Robust and Provably Secure Attribute-Based Encryption Supporting Access Revocation and Outsourced Decryption

Anis Bkakria(✉)

IRT SystemX, Palaiseau, France
Anis.Bkakria@irt-systemx.fr

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: ABE · 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 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 [26], security tops the list of the biggest cloud challenges. Hence, users/companies are reluctant to outsource their important data to non fully-trusted Cloud servers.

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 [23] allows simultaneous confidentiality preservation and fine-grained access control. ABE has succeeded in attracting considerable research efforts [2,32] which allows defining 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 for issuing decryption keys to data 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 users at access time. Last but not least, the lack of security in standard models. We provide a full comparison with related literature in Sect. 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 enables attribute revocation while achieving forward secrecy. Third, it enables to outsource most part of the decryption process to the cloud server while ensuring that the latter learns 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 a theoretical comparison with similar constructions to show that ours introduces linear storage and computation overheads that occur 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 Sect. 6, we provide the security results. In Sect. 7, we discuss the complexity of our construction. Finally, Sect. 8 concludes.

2 Related Work

Revocable ABE. Several researchers have been devoted to build ABE constructions allowing access revocation. Liang et al. [15] introduce a provably selectively secure CP-ABE construction that enables access revocation through proxy re-encryption. Using the latter technique, Luo et al. [19] designed a selectively secure and small attribute universe based CP-ABE supporting policy updating. Always relying on proxy re-encryption, Yu et al. [31] propose an AND-gate policy based ABE construction enabling attribute and user revocation. 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. [10] propose a selectively secure