secrets $S = \sum_{i=1}^{n} s_i$ will represented the master secret that will be shared by the participant. Nevertheless, S is not known to any participant. Second, each participant P_i computes the sub-shares $s_{i,j} = f_i(z_j), 1 \leq j \leq n, j \neq i$ and securely sends $s_{i,j}$ to P_j. Once a participant P_i receives sub-shares from all other $n-1$ participants, he/she/it computes $s_{i,i} = f_i(z_i)$ and computes it own master share as $S_i' = \sum_{j=1}^{n} s_{j,i}$. Once each participant P_i has computed his master share S_i', the master secret key S can be constructed using the Lagrange interpolating formula by any t out of n participants. Let us denote by $S_k', 1 \leq k \leq t$ the set of master shares to be used, the master secret can be constructed as following.

$$\sum_{k=1}^{t} \left(S_k' \cdot \prod_{j=1, j \neq k}^{t} \frac{z_j}{z_j - z_i} \right) = \sum_{i=1}^{n} S_i = S$$

4 System and Security Models

In this section, we introduce the model we are considering. Then we define the scheme we are proposing, the considered threat model, and the security requirements we aim to ensure.

4.1 System Model

The system we consider to build our scheme is composed of five entities: A decentralized certificate authority, multiple attribute authorities, data provides, data users, and a cloud server provider.

- The decentralized certificate authority (DCA) is a consortium blockchain-based PKI management system e.g., [] which is responsible of the setup of the system by choosing it parameters such as, the set of attributes and their respective public key components as well as the bilinear environment to be used. It is also in charge of registering data users and attribute authorities. Finally, DCA is responsible of choosing he robustness level that should be satisfied, i.e., the number of entities that should be compromised to violate the confidentiality of the shared data. We stress that the DCA is not involved in decryption key issuing and access revocation.
- Attribute authorities (AAs) are a set of entities that collaboratively control the access to the shared data by cooperatively issuing decryption keys to data users, and revocation key to the cloud server provider.
- Cloud storage provider (CSP) is responsible of providing data storage and computation capabilities, e.g., outsourced decryption and ciphertext re-encryption.
- The data provider (DP) is the entity that aims to share its data. It encrypts his/her data using a chosen access policy that specifies who can get access to his/her data.
- The data user (DU) represents an entity that will access and use the shared data. A DC is labeled by a set of attributes. It is supposed to be able to download any encrypted (shared) data from the cloud service. However, only DCs who are labeled with proper attributes can successfully decrypt the retrieved encrypted data.

4.2 Definition Of Our Construction

Our construction consists of eight algorithms: `GlobalSetup, CAKeyGen, AAKeyGen, Encrypt, Revoke, DecKeyGen, PartialDecrypt`, and `Decrypt`. The algorithms `GlobalSetup` and `CAKeyGen` are performed by DCA. `AAKeyGen` is performed by AAs. `DecKeyGen` is performed collaboratively between a DU and AAs. `Revoke` is collaboratively performed by AAs and CSP. `Encrypt` is performed by DP. `PartialDecrypt` algorithm involves DU and CSP. Finally, `Decrypt` is performed by DU.

- `GlobalSetup`$(\lambda, t, \Sigma) \rightarrow env$ is a probabilistic algorithm that takes as input the security parameter λ, a robustness level t, and a set of attributes Σ. It outputs the public parameters of the system env. The latter will be implicitly used by all the other algorithms and so will be omitted.
- `AAKeygen`$() \rightarrow (SK, PK)$ is a probabilistic algorithm that returns a secret key share SK and a master public key share PK. ¡
- `CAKeyGen`$(\{PK_i\}) \rightarrow MPK$ is a probabilistic algorithm that takes as input the master public key shares $\{PK_i\}$ of the involved AAs and outputs a master public key MPK.
- `Encrypt`$(M, \mathbb{M}) \rightarrow \chi$ is a probabilistic algorithm that takes a message M and access structure $\mathbb{M}$ and outputs an encrypted data item bundle χ.
- `DecKeyGen`$(MPK,\ AA) \rightarrow \mathcal{K}$ is a probabilistic algorithm that takes as input the master public key MPK and the set of registered attribute authorities AA and output a secret decryption key $\mathcal{K}$.
- `Revoke`$(u,\ \{\sigma_i\}) \rightarrow (MPK, ReK)$ is a probabilistic algorithm that takes as input a data user u and a set of attributes $\{\sigma_i\}$ and returns an updated master public key MPK and an updated re-encryption key ReK.
- `PartialDecrypt`$(\mathcal{SK}_{csp}, \overline{\mathcal{K}},\ ind) \rightarrow ()$ is a deterministic algorithm that takes as input the secret key $\mathcal{SK}_{csp}$ of the CSP, a randomized decryption key $\overline{\mathcal{K}}$, and the index ind of the data item to decrypt χ and returns a partially decrypted data item $\overline{\chi}$.
- `Decrypt`$(\mathcal{K}, \overline{\chi})$ is a deterministic algorithm that takes as input a data used the decryption key $\mathcal{K}$ and a partially decrypted data item $\overline{\chi}$.

4.3 Threat Model

In our scheme, as the DCA capabilities are supposed to be provided by a blockchain-based PKI management system, then we fairly assume that the DCA is an trusted single point of failure-free entity. Hence, DCA is supposed to issue correct signed certificates to the different registered entities involved in the system.

Attribute authorities involved in the system are considered as *honest-but-curious* entities. They are honest in the sense that they are supposed to correctly perform the different operations of our construction, but we suppose that some of them can be corrupted by an adversary who aim to learn as much information as possible about the data. Similarly, we assume that the CSP is also *honest-but-curious* as it will correctly follow the proposed protocol, yet may try to lean information about the shared data.

Data consumers are considered in our construction to be malicious entities that can collude with each other and/or with compromised attribute authorities.

Finally, we suppose that the CSP will not collude with data users to infer more information about the shared data. We believe that this last assumption is fairly reasonable since, in a free market environment, an open dishonest behavior will result in considerable damages for the involved entities.

4.4 Security Requirements

Three security requirements are considered in our construction. Collusion resistance, robustness, as well as forward and backward secrecy.

Collusion Resistance Since we suppose that multiple malicious data users may collude to gain access to an encrypted data that none of them can access alone, we require our construction to be secure against such collusion attack as formalize by the following definition.

Definition 3 (Collusion Resistance). *Let λ be the security parameter, $\mathcal{A}$ be the adversary, $\mathcal{C}$ be the challenger. We consider the following game that we denote $Exp^{C}_{\mathcal{A}}$.*

1. *Setup: $\mathcal{C}$ executes the algorithms `GlobalSetup`, `AAKe- yGen`, and `CAKeyGen`. It then shares the public parameters env and the master public key MPK with $\mathcal{A}$.*
2. *Query – Phase 1: $\mathcal{A}$ can make a set of n adaptive secret decryption key queries by executing `DecKeyGen`. For each query Q_i, $\mathcal{A}$ is allowed to choose the set of attributes Σ_i that should be involved in the decryption key. For each query Q_i, $\mathcal{C}$ executes DecKeyGen to generate a valid secret decryption key $\mathcal{K}_i$ and sends it back to $\mathcal{A}$.*
3. *Challenge: $\mathcal{A}$ chooses two equal-length messages M_0, M_1, and a challenge access structure $\mathbb{M}^*$ such that $\forall i \in [1,n], \Sigma_i$ does not satisfy $\mathbb{M}^*$. Then it sends them to $\mathcal{C}$. The latter chooses a random $\beta \in \{0,1\}$, encrypts M_β under $\mathbb{M}^*$ to get the challenge ciphertext $\mathcal{C}^*$, and sends $\mathcal{C}^*$ to $\mathcal{A}$.*
4. *Query – Phase 2: $\mathcal{A}$ can make adaptive secret decryption key queries as in phase 1. At this level, the only restriction is that the set of attributes Σ_i involved in each query does not satisfies the challenge access structure $\mathbb{M}^*$, otherwise, $\mathcal{A}$ will trivially win the game by running the Decrypt algorithm.*
5. *Guess: $\mathcal{A}$ outputs its guess β' of β.*

We define $\mathcal{A}$'s advantage by $Adv^{Exp^{C}_{\mathbb{A}}}(\lambda) = |Pr[\beta = \beta'] - 1/2|$. Our construction is said to be collusion resistant if $Adv^{Exp^{C}_{\mathbb{A}}}(\lambda)$ is negligible.

Robustness According to the threat model we are considering (Section 4.3), we suppose that a subset of attribute authorities can be compromised by an adversary. Hence, we require our construction to be robust. That is, any encrypted data item remains fully protected against unauthorized entities as far as no more than $t-1$ attribute authorities are compromised. We formalize the robustness requirement using the following definition.

Definition 4 ((t,n)-Robustness). *Let λ be the security parameter, $\mathcal{A}$ be the adversary, and $\mathcal{C}$ be the challenger. We consider the following game that we denote $Exp^{R}_{\mathcal{A}}$. We omit the first four steps of the game since they are the same as defined in $Exp^{C}_{\mathbb{A}}$ (Definition 3).*

5. *Compromise: In this step, $\mathcal{A}$ adaptively chooses $t-1 < n$ attribute authorities and compromises them to get their master secret key shares $sk_i, i \in [1, t-1]$.*
6. *Guess: $\mathcal{A}$ outputs its guess β' of β.*

We define $\mathcal{A}$'s advantage by $Adv^{Exp^{R}_{\mathbb{A}}}(\lambda) = |Pr[\beta = \beta'] - 1/2|$. Our construction is said to be (t,n)-Robust if $Adv^{Exp^{R}_{\mathbb{A}}}(\lambda)$ is negligible.

Forward Secrecy. Forward secrecy is a mandatory property for enabling secure revocation. In the context of attribute based encryption, forward secrecy requires that it should not be feasible for a data user to decrypt the previous and subsequent ciphertexts, if his/her attributes required in decryption are revoked. This requirement is formalized using the following definition.

Definition 5 (Forward Secrecy). *Let λ be the security parameter, $\mathcal{A}$ be the adversary, and $\mathcal{C}$ be the challenger. We consider the following game that we denote $Exp^{B-F}_{\mathcal{A}}$. The first step of this game is the same as defined in $Exp^{C}_{\mathbb{A}}$ (Definition 3) and so will be omitted.*

2. *Query – Pre revocation: this phase is carried out according to the following two steps.*
   (a) *$\mathcal{A}$ performs a secret decryption key queries by executing `DecKeyGen`. For this query, $\mathcal{A}$ is allowed to choose the set of attributes Σ^* that should be involved in the secret decryption key. Once the query is received by $\mathcal{C}$, it executes DecKeyGen to generate a valid secret decryption key $\mathcal{K}$ and sends it back to $\mathcal{A}$.*
   (b) *$\mathcal{A}$ chooses two equal-length messages M_0, M_1, and an access structure $\mathbb{M}_1^*$ such that Σ^* satisfies $\mathbb{M}_1^*$, i.e., $\mathcal{A}$ can use the secret decryption key $\mathcal{K}$ requested in the step (2)(a) to decrypt the two data item M_0, M_1. $\mathcal{A}$ sends M_0, M_1, and $\mathbb{M}_1^*$ to $\mathcal{C}$ who encrypts M_0, M_1 to get the two ciphertexts C_0, C_1.*
3. *Revocation: $\mathcal{C}$ revokes a subset $\Sigma' \subseteq \Sigma^*$ of attributes from $\mathcal{A}$ such that $\Sigma^* \backslash \Sigma'$ no longer satisfies $\mathbb{M}_1^*$.*
4. *Query – Post revocation: $\mathcal{A}$ chooses two equal-length messages M_0', M_1', and an access structure $\mathbb{M}_2^*$ such that $\Sigma^* \backslash \Sigma'$ does not satisfy $\mathbb{M}_2^*$. $\mathcal{A}$ sends M_0', M_1', and $\mathbb{M}_2^*$ to $\mathcal{C}$ who encrypts M_0', M_1' to get the two ciphertexts C_0', C_1'.*
5. *Challenge: $\mathcal{C}$ chooses randomly $\beta_1, \beta_2 \in \{0,1\}$, re-encrypts C_β and C'_β using the re-encryption key ReK and sends them to $\mathcal{A}$.*
6. *Guess: $\mathcal{A}$ outputs its guesses β_1' and β_2' of β_1 and β_2 respectively.*

We define $Adv^{Exp^{B-F}_{\mathcal{A},\beta_1}}(\lambda) = |Pr[\beta_1 = \beta_1'] - 1/2|$ and $Adv^{Exp^{B-F}_{\mathcal{A},\beta_2}}(\lambda) = |Pr[\beta_2 = \beta_2'] - 1/2|$. Our construction is said to be forward secure if $Adv^{Exp^{B-F}_{\mathcal{A},\beta_1}}(\lambda)$ and $Adv^{Exp^{B-F}_{\mathcal{A},\beta_2}}(\lambda)$ are negligible.

5 Our Proposed Scheme

We now present the detailed constructions. We emphasis that, since the DCA capabilities are provided by a consortium blockchain-based PKI management system, we use the same blockchain as a trusted shared storage to store the different public elements exchanged between the different parties that compose our system. Hence, the consortium blockchain is supposed to be accessible to all the entities involved in the system. In the sequel, we refer to the consortium blockchain as $\mathcal{B}$.

5.1 System Initialization

The initialization of the system is performed as described in the following.

GlobalSetup Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e(\cdot,\cdot))$ be a bilinear environment, g_1 and g_2 be a random elements of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. The DCA defines a cryptographic hash functions $H : \mathbb{G}_T \rightarrow 0,1^m$ for some m. Then, it chooses an unforgeable under adaptive chosen message attacks signature system Ξ and generates a signature key CMK and a verification key VMK. This process sends the pubic parameters $env = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e(\cdot,\cdot), g_1, g_2, H, \Xi, CMK, VMK)$ to $\mathcal{B}$ for storage.

System Initialization The system is initialized using the following steps.

- **AA registration**: Each attribute authority AA_i uses Ξ generates a private key $\mathcal{SK}_{AA_i}$ and a public key $\mathcal{PK}_{AA_i}$ and sends a registration request to DCA. If AA_i is a legal authority, the DCA generates a random global identity $aid \in Z_p$, issues a signed certificate $\mathcal{C}ert_{aid}$, and submits the couple $(aid, \mathcal{C}ert_{aid})$ for storage in $\mathcal{B}$.
- **CSP registration:** This step is triggered when CSP sends a registration query to the DCA. Then the DCA assigns a random identifier $cspid \in \mathbb{Z}_p$, issues a certificates $Cert_{cspid}$, and submits the couple $(cspid, Cert_{cspid})$ for storage in $\mathcal{B}$. The secret key of CSP will be initially empty $\mathcal{SK}_{csp} = \emptyset$. The latter will be used to enforce access revocation by updating the encrypted data item.
- **Robustness level selection:** Let us suppose that n attribute authorities $AA = \{A_1, \cdots, A_n\}$ are registered in the system. Using this process, the DCA chooses the robustness level t ($t < n$) that should be satisfied and and sends the master public key $MPK = (env, n, t)$ for storage in $\mathcal{B}$.
- **AA key generation:** This process is performed by each one of the n attribute authorities. The process requires the cooperation of the attribute authorities with each other to call the trusted third party free threshold secret sharing (Section 3.2). Each A_i performs the following steps:
  - Select three random scalars $a_i, \alpha_i, \overline{\alpha}_i \in \mathbb{Z}_p$ as a sub-secrets and generate three random polynomials $f_i(x)$, $h_i(x)$, and $\overline{h}_i(x)$ of degree $t-1$ such that $f_i(0) = a_i$, $h_i(0) = \alpha_i$, and $\overline{h}_i(0) = \overline{\alpha}_i$. Then for all $j \in \{1, 2, \cdots, n\} \backslash \{i\}$, calculate $s_{i,j}^{\{a\}} = f_i(aid_j)$, $s_{i,j}^{\{\alpha\}} = h_i(aid_j)$, and $s_{i,j}^{\{\overline{\alpha}\}} = \overline{h}_i(aid_j)$, and send $s_{i,j}^{\{a\}}$, $s_{i,j}^{\{\alpha\}}$, and $s_{i,j}^{\{\overline{\alpha}\}}$ securely to A_j.
  - After receiving $s_{j,i}^{\{a\}}$, $s_{j,i}^{\{\alpha\}}$ and $s_{j,i}^{\{\overline{\alpha}\}}$ from all other $n-1$ AAs, each A_i calculates

$$sk_i^{\{k\}} = \sum_{j=1}^{n} s_{j,i}^{\{k\}}, \quad k \in \{a, \alpha, \overline{\alpha}\}$$

    $pka_i = g_1^{sk_i^{\{a\}}}, \overline{pka_i} = g_2^{sk_i^{\{a\}}}, pke_i = e(g_1, g_2)^{sk_i^{\{\alpha\}}}$, $pkr_i = g_2^{sk_i^{\{\overline{\alpha}\}}}$. Finally, each A_i sends its master public key share $PK_i = \{pka_i, \overline{pka_i}, pke_i, pkr_i\}$ to $\mathcal{B}$ for storage. The secret key share $SK_i = \{sk_i^{\{k\}}\}_{k \in \{a,\alpha,\overline{\alpha}\}}$.
  - For each attribute $\sigma \in \Sigma$, choose a random scalar $\theta_{\sigma,i} \in \mathbb{Z}_p$, compute $\Theta_{\sigma,i} = g_1^{\theta_{\sigma,i}}$ and $\overline{\Theta}_{\sigma,i} = g_2^{\theta_{\sigma,i}}$ and send it to the $\mathcal{B}$ for storage.
  - For each attribute $\sigma \in \Sigma$, store an initially empty list of users $\mathcal{U}_{r,\sigma}$ denoting the users to whom the attribute σ is revoked.
- **Master public key generation**: This step is performed by the DCA which randomly selects t out of the n AAs master public key shares $\{PK_1, \cdots, PK_n\}$. Let us denote by $\mathcal{I}$ the set of indices of the t chosen master public key shares. The global public key of the system is then computed as follows.

$$pka = \prod_{i=1}^{t} (pka_i)^{\prod_{j \in \mathcal{I}, j \neq i} \frac{aid_i}{aid_j - aid_i}} = g_1^a$$

$$\overline{pka} = \prod_{i=1}^{t} (\overline{pka_i})^{\prod_{j \in \mathcal{I}, j \neq i} \frac{aid_i}{aid_j - aid_i}} = g_2^a$$

$$pke = \prod_{i=1}^{t} (pke_i)^{\prod_{j \in \mathcal{I}, j \neq i} \frac{aid_i}{aid_j - aid_i}} = e(g_1, g_2)^{\alpha}$$

$$pkr = \prod_{i=1}^{t} (pkr_i)^{\prod_{j \in \mathcal{I}, j \neq i} \frac{aid_i}{aid_j - aid_i}} = g_2^{\overline{\alpha}}$$

    with $a = \sum_{i=1}^{n} a_i$, $\alpha = \sum_{i=1}^{n} \alpha_i$, and $\overline{\alpha} = \sum_{i=1}^{n} \overline{\alpha_i}$. Then the CA computes Θ_σ and $\overline{\Theta}_\sigma$, $\forall \sigma \in \Sigma$ as follows:

$$\Theta_\sigma = \prod_{i=1}^{n} \Theta_{\sigma,i}, \quad \overline{\Theta}_\sigma = \prod_{i=1}^{n} \overline{\Theta}_{\sigma,i}$$

Finally, DCA sends the master public key $MPK = \{env, pka, \overline{pka}, pke, pkr, \Theta_\sigma, \overline{\Theta}_\sigma, \pi_p = 0\}_{\sigma \in \Sigma}$ to $\mathcal{B}$ for storage. We note that the scalar π_p represents the timestep on which the public key is created. Meanwhile, the master key $MK = \{a, \alpha, \overline{\alpha}\}$ is shared among the different AAs and does not need to be obtained by any entity.

We note here that the master secret shared key $MSK = \{a, \alpha, \overline{\alpha}\}$ where $a = \sum_{i=1}^{n} a_i$, $\alpha = \sum_{i=1}^{n} \alpha_i$, and $\overline{\alpha} = \sum_{i=1}^{n} \overline{\alpha_i}$ is decided in the system, but it does not need to be obtained by any entity in the system.

5.2 Data Sharing

The data encryption operation Encrypt is performed by the data provider independently. Before outsourcing the data file to the CSP, similarly to most ABE schemes, the data to be shared M is firstly encrypted using a secure symmetric key algorithm (e.g., AES). Then, the used symmetric key will be encrypted as we described in the following steps.

- The data provider starts by defining a monotone boolean formula involving a subset of attributes $\Sigma^* \subseteq \Sigma$ as the access policy that should be enforced as a on the data to be outsourced/shared. Then he/she executes the `Encrypt` algorithm who picks a random scalar $s \in \mathbb{Z}_p$ and uses the component pke of the master public key MPK to generate the symmetric key κ as:

$$\kappa = H(pke^s) = H\left(e(g_1, g_2)^{\alpha \cdot s}\right)$$

  The symmetric key κ is then used to encrypt the data item to be shared M to get a ciphertext that we denote $E_\kappa(M)$.
- The chosen access policy is then transformed into a Linear Secret Sharing Scheme (LSSS) access structure $(\mathbb{M}, \rho)$ as described in [LC10] where $\mathbb{M}$ is an $l \times k$ LSSS matrix and $\rho(x)$ maps each row of $\mathbb{M}$ to an attribute $\sigma \in \Sigma$. Next, in order make sure that only the keys that satisfies $(\mathbb{M}, \rho)$ will be able compute κ, we hide the random element s used to generate κ by choosing a random vector $\vec{v} = \{s, v_2, \cdots, v_k\} \in \mathbb{Z}_p^k$. For each row vector $\mathbb{M}_i$ of $\mathbb{M}$, $\lambda_i = \mathbb{M}_i \cdot \vec{v}^\top$ is calculated and a random scalar $r_i \in \mathbb{Z}_p$ is chosen. The ciphertext encrypting the symmetric encryption key κ is computed as following:

$$\mathcal{C} = \{C' = g_2^s,\ C_i = \overline{pka}^{\lambda_i} \cdot \overline{\Theta}_{\rho(i)}^{-r_i},\ D_i = g_2^{r_i}\}_{i \in [1,l]}$$

  Finally, the data owner sends the encrypted data item bundle $\chi = (E_\kappa(M), \mathcal{C}, \pi_c = \pi_p)$. Similarly, π_c is used to denote the timestep on which the data item has been encrypted. The timestep of encrypted data item is the same as the timestep associated the the master public key MPK.

5.3 User Registration and Key Generation

When a user u_i joins the system, he/she sends a registration query to the DCA to get a unique uid and a signed certificate $Cert_{uid}$. Let us denote by Σ_{u_i} the set of attributes that has to be assigned to u_i according to the role he/she plays in the system. Thus, in order of generate a secret decryption key, u_i has to perform the following two steps.

- First, the data user u_i selects t out of the n registered attribute authorities according to his/her own preferences. Then, u_i separately queries each of the selected t attribute authorities to request a secret decryption key share. We emphasis that a data user will be able to generate a valid secret decryption key if and only if he/she gets t secret decryption key shares from t different attribute authorities. In order to request a secret decryption key share from an attribute authority A_j, u_i sends a secret decryption issuance query containing the identifier uid of u_i and signed using $Cert_{uid}$ to A_j. Once the query is recieved by the A_j, its starts by checking the signature of DCA on $Cert_{uid}$ then authenticates the request content by verifying the signature of u_i on the request. If u_i is authorized to access the shared data, then A_j assigns the set of attributes

$$\Sigma_{u_i}^{(j)} = \Sigma_{u_i} \backslash \{\sigma | u_i \in \mathcal{U}_{r,\sigma}\}$$

  to the u_i where $\{\sigma | u_i \in \mathcal{U}_{r,\sigma}\}$ is the set of attributes that has been revoked from u_i. Then, A_j chooses a random scalar $b_j \in \mathbb{Z}_p$ and then uses the MPK to generate a secret decryption key share for u as following:

$$\mathcal{K}_{u_i,j} = \left\{K_j = g_1^{sk_j} \cdot pka^{b_j},\ L_j = g_1^{b_j},\ K_\sigma = \Theta_\sigma^{b_j}\right\}_{\sigma \in \Sigma_{u_i}^{(j)}}$$

  We emphasis here that the queried attributes authorities may assign different attributes $\Sigma_{u_i}^{(j)}$ to u_i. If that is the case, the computed decryption key will involves only the set of attributes $\Sigma_{u_i} = \cap_{j=0}^{t} \Sigma_{u_i}^{(j)}$ that are assigned by all t attribute authorities.
- Once u_i gains t secret decryption key shares from t different attribute authorities, he/she computes his/her secret decryption key as following.

$$
\begin{aligned}
K &= \prod_{i=1}^{t} K_i^{\prod_{j=1,j\neq i}^{t} \frac{aid_j}{aid_j - aid_i}} \\
&= g_1^{\alpha} \cdot g_1^{a \cdot \sum_{i=1}^{t} (b_i \cdot \prod_{j=1,j\neq i} \frac{aid_j}{aid_j - aid_i})} \\
L &= g_1^{\sum_{i=1}^{t} (b_i \cdot \prod_{j=1,j\neq i} \frac{aid_j}{aid_j - aid_i})} \\
K_\sigma &= \Theta_\sigma^{\sum_{i=1}^{t} (b_i \cdot \prod_{j=1,j\neq i} \frac{aid_j}{aid_j - aid_i})}, \quad \sigma \in \Sigma_{u_i}
\end{aligned}
$$

For sake of simplicity, let us introduce the parameter d:

$$
d = \sum_{i=1}^{t} \left(b_i \cdot \prod_{j=1,j\neq i} \frac{aid_j}{aid_j - aid_i} \right)
$$

Therefore, the user's secret key can be simplified as following:

$$
\mathcal{K}_u = \left\{ K = g_1^{\alpha} \cdot g_1^{a \cdot d},\ L = g_1^{d},\ K_\sigma = \Theta_\sigma^{d},\ \pi_u = \pi_p \right\}_{\sigma \in \Sigma_{u_i}}
$$

5.4 Access Revocation

Our construction aims to achieve robust fine-grained and on-demand access revocation. That is, it requires the collaboration of t out of the n registered attribute authorities to perform access revocation. The revocation process is performed according to the following three steps.

Collaborative Revocation Request The access revocation process is triggered by an attribute authority A_i who wants to revoke the access granted by specific set of attributes $\Sigma_r \in \Sigma$ to a specific set of data users $\mathcal{U}_r$. A_i generates a random scalar $t \in \mathbb{Z}_p$ and computes

$$
\mathcal{R}_i = \{ R_i = g_2^{t \cdot sk_i^{\overline{\alpha}}}, R_v = pkr^t, R_b = g_2^t, \Sigma_r, \mathcal{U}_r \}
$$

Then, A_i signs $\mathcal{R}_i$ using $Cert_{aid}$, sends it to $\mathcal{B}$ for storage, and broadcasts the revocation request to other registered attribute authorities. . Once the revocation request is received by the attribute authorities, each A_j, $j \in [1, n] \backslash \{i\}$ will analyze the access revocation request. If it is legitimate, it computes the revocation request share as following

$$
\mathcal{R}_j = \{ R_j = R_b^{sk_j^{\overline{\alpha}}}, R_v = pkr^t, \Sigma_r, \mathcal{U}_r \}
$$

signs $\mathcal{R}_j$ and sends it to $\mathcal{B}$ for storage.

Each of the attribute authorities that participates to the previous collaborative revocation request will update locally the list of revoked user for each attribute as following:

$$
\forall \sigma \in \Sigma_r : \mathcal{U}_{r,\sigma} = \mathcal{U}_{r,\sigma} \cup \mathcal{U}_r
$$

.

Revocation Enforcement Once more than $t-1$ shares for a revocation request are committed to $\mathcal{B}$, the CSP computes

$$
\prod_{i=1}^{t} R_i^{\prod_{j=1,j\neq i}^{t} \frac{aid_j}{aid_j - aid_i}}
$$

and checks if the latter is equal to R_v. If it is the case, the CSP will be sure that the access revocation is requested by at least t out of the registered n attribute authorities. This is due to the usage of the trusted third party free threshold secret sharing, the collaboration of t attribute authorities are required to compute the shared secret scalar $\overline{\alpha}$.

Then, the CSP generates a random scalar $z \in \mathbb{Z}_p$ and updates the master public key MPK as

$$
\forall \sigma \in \Sigma_r :\ \Theta_\sigma \leftarrow \Theta_\sigma^z, \overline{\Theta}_\sigma \leftarrow \overline{\Theta}_\sigma^z, \text{ and } \pi_p \leftarrow \pi_p + 1
$$

Then, the CSP updates its secret key $\mathcal{SK}_{csp}$ as

$$
\mathcal{SK}_{csp} \leftarrow \mathcal{SK}_{csp} \cup \{ \mu_{\pi_p} = z \}
$$

Finally, the CSP sends the updated MPK for storage in $\mathcal{B}$.

Secret Decryption Key Updating Once an access revocation request is enforced, each data user needs to request new fresh secret decryption as described in Section 5.3.

5.5 Outsourced Pre-decryption and User Decryption

Our construction allows a data user to outsource a part of the decryption computation to the CSP. The main objective of this feature is to allow a data user to shift expensive computations, i.e., pairing operations to CSP without disclosing any information about the shared data to CSP. Outsourced pre-decryption is performed according to the following steps.

Secret Decryption Key Randomization The data user u start by randomizing its secret decryption key as follows. It picks a random scalar $x \in \mathbb{Z}_p$ and computes:

$$\overline{\mathcal{K}}_u = \{K' = K^x, L' = L^x, K'_\sigma = K^x_\sigma, \pi_u\}$$

Then, u sends $\overline{\mathcal{K}}_u$ and the index ind of the data item to be accessed to the CSP.

Outsourced Pre-decryption After receiving ($\overline{\mathcal{K}}_u$, ind), the CSP performs the `PartialDecrypt` algorithm which we detail in Algorithm 1. To be useful for pre-decrypting the data item, the randomized secret decryption key $\overline{\mathcal{K}_u}$ should includes a set of attributes Σ_u that satisfies the access structure $(\mathbb{M}, \rho)$ used to encrypt the requested data item. As detailed in Algorithm 1, the `PartialDecrypt` algorithm takes as input the secret key $\mathcal{SK}_{csp}$ of the CSP, randomized secret decryption key $\overline{\mathcal{K}_u}$, the master public key MPK, and the index ind of the data item to encrypt. It outputs a pre-decrypted symmetric key $\overline{\kappa}$ and the data item ciphertext $E_{kappa}(M)$.

Input: $\mathcal{SK}_{csp}, \overline{\mathcal{K}_u}, MPK, ind$
Output: $\overline{\kappa}, E_\kappa(M)$
$(E_\kappa(M), \mathcal{C}, \pi_c) =$ `get_item(ind)`
$\mathbb{I} = \{i : \rho(i) \in \Sigma_u\}$
if $\pi_u < \pi_p$ **then**
  return `exception("outdated decryption key")`
end
$\{C', C_i, D_i\}_{i\in[1,l]} = \mathcal{C}$
if $\pi_c < \pi_p$ **then**
  z = 1
  for $in \in]\pi_c, \pi_p]$ **do**
    $z = z * \mu_i$
  end
  foreach $i \in [1, l]$ **do**
    $D_i = D_i^z$
  end
  `update_data_item(ind,`$\mathcal{C}$`, `π_p`))`
end
$\overline{\kappa} = \frac{e(K',C')}{\prod_{i\in\mathbb{I}}(e(L',C_i)\cdot e(K'_i,D_i))^{w_i}}$
return $(\overline{\kappa}, E_\kappa(M))$

Algorithm 1: Outsourced Pre-decryption. $\mathbb{M}_u$ is used to denote the sub-matrix of $\mathbb{M}$, where each row of $\mathbb{M}_u$ corresponds to an attribute in Σ_u, $\mathbb{I}$ is a subset of $\{1, 2, \cdots, l\}$, and $\mathbb{M}_i$ is used to denote the ith row of the matrix $\mathbb{M}$. Finally, let $\{w_i\}_{i\in\mathbb{I}}$ be constants such that $\sum_{i\in\mathbb{I}} w_i \cdot \mathbb{M}_i = (1, 0, \cdots, 0)$.

User Decryption Once the user retrieves the pre-decrypted symmetric key $\overline{\kappa}$ and the ciphertext $E_\kappa(M)$, the user can recover the symmetric key κ by performing the following operations:

$$\kappa = H\left(\overline{\kappa}^{1/x}\right)$$

which can be used to decrypt the data item.

6 Security Results

This section presents the security results of our construction. First, in Theorem 1 we show that the our construction provides a correct and fine grained access control capabilities. Then, we prove the collusion resistance, the robustness and the backward and forward secrecy properties in Theorems 2, 3, and 4 respectively.

Theorem 1 (Correctness). *Given a data user u to whom a set of attributes Σ_u is assigned and a ciphertext $\mathcal{C}$ encrypted using an access structure $\mathbb{M}$. Let us denote by $\Sigma_r \subset \Sigma_u$ the revoked attributes for u. As long as $\Sigma_u \backslash \Sigma_r$ satisfies $\mathbb{M}$, then the secret decryption key issued to u allows recovering the plaintext of $\mathcal{C}$.*

Proof. The proof is by contradiction. Suppose, contrary to the statement of the theorem, that there exists a timestep π of the system on which the user u cannot recover the plaintext of $\mathcal{C}$.

As described in Section 5.1, CSP's secret key is initially empty ($SK_{csp} = \emptyset$) and the master public key $MPK = \{env, pka, \overline{pka}, pke, pkr, \Theta_\sigma, \overline{\Theta}_\sigma, \pi_p = 0\}_{\sigma \in \Sigma}$. Let us denote by $\Theta_{\sigma,\pi}$ (resp. $\overline{\Theta}_{\sigma,\pi}$) the value of Θ_σ (resp. $\overline{\Theta}_\sigma$) at timestep π of the system.

Let us now suppose that m access revocations are performed in the system, each revokes the set of attributes $\Sigma_{r,i} \in \Sigma_u$ from u, $i \in [1, m]$. According to the access revocation process of our construction (Section 5.4), each access revocation enforced at the timestep π of the system adds a random scalar $\mu_{\pi+1} \in \mathbb{Z}p$ to $\mathcal{SK}_{csp}$ and updates the MPK as $\Theta_\sigma \leftarrow \Theta_\sigma^{\mu_{\pi+1}}$ and $\overline{\Theta}_\sigma \leftarrow \overline{\Theta}_\sigma^{\mu_{\pi+1}}$.

Let us denote by $\mathcal{I}_\sigma = \{i | \sigma \in \Sigma_{r,i}\}$ the set indices of access revocations in which the attribute σ is involved. At the timestep π,i.e., after performing π access revocations, we have $\mathcal{SK}_{csp} = \{\mu_i\}_{i=1}^{\pi}$, and $\forall \sigma \in \Sigma : \Theta_\sigma = \Theta_{\sigma,0}^{\prod_{i \in \mathcal{I}_\sigma} \mu_i}$ and $\overline{\Theta}_\sigma = \overline{\Theta}_{\sigma,0}^{\prod_{i \in \mathcal{I}_\sigma} \mu_i}$.

According to Section 5.3, at any timestep $\pi^* \in [0, m]$, an updated secret decryption key, i.e., a secret decryption key issued at timestep p_i will have the form

$$\mathcal{K}_u = \left\{ K = g_1^\alpha \cdot g_1^{a \cdot d},\ L = g_1^d,\ K_\sigma = \Theta_{\sigma,0}^{d \cdot \prod_{i \in \mathcal{I}_\sigma} \mu_i},\ \pi_u = \pi^* \right\}_{\sigma \in \Sigma_{u_i}}$$

Now, according to Algorithm 1 (lines 7 to 16), an updated encrypted data item is a data item where $\pi_c = \pi_p$ and

$$\mathcal{C} = \{ C' = g_2^s,\ C_i = \overline{pka}^{\lambda_i} \cdot \overline{\Theta}_{\rho(i),0}^{-r_i \cdot \prod_{i \in \mathcal{I}_{\rho(i)}} \mu_i},\ D_i = g_2^{r_i} \}_{i \in [1,l]}$$

Now, according to the outsourced pre-decryption process of our construction, the randomized secret decryption key is

$$\overline{\mathcal{K}}_u = \{ K' = K^x, L' = L^x, K'_\sigma = K_\sigma^x, \pi_u \}$$

As we suppose that $\Sigma_u \backslash \Sigma_r$ satisfies the access structure $\mathbb{M}$ used to encrypt $\mathcal{C}$. Then, we have

$$\exists \mathbb{I} \subseteq \{i | \rho(i) \in \Sigma_u \backslash \Sigma_r\}, \exists\ w_i : \sum_{i \in \mathbb{I}} w_i \cdot \mathbb{M}_i = (1, 0, \cdots, 0)$$

where $\mathbb{M}_i$ denotes the ith row of the matrix $\mathbb{M}$. Then, according to Algorithm 1 (line 17), we have

$$\begin{aligned}
\overline{\kappa} &= \frac{e(K', C')}{\prod_{i \in \mathbb{I}} \left(e(L', C_i) \cdot e(K'_i, D_i) \right)^{w_i})} \\
&= \frac{e(g_1^{\alpha \cdot x} \cdot g_1^{a \cdot d \cdot x}, g_2^s)}{\prod_{i \in \mathbb{I}} \left(e(g_1^{d \cdot x}, g_2^{a \cdot \lambda_i} \cdot \overline{\Theta}_{\rho(i),0}^{-r_i \cdot \prod_{j \in \mathcal{I}_{\rho(i)}} \mu_i}) \cdot e(\Theta_{\rho(i),0}^{d \cdot x \cdot \prod_{j \in \mathcal{I}_\sigma} \mu_i}, g_2^{r_i}) \right)^{w_i}} \\
&= e(g_1, g_2)^{\alpha \cdot s \cdot x}.
\end{aligned}$$

Then by applying the user decryption operation we get

$$H\left(\overline{\kappa}^{1/x} \right) = H\left(e(g_1, g_2)^{\alpha \cdot s} \right) = \kappa$$

Finally, κ can be used to recover the plaintext of $\mathcal{C}$ which contradicts our assumption.

Theorem 2 (Collusion Resistance). *Our construction is collusion resistant under the GDHE assumption.*

Proof. We prove the collusion resistance property using the security game $Exp_{\mathcal{A}}^{C}$ (Definition 3). In the following we show that the advantage of the adversary $\mathcal{A}$ to win $Exp_{\mathcal{A}}^{C}$ is negligible under the GDHE assumption (Definition 2).

Let us denote by $\Omega_{\mathbb{G}_1}$, $\Omega_{\mathbb{G}_2}$, and $\Omega_{\mathbb{G}_t}$ the set of elements of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_t$ respectively that are to be known by $\mathcal{A}$ during $Exp_{\mathcal{A}}^{C}$. According to the latter, first the challenger performs `GlobalSetup`, `AAKeyGen`, and `CAKeyGen` and sends $env = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e(\cdot,\cdot), g_1, g_2, H, \Xi, CMK, VMK)$ and $MPK = \{env, pka, \overline{pka}, pke, pkr, \Theta_\sigma, \overline{\Theta}_\sigma, \pi_p = 0\}_{\sigma \in \Sigma}$ to $\mathcal{A}$. Hence, at the end of the setup phase, we have:

$$\begin{aligned}
\Omega_{\mathbb{G}_1} &= \{pka, \Theta_\sigma\}_{\sigma \in \Sigma} \\
\Omega_{\mathbb{G}_2} &= \{\overline{pka}, pkr, \overline{\Theta}_\sigma\}_{\sigma \in \Sigma} \\
\Omega_{\mathbb{G}_t} &= \{pke\}
\end{aligned}$$

Afterwards, during the first phase of the query step, each secret decryption request query Q_j issued by $\mathcal{A}$ and answered by $\mathcal{C}$ adds $\{g_1^{sk_{i,j}} \cdot pka^{b_{i,j}}, g_1^{b_{i,j}}, \Theta_\sigma^{b_{i,j}}\}_{\sigma \in \Sigma_j, i \in \mathbb{A}_j}$ to $\Omega_{\mathbb{G}_1}$ and nothing to $\Omega_{\mathbb{G}_2}$ and $\Omega_{\mathbb{G}_t}$, where $\mathbb{A}_j$ is used to denote the indices (in the set of all registered AA) of the attribute authorities used by $\mathcal{C}$ to answer the query Q_j. Then, in the challenge phase, the ciphertext $\mathcal{C}^*$ returned to $\mathcal{A}$ adds $\{g_2^s, \overline{pka}^{\lambda_i} \cdot \overline{\Theta}_{\rho(i)}^{r_i}, g_2^{r_i}\}_{i \in [1,l]}$ to $\Omega_{\mathbb{G}_2}$ and nothing to $\Omega_{\mathbb{G}_1}$ and $\Omega_{\mathbb{G}_t}$. Then, similarly to the first phase, the second phase of the query step adds $\{g_1^{sk_{i,j}} \cdot pka^{b_{i,j}}, g_1^{b_{i,j}}, \Theta_\sigma^{b_{i,j}}\}_{\sigma \in \Sigma_j}$ to $\Omega_{\mathbb{G}_1}$ and nothing to $\Omega_{\mathbb{G}_2}$ and $\Omega_{\mathbb{G}_t}$. Hence, at the end of the second phase of the query step, supposing that $\mathcal{A}$ performs m secret decryption queries during the query phase, he/she disposes of the following elements

$$\begin{aligned}
\Omega_{\mathbb{G}_1} =& \{pka, \Theta_\sigma, g_1^{sk_{i,j}} \cdot pka^{b_{i,j}}, g_1^{b_{i,j}}, \Theta_\sigma^{b_{i,j}}\}_{\sigma \in \Sigma_j, j \in [1,m], i \in \mathbb{A}_i} \\
\Omega_{\mathbb{G}_2} =& \{\overline{pka}, pkr, \overline{\Theta}_\sigma, g_2^s, \overline{pka}^{\lambda_i} \cdot \overline{\Theta}_{\rho(i)}^{r_i}, g_2^{r_i}\}_{i \in [1,l]} \\
\Omega_{\mathbb{G}_t} =& \{pke\}
\end{aligned}$$

Hence, to prove the theorem we need to show that the advantage of $\mathcal{A}$ to decide whether the element $U \in \mathbb{G}_t$ used to generate the symmetric key κ is equal to pke^s or a random element of $\mathbb{G}_t$ is negligible. According to the GHDE assumption, the previous statement can be proved by showing that $f = \alpha \cdot s$ is independent of (R, S, T) where

$$R = \{a, \sum_{i=1}^{n} \theta_{\sigma,i}, sk_{i,j} + b_{i,j}, b_{i,j} \cdot \sum_{i=1}^{n} \theta_{\sigma,i}\}_{\sigma \in \Sigma, j \in [1,m], i \in \mathbb{A}_i}$$
$$S = \{a, \overline{\alpha}, \sum_{i=1}^{n} \theta_{\sigma,i}, s, a \cdot \lambda_j + r_j \cdot \sum_{i=1}^{n} \theta_{\sigma,i}, r_j\}_{i \in [1,n], j \in [1,l]}$$
$$T = \{\alpha\}$$

According to Definition 1, to prove that f is independent of (R, S, T), we need to prove that no combination of elements from (R, S, T) could exists with non-negligible probability such that there exists constants $\vartheta_{i,j}^{(a)}$, $\vartheta_k^{(b)}$ so that the following equation holds:

$$\alpha \cdot s = \sum_{i,j} \vartheta_{i,j}^{(a)} \cdot r_i \cdot s_j + \sum_k \vartheta_k^{(b)} \cdot t_k \tag{1}$$

At this level, we emphasis that since a, $\theta_{\sigma,i}$, and $b_{i,j}$ are random variables in $\mathbb{Z}_p$, then the probability that a combination of these elements will give f is negligible. Then all elements of R involving a, $\theta_{\sigma,i}$, and $b_{i,j}$ cannot be part of Equation 3, which means that $R = \emptyset$. Similarly we can exclude the elements a, $\overline{\alpha}$, $\sum_{i=1}^{n} \theta_{\sigma,i}$, and r_j from S to get $S = \{s\}$. So, let $\vartheta^{(1)}$, $\vartheta^{(2)}$ be constant such that:

$$\alpha \cdot s = \vartheta^{(1)} \cdot s + \vartheta^{(2)} \cdot \alpha \tag{2}$$

To conclude the proof, we need to show that no $\vartheta^{(1)}$ and $\vartheta^{(2)}$ can exist such that Equation 2 holds. So let us consider Equation 2 as a polynomial in α and we regroup the monomials according to their degree in α to get $\vartheta^{(1)} \cdot s = 0$. Since s is a random variable in $\mathbb{Z}_p$, then $\vartheta^{(1)} = 0$. Finally, we do the same by considering Equation 2 as a polynomial in s we get $\alpha = \vartheta^{(1)}$ **(I)**. However, since α is generated randomly from $\mathbb{Z}_p$, then $\vartheta^{(1)} \neq 0$ with overwhelming probability, which contradicts (I) and concludes the proof.

Theorem 3 (Robustness). *Our construction is (t, n)-robust under the GDHE assumption.*

Proof. According to Definition 4, to prove the robustness of our construction, we need to show that the advantage of the adversary $\mathcal{A}$ for winning the game $Exp_{\mathcal{A}}^{R}$ is negligible under the GDHE assumption.

Similarly, we use $\Omega_{\mathbb{G}_1}$, $\Omega_{\mathbb{G}_2}$, and $\Omega_{\mathbb{G}_t}$ to denote the set of elements of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_t$ respectively that are to be known by $\mathcal{A}$ during $Exp_{\mathbb{A}}^{C}$. As the first four steps of the $Exp_{\mathcal{A}}^{R}$ security game (Definition 4) are the same in the security game $Exp_{\mathcal{A}}^{C}$ (Definition 3), then we follow the same strategy as in the proof of Theorem 2 to show that at the end of the four step of the security game $Exp_{\mathcal{A}}^{R}$, we have

$$\Omega_{\mathbb{G}_1} = \{pka, \Theta_\sigma, g_1^{sk_{i,j}} \cdot pka^{b_{i,j}}, g_1^{b_{i,j}}, \Theta_\sigma^{b_{i,j}}\}_{\sigma \in \Sigma_j, j \in [1,m], i \in \mathcal{A}_i}$$
$$\Omega_{\mathbb{G}_2} = \{\overline{pka}, pkr, \overline{\Theta}_\sigma, g_2^s, \overline{pka}^{\lambda_i} \cdot \overline{\Theta}_{\rho(i)}^{r_i}, g_2^{r_i}\}_{i \in [1,l]}$$
$$\Omega_{\mathbb{G}_t} = \{pke\}$$

In the step 5 of $Exp_{\mathcal{A}}^{R}$, $\mathcal{A}$ can adaptively choose $t-1$ out the n registered attribute authorities and compromise them to get their secret key shares. Let us denote by AA_c the set of compromised attribute authorities. Knowing the secret key shares SK_A, $A \in AA_c$, the adversary $\mathcal{A}$ can compute $G_i^{SK_A^z}$ for all $z \in \mathbb{Z}$, all $G_i \in \Omega_{\mathbb{G}_i}$, and all $A \in AA_c$. Therefore, to prove the theorem, we need to show that the advantage of $\mathcal{A}$ to decide whether the element $U \in \mathbb{G}_t$ used to generate the symmetric key κ is equal to pke^s or a random element of $\mathbb{G}_t$ is negligible. Relying on the GDHE assumption, the previous statement can be proved if we can show that $f = \alpha \cdot s$ is independent of (R, S, T) where

$$R = \{a \cdot SK_A^z, SK_A^z \cdot \sum_{i=1}^{n} \theta_{\sigma,i}, (sk_{i,j} + b_{i,j}) \cdot SK_A^z, SK_A^z \cdot b_{i,j} \cdot \sum_{i=1}^{n} \theta_{\sigma,i}\}_{\sigma \in \Sigma, j \in [1,m], i \in \mathcal{A}_i, z \in \mathbb{Z}, A \in AA_c}$$
$$S = \{a \cdot SK_A^z, \overline{\alpha} \cdot SK_A^z, SK_A^z \cdot \sum_{i=1}^{n} \theta_{\sigma,i}, s \cdot, SK_A^z \cdot (a \cdot \lambda_j + r_j \cdot \sum_{i=1}^{n} \theta_{\sigma,i}), SK_A^z \cdot r_j\}_{i \in [1,n], j \in [1,l], z \in \mathbb{Z}, A \in AA_c}$$
$$T = \{\alpha \cdot SK_A^z\}_{z \in \mathbb{Z}, A \in AA_c}$$

Now based on the dependence definition (Definition 1), to prove the theorem we need to show that no combination of elements of (R, S, T) could exists with non-negligible probability such that there exists constants $\vartheta_{i,j}^{(a)}$, $\vartheta_k^{(b)}$ so that the following equation holds:

$$\alpha \cdot s = \sum_{i,j} \vartheta_{i,j}^{(a)} \cdot r_i \cdot s_j + \sum_k \vartheta_k^{(b)} \cdot t_k \tag{3}$$

By, following the same reasoning as in the proof of Theorem 3, we can exclude all elements containing $\overline{\alpha}$, a, $\theta_{\sigma,i}$, and $b_{i,j}$ from the sets R, S, and T. Then we get

$$R = \emptyset, \quad S = \{s \cdot SK_A^z\}_{z \in \mathbb{Z}, A \in AA_c} \text{ and } \quad T = \{\alpha \cdot SK_A^z\}_{z \in \mathbb{Z}, A \in AA_c}$$

So, let $\vartheta_{z,A}^{(1)}$, $\vartheta_{z,A}^{(2)}$ be constant such that:

$$\alpha \cdot s = \sum_{z \in \mathbb{Z}, A \in AA_c} \vartheta_{z,A}^{(1)} \cdot s \cdot SK_A^z + \sum_{z \in \mathbb{Z}, A \in AA_c} \vartheta_{z,A}^{(2)} \cdot \alpha \cdot SK_A^z \tag{4}$$

To prove that f is independent of (R, S, T), we need to show that no $\vartheta_{z,A}^{(1)}$ and $\vartheta_{z,A}^{(2)}$ for all $z \in \mathbb{Z}$ and all $A \in AA_c$ can exist such that Equation 4 holds. Now let us consider the previous equation as a polynomial in s. We regroup the monomials according to their degree in s to get

(i) $\sum_{z \in \mathbb{Z}, A \in AA_c} \vartheta_{z,A}^{(2)} \cdot \alpha \cdot SK_A^z = 0$
(ii) $\alpha \cdot s = \sum_{z \in \mathbb{Z}, A \in AA_c} \vartheta_{z,A}^{(1)} \cdot s \cdot SK_A^z$

First, let us focus on Equation (i). If the latter holds, then this means that one of the following condition holds:

(i.1) α is a root of non-zero polynomial of degree one
(i.2) SK_A is a root of non-zero polynomial of degree z
(i.3) $\forall z \in \mathbb{Z}, A \in AA_c \vartheta_{z,A}^{(2)} = 0$

Since α and SK_A are chosen randomly from $\mathbb{Z}_p$, with p is large prime, then the first two conditions hold with negligible probability of $1/p$ and z/p respectively. Therefore, we deduce **(I)** that condition (i.3) holds with overwhelming probability.

Now let us focus on Equation (ii) and consider it as a polynomial in s, then we get

$$\begin{aligned} \alpha &= \sum_{z \in \mathbb{Z}, A \in AA_c} \vartheta_{z,A}^{(1)} \cdot SK_A^z \\ \sum_{i=1}^{n} f_i(0) &= \sum_{z \in \mathbb{Z}, A \in AA_c} \vartheta_{z,A}^{(1)} \cdot \left(\sum_{i=1}^{n} f_i(id_A) \right)^z \end{aligned} \tag{5}$$

At this level, let us consider the previous equation (Equation 5) as a polynomial in f_i. We regroup the monomials according to their degree in f_i to get the following:

(ii.1) $\forall z > 1, \sum_{z \in \mathbb{Z}, A \in AA_c} \vartheta_{z,A}^{(1)} \cdot \left(\sum_{i=1}^{n} f_i(id_A)\right)^z = 0$
(ii.2) $\sum_{i=1}^{n} f_i(0) = \sum_{A \in AA_c} \vartheta_{z,A}^{(1)} \cdot \sum_{i=1}^{n} f_i(id_A)$

From equation (ii.1), since f_i is randomly chosen from $\mathbb{Z}_p$, then we can deduce **(II)** that, with overwhelming probability ($\simeq z/p$), for all $z > 1$, all $A \in AA_c$, $\vartheta_{z,A}^{(1)} = 0$.

Now let us focus on Equation (ii.2). Since AA_c is composed of $t-1$ attribute authorities, let us denote by $\{A_1^{(c)}, \cdots, A_t^{(c)}\}$ the set of attribute authorities in AA_c. Then using the previous notation, we can rewrite Equation (ii.2) as

$$\sum_{i=1}^{n} f_i(0) = \sum_{i=1}^{n} \sum_{j=1}^{t-1} \vartheta_{1,t}^{(1)} \cdot f_i(id_A) \tag{6}$$

As $\forall i \in [1, n]$, f_i is randomly chosen from $\mathbb{Z}_p^{t-1}$, then we have

$$\forall i \in [1, n], f_i(0) = \sum_{j=1}^{t-1} \vartheta_{1,t}^{(1)} \cdot f_i(id_A) \tag{7}$$

Now, by considering the fact that for all $i \in [1, n]$, f_i is a polynomial of degree $t-1$, then if Equation 7 holds, then one can use it to break the security of Shamir's secret sharing scheme [Sha79] which has been proven to be unconditionally secure. Then no $\vartheta_{1,t}^{(1)}$ can be found so that Equation 7 holds **(III)**. Then, (I), (II), and (III) conclude the proof.

Theorem 4. *Our construction is backward and forward secure under the GDHE assumption.*

Proof. According to Definition 5, to prove that our scheme ensures forward (resp. backward) secrecy, we need to show that $Adv^{Exp_{\mathcal{A},\beta_1}^{B-F}}(\lambda)$ (resp. $Adv^{Exp_{\mathcal{A},\beta_2}^{B-F}}(\lambda)$) is negligible under the GDHE assumption.

Similarly to previous proofs, let us denote by $\Omega_{\mathbb{G}_1}$, $\Omega_{\mathbb{G}_2}$, and $\Omega_{\mathbb{G}_t}$ the set of elements of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_t$ respectively that are to be known by $\mathcal{A}$ during $Exp_{\mathbb{A}}^{B-F}$. At the end of the step (2)(a), the secret decryption key issued by $\mathcal{C}$ and sent to $\mathcal{A}$ adds

$$\{g_1^{\alpha} \cdot g_1^{a \cdot d}, g_1^d, \Theta_\sigma^d\}_{\sigma \in \Sigma^*}$$

to $\Omega_{\mathbb{G}_1}$ and nothing to $\Omega_{\mathbb{G}_2}$ and $\Omega_{\mathbb{G}_t}$. The step (2)(b) of $Exp_{\mathbb{A}}^{B-F}$ adds nothing to $\Omega_{\mathbb{G}_1}$, $\Omega_{\mathbb{G}_2}$ and $\Omega_{\mathbb{G}_t}$. Furthermore, during the step (3) of $Exp_{\mathbb{A}}^{B-F}$, the access revocation operation adds the random scalar $\mu \in \mathbb{Z}_p$ to the secret key SK_{CSP} of the CSP and update the master public key MPK by setting $\Theta \leftarrow \Theta_\sigma^\mu$ and $\overline{\Theta}_\sigma \leftarrow \overline{\Theta}_\sigma^\mu$ for all $\sigma \in \Sigma'$. Hence, the revocation step adds Θ_σ^μ to $\mathbb{G}_1$, $\overline{\Theta}_\sigma^\mu$ to $\mathbb{G}_2$ and nothing to $\mathbb{G}_t$. Similarly to the step (2)(b), the step (4) adds nothing to $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$. Finally, the challenge step (step (5)) adds

$$\{g_2^s, \overline{pka}^{\lambda_i} \cdot \overline{\Theta}_{\rho(i)}^{\mu \cdot r_i}, g_2^{r_i}, g_2^{s'}, \overline{pka}^{\lambda_i'} \cdot \overline{\Theta}_{\rho'(i)}^{\mu \cdot r_i'}, g_2^{r_i'}, \overline{\Theta}_\sigma^\mu, pkr\}_{\rho(i) \in \Sigma^*, \rho'(i) \in \Sigma^* \setminus \Sigma', \sigma \in \Sigma'}$$

to $\Omega_{\mathbb{G}_2}$ and nothing to $\Omega_{\mathbb{G}_1}$ and $\Omega_{\mathbb{G}_t}$. Hence, at the end of step (5), we have

$$\Omega_{\mathbb{G}_1} = \{g_1^{\alpha} \cdot g_1^{a \cdot d}, g_1^{d}, \Theta_{\sigma}^{d}, \Theta_{\sigma}^{\mu}\}_{\sigma^* \in \Sigma^*, \sigma \in \Sigma'}$$
$$\Omega_{\mathbb{G}_2} = \{g_2^{s}, \overline{pka}^{\lambda_i} \cdot \overline{\Theta}_{\rho(i)}^{\mu \cdot r_i}, g_2^{r_i}, g_2^{s'}, \overline{pka}^{\lambda'_i} \cdot \overline{\Theta}_{\rho'(i)}^{\mu \cdot r'_i}, g_2^{r'_i}, \overline{\Theta}_{\sigma}^{\mu}, pkr\}_{\rho(i) \in \Sigma^*, \rho'(i) \in \Sigma^* \setminus \Sigma', \sigma \in \Sigma'}$$
$$\Omega_{\mathbb{G}_t} = \{pke\}$$

Therefore, to prove backward (resp. forward) secrecy, we need to show that the advantage of $\mathcal{A}$ to decide whether $U = e(g_1, g_2)^{\alpha \cdot s}$ (resp. $U = e(g_1, g_2)^{\alpha \cdot s'}$) or a random element of $\mathbb{G}_t$ is negligible. Based on the GDHE assumption, the previous statement can be proved if we can show that $f = \alpha \cdot s$ (resp. $f' = \alpha \cdot s'$) is independent of (R,S,T) where

$$R = \left\{ \alpha + a \cdot d, d, d \cdot \sum_{i=1}^{n} \theta_{\sigma,i}, \mu \cdot \sum_{i=1}^{n} \theta_{\sigma,i} \right\}_{\sigma \in \Sigma^*, j \in [1,m], i \in \mathbb{A}_i}$$
$$S = \left\{ s, a \cdot \lambda_i, \mu \cdot r_i \cdot \sum_{i=1}^{n} \theta_{\sigma,i}, r_i, s', a \cdot \lambda'_i, \mu \cdot r'_i \cdot \sum_{i=1}^{n} \theta_{\sigma,i}, r'_i \right\}_{\rho(i) \in \Sigma^*, \rho'(i) \in \Sigma^* \setminus \Sigma', \sigma \in \Sigma'}$$
$$T = \{\alpha\}$$

Then to show that f (resp. f') is independent of (R,S,T), we must show that there exists no constants $\vartheta_{i,j}^{(a)}$, $\vartheta_k^{(b)}$, $\vartheta_{i,j}^{\prime(a)}$, $\vartheta_k^{\prime(b)}$ such that

$$\alpha \cdot s = \sum_{i,j} \vartheta_{i,j}^{(a)} \cdot r_i \cdot s_j + \sum_{k} \vartheta_k^{(b)} \cdot t_k \qquad \text{and} \tag{8}$$

$$\alpha \cdot s' = \sum_{i,j} \vartheta_{i,j}^{\prime(a)} \cdot r_i \cdot s_j + \sum_{k} \vartheta_k^{\prime(b)} \cdot t_k \tag{9}$$

Let us start by Equation 8. Since a, d , $\theta_{\sigma,i}$, α, s', r_i, r'_i, μ and s are random variables in $\mathbb{Z}_p$ and that a, d, $\theta_{\sigma,i}$, s', r_i, r'_i, and μ, are independent of α and s, then the probability that a combination of elements involving a, d, $\theta_{\sigma,i}$, s', r_i, r'_i, and μ but not α, s, and λ_i will give f is negligible. So we can exclude all elements of R,S, and T involving a, d, $\theta_{\sigma,i}$, s', r_i, r'_i, and μ but not α, s, and λ_i from Equation 8. Therefore we have

$$\alpha \cdot s = \vartheta^{(1)}(\alpha + a \cdot d) \cdot s + \sum_{\rho(i) \in \Sigma^*} \vartheta_i^{(2)}(\alpha + a \cdot d) \cdot (a \cdot \lambda_i) + \vartheta^{(3)} \cdot \alpha \tag{10}$$

Let us consider Equation 10 as a polynomial in α and regroup the the monomials according to their degree in α to get

(i) $\alpha \cdot s = \vartheta^{(1)} \cdot \alpha \cdot s + a \cdot \alpha \sum_{\rho(i) \in \Sigma^*} \vartheta_i^{(2)} \cdot \lambda_i + \vartheta^{(3)} \cdot \alpha$
(ii) $\vartheta^{(1)} \cdot a \cdot d \cdot s + a^2 \cdot d \sum_{\rho(i) \in \Sigma^*} \vartheta_i^{(2)} \cdot \lambda_i = 0$

Let us now focus on Equation (i) by considering it as polynomial on a. By regrouping the monomials according to their degree, we have:

(i.1) $s = \vartheta^{(1)} \cdot s + \vartheta^{(3)}$
(i.2) $\sum_{\rho(i) \in \Sigma^*} \vartheta_i^{(2)} \cdot \lambda_i = 0$

At this level, by considering Equation (i.1) as polynomial is s, we get $\vartheta^{(1)} = 1$. Let us now focus on Equation (ii) by considering it as polynomial in a. Then we have $\vartheta^{(1)} \cdot d \cdot s = 0$. As d and s are randomly chosen from $\mathbb{Z}_p$, then $\vartheta^{(1)} = 0$ witch overwhelming probability, which contradicts $\vartheta^{(1)} = 1$ and prove that Equation 8 does not hold. We can follow the same previous steps to prove that Equation 9 does not hold.

7 The complexity

In this section, we analyze the practicability of our construction regarding several properties. We evaluate the storage, communication overheads, and the computation cost of the different operations used by our construction. We also evaluate compare the sizes of the different keys to be used by the entities involved in our construction. For a better understanding of the evaluation results, we conduct a comparative analysis between our construction and TMACS [LXXH15]. Both schemes are achieving robustness while providing ABE-based fine grained access control. The notations we used in the complexity evaluation are described in Table 2.

7.1 Storage Complexity

In Table 3, we compare the storage complexity of our construction and TMACS on the different entities. Specifically, we quantify the storage complexity in terms of the size of elements (e.g., group elements, certificates, etc.) that are to be stored by each entity. Compared to TMACS, our construction requires less information to be stored by DCA and AA. In addition the size of an encrypted data item in our scheme is smaller than the one of TMACS. On the DU side, no more information needs to stored compared to TMACS. When comes to CSP, our construction requires the CSP to store a revocation key of size linear to the number of performed revocation operations. While the latter can be large in some use cases, we stress that the CSP is supposed to have unlimited storage space. Note that the TMACS scheme does not require the CSP to store any key as it does not allow access revocation.

Notation	Description
n	The number of AAs
t	Robustness level ($\leq n$)
N_Σ	The number of attributes in the system
N_u	The number of attribute held by a user
$N_\mathcal{U}$	The number of user in the system
$N_\mathcal{O}$	The number of data owner in the system
$N_\mathcal{I}$	Average data item size
l	The number of rows of the access matrix
$S_{\mathbb{G}_*}$	The size of an element in $\mathbb{G}_*$, $* \in \{0, 1, t\}$
$S_{\mathbb{Z}_p}$	The size of an element in $\mathbb{Z}_p$
π	Number of performed access revocation
$\mathcal{X}$	Set of data encrypted items
$N_{\mathcal{U},R}$	Number of users concerned by a revocation request
$N_{\Sigma,R}$	Number of attributes concerned by a revocation request
$\mathcal{M}$	A group exponentiation operation
$\mathcal{E}$	A pairing operation

Table 2: Notations used in the conducted evaluation

Entity		This work	TMACS
DCA		$(N_\mathcal{U} + n)S_{\mathbb{Z}_p}$	$(2N_\Sigma + 10)S_{\mathbb{G}_1} + (2N_\mathcal{U} + n)S_{\mathbb{Z}_p}$
AA		$(3 + 2N_\Sigma)S_{\mathbb{Z}_p}$	$(6 + 2N_\Sigma)S_{\mathbb{Z}_p}$
DP		-	-
DU		$(2 + N_u)S_{\mathbb{G}_1}$	$(2 + N_u)S_{\mathbb{G}_1}$
CSP	**Keys**	$\max_{\mathcal{C}\in\mathcal{X}}(\pi - \pi_\mathcal{C})S_{\mathbb{Z}_p}$	-
	Data	$(1 + 2l)S_{\mathbb{G}_2}$	$(1 + 2l)S_{\mathbb{G}_2} + S_{\mathbb{G}_t}$

Table 3: Comparison of the storage complexity

7.2 Communication Complexity

In Table 4, we provide a comparison of the theoretical communication complexities incurred by the different processes involved in our construction and TMACS. Specifically, we quantify the amount of information (in terms of the sizes of used group elements) exchanged by each entity during the different processes. As illustrate in the table, our construction does not include any communication overhead compared to TMACS, except for (i) AA during the setup process and (ii) DU during the data decryption process. However, we stress that both (linear) overheads can be tolerated since the first occurs only once during the setup phase, while the second permit the DU to considerably reduce the amount of computations required for data decryption (Table 5) by using the outsourced pre-decryption. Besides, our construction slightly reduces the communication complexity for DCA and CSP during the setup and the data decryption processes respectively.

Process	Entities	This work	TMACS
Setup	DCA	$\mathbf{(2n + 2N_\mathcal{U} + 2)S_{\mathbb{Z}_p}}$	$4N_\mathcal{U} + 4N_\mathcal{O} + 2nN_\Sigma$
	AA	$(5n+3)S_{\mathbb{Z}_p} + (N_\Sigma + 1)S_{\mathbb{G}_1} + (N_\Sigma + 2)S_{\mathbb{G}_2} + S_{\mathbb{G}_t}$	$\mathbf{(2n + 1)S_{\mathbb{Z}_p} + S_{\mathbb{G}_1}}$
	DU	$4S_{\mathbb{Z}_p}$	$4S_{\mathbb{Z}_p}$
	DO	$(2N_\mathbb{U} + 1)S_{\mathbb{G}_2}$	$(2N_\mathbb{U} + 1)S_{\mathbb{G}_2}$
Decryption Key Generation	AA	$(2 + N_u)S_{\mathbb{G}_1}$	$(2 + N_u)S_{\mathbb{G}_1}$
	DU	$(2 + N_u)tS_{\mathbb{G}_1}$	$(2 + N_u)tS_{\mathbb{G}_1}$
Data Decryption	DU	$(2 + N_u)S_{\mathbb{G}_1}$	**-**
	CSP	$\mathbf{S_{\mathbb{G}_t} + N_\mathcal{I}}$	$(2 + N_u)S_{\mathbb{G}_2} + N_\mathcal{I}$
Revocation	CSP	$(t + 1)S_{\mathbb{G}_2} + N_{\Sigma,R}(S_{\mathbb{G}_1} + S_{\mathbb{G}_2})$	N/A
	AA	$3S_{\mathbb{G}_2} + (2+N_u)(N_\mathcal{U} - N_{\mathcal{U},R})S_{\mathbb{G}_1}$	N/A
	DU	$(2 + N_u)tS_{\mathbb{G}_1}$	

The items in bold represents the smallest communication complexity of two compared schemes, the sign (-) is used to indicate that no communication are required by an entity, and we used (N/A) to indicate that the process is not supported by a scheme.

Table 4: Comparison of the communication complexity

7.3 Computation Complexity

In this section, we give a comparative analysis of the computational complexity of our scheme and TMACS. Table 5 shows, for the two constructions, the computational complexities of the different entities

of the system when performing the different processes. The processes complexities are quantifies mainly in terms of the number of group exponentiations and pairings. We note here that we ignore computations related to certificate generation and signature as they are performed by most existing ABE schemes.

Process	**Entities**	**This work**	**TMACS**
Setup	DCA	$4t\mathcal{E}$	$\boldsymbol{t\mathcal{E}}$
	AA	$(4+2N_{\Sigma})\mathcal{E}$	$\boldsymbol{\mathcal{E}}$
Decryption Key Generation	AA	$(2+2N_u)\mathcal{M}$	$(2+2N_u)\mathcal{M}$
	DU	$(2t+tN_u)\mathcal{M}$	$(2t+tN_u)\mathcal{M}$
Encryption	DO	$\mathcal{E}+(2l+1)\mathcal{M}$	$\mathcal{E}+(2l+1)\mathcal{M}$
Decryption	DU	$\boldsymbol{\mathcal{M}}$	$(1+2l)\mathcal{E}$
	CSP	$\boldsymbol{(1+2l)\mathcal{E}+(\pi-\pi_c)\mathcal{M}}$	-
Revocation	CSP	$(t+2N_{\Sigma,R})\mathcal{M}$	N/A
	AA	$3\mathcal{M}N$	N/A
	DU	$(2t+tN_u)\mathcal{M}$	N/A

The items in bold represents the smallest communication complexity of two compared schemes, the sign (-) is used to indicate that no communication are required by an entity, and we used (N/A) to indicate that the process is not supported by a scheme.

Table 5: Comparison of the computation complexity

The comparison shows that our construction does not include any computation overhead compared to TMACS for performing data encryption and decryption key generation processes. Thanks to the outsourcing pre-decryption, our construction drastically reduces the amount of computations required to be performed by DU for decrypting a data item. However, our construction requires linearly in the number of attributes more group exponentiations than TMACS to perform the setup process. Again, we stress that this later computation overhead can be tolerated as it occurs only once during the setup phase.

8 Conclusion

In this work, we propose the first CP-ABE fine-grained access control construction for outsourced data that enable the following features simultaneously. First, it ensures robustness for both decryption key issuing and access revocation while achieving forward secrecy. Second, it enables outsourced decryption to reduce the decryption overhead for data users that have limited computational resources. Third, it achieves adaptive (full) security in standard models. In the future, it will be interesting to investigate whether we can extend the construction to ensure backward secrecy.

References

[ADSLK19] Ruqayah R Al-Dahhan, Qi Shi, Gyu Myoung Lee, and Kashif Kifayat. Survey on revocation in ciphertext-policy attribute-based encryption. *Sensors*, 19(7):1695, 2019.

[BI92] Michael Bertilsson and Ingemar Ingemarsson. A construction of practical secret sharing schemes using linear block codes. In *International Workshop on the Theory and Application of Cryptographic Techniques*, pages 67–79. Springer, 1992.

[Boy08] Xavier Boyen. The uber-assumption family. In Steven D. Galbraith and Kenneth G. Paterson, editors, *Pairing-Based Cryptography – Pairing 2008*, pages 39–56, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE symposium on security and privacy (SP'07)*, pages 321–334. IEEE, 2007.

[CDLQ16] Hui Cui, Robert H Deng, Yingjiu Li, and Baodong Qin. Server-aided revocable attribute-based encryption. In *European Symposium on Research in Computer Security*, pages 570–587. Springer, 2016.

[EMN+09] Keita Emura, Atsuko Miyaji, Akito Nomura, Kazumasa Omote, and Masakazu Soshi. A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In *International Conference on Information Security Practice and Experience*, pages 13–23. Springer, 2009.

[GJPS08] Vipul Goyal, Abhishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute based encryption. In *International Colloquium on Automata, Languages, and Programming*, pages 579–591. Springer, 2008.

[HN11] Junbeom Hur and Dong Kun Noh. Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Transactions on Parallel and Distributed Systems*, 22(7):1214–1221, 2011.

[HSM+14] Jinguang Han, Willy Susilo, Yi Mu, Jianying Zhou, and Man Ho Au. Ppdcp-abe: Privacy-preserving decentralized ciphertext-policy attribute-based encryption. In *European Symposium on Research in Computer Security*, pages 73–90. Springer, 2014.

[HSMY12] Jinguang Han, Willy Susilo, Yi Mu, and Jun Yan. Privacy-preserving decentralized key-policy attribute-based encryption. *IEEE transactions on parallel and distributed systems*, 23(11):2150–2162, 2012.

[ISN89] Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72(9):56–64, 1989.

[KA+18] Praveen Kumar, PJA Alphonse, et al. Attribute based encryption in cloud computing: A survey, gap analysis, and future directions. *Journal of Network and Computer Applications*, 108:37–52, 2018.

[KW93] Mauricio Karchmer and Avi Wigderson. On span programs. In *[1993] Proceedings of the Eigth Annual Structure in Complexity Theory Conference*, pages 102–111. IEEE, 1993.

[LC10] Zhen Liu and Zhenfu Cao. On efficiently transferring the linear secret-sharing scheme matrix in ciphertext-policy attribute-based encryption. *IACR Cryptol. ePrint Arch.*, 2010:374, 2010.

[LCLS09] Xiaohui Liang, Zhenfu Cao, Huang Lin, and Jun Shao. Attribute based proxy re-encryption with delegating capabilities. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 276–286, 2009.

[LCLX09] Xiaohui Liang, Zhenfu Cao, Huang Lin, and Dongsheng Xing. Provably secure and efficient bounded ciphertext policy attribute based encryption. In *Proceedings of the 4th international symposium on information, computer, and communications security*, pages 343–352, 2009.

[LHC10] Song Luo, Jianbin Hu, and Zhong Chen. Ciphertext policy attribute-based proxy re-encryption. In *International Conference on Information and Communications Security*, pages 401–415. Springer, 2010.

[LML+16] Qi Li, Jianfeng Ma, Rui Li, Ximeng Liu, Jinbo Xiong, and Danwei Chen. Secure, efficient and revocable multi-authority access control system in cloud storage. *Computers & Security*, 59:45–59, 2016.

[LSW10] Allison Lewko, Amit Sahai, and Brent Waters. Revocation systems with very small private keys. In *2010 IEEE Symposium on Security and Privacy*, pages 273–285. IEEE, 2010.

[LW11] Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 568–588. Springer, 2011.

[LW16] Zhen Liu and Duncan S Wong. Practical attribute-based encryption: traitor tracing, revocation and large universe. *The Computer Journal*, 59(7):983–1004, 2016.

[LXXH15] Wei Li, Kaiping Xue, Yingjie Xue, and Jianan Hong. Tmacs: A robust and verifiable threshold multi-authority access control system in public cloud storage. *IEEE Transactions on parallel and distributed systems*, 27(5):1484–1496, 2015.

[LYH+17] Jiguo Li, Wei Yao, Jinguang Han, Yichen Zhang, and Jian Shen. User collusion avoidance cp-abe with efficient attribute revocation for cloud storage. *IEEE Systems Journal*, 12(2):1767–1777, 2017.

[NYO08] Takashi Nishide, Kazuki Yoneyama, and Kazuo Ohta. Attribute-based encryption with partially hidden encryptor-specified access structures. In *International conference on applied cryptography and network security*, pages 111–129. Springer, 2008.

[OSW07] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 195–203, 2007.

[Ped91] Torben Pryds Pedersen. A threshold cryptosystem without a trusted party. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 522–526. Springer, 1991.

[QZZC19] Baodong Qin, Qinglan Zhao, Dong Zheng, and Hui Cui. (dual) server-aided revocable attribute-based encryption with decryption key exposure resistance. *Information Sciences*, 490:74–92, 2019.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 457–473. Springer, 2005.

[TKN21] Junichi Tomida, Yuto Kawahara, and Ryo Nishimaki. Fast, compact, and expressive attribute-based encryption. *Designs, Codes and Cryptography*, 89(11):2577–2626, 2021.

[Wat11] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *International workshop on public key cryptography*, pages 53–70. Springer, 2011.

[Whi20] Danni White. Top 5 cloud computing predictions for 2020. https://www.techfunnel.com/information-technology/top-5-cloud-computing-predictions-for-2020/, 2020. Accessed: 2022-04-11.

[XHY+21] Hu Xiong, Xin Huang, Minghao Yang, Lili Wang, and Shui Yu. Unbounded and efficient revocable attribute-based encryption with adaptive security for cloud-assisted internet of things. *IEEE Internet of Things Journal*, 2021.

[YCTH18] Lo-Yao Yeh, Pei-Yu Chiang, Yi-Lang Tsai, and Jiun-Long Huang. Cloud-based fine-grained health information access control framework for lightweightiot devices with dynamic auditing andattribute revocation. *IEEE Transactions on Cloud Computing*, 6(2):532–544, 2018.

[YJR13a] Kan Yang, Xiaohua Jia, and Kui Ren. Attribute-based fine-grained access control with efficient revocation in cloud storage systems. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pages 523–528, 2013.

[YJR+13b] Kan Yang, Xiaohua Jia, Kui Ren, Bo Zhang, and Ruitao Xie. Dac-macs: Effective data access control for multiauthority cloud storage systems. *IEEE Transactions on Information Forensics and Security*, 8(11):1790–1801, 2013.

[YWRL10] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Attribute based data sharing with attribute revocation. In *Proceedings of the 5th ACM symposium on information, computer and communications security*, pages 261–270, 2010.

[ZDX+20] Yinghui Zhang, Robert H Deng, Shengmin Xu, Jianfei Sun, Qi Li, and Dong Zheng. Attribute-based encryption for cloud computing access control: A survey. *ACM Computing Surveys (CSUR)*, 53(4):1–41, 2020.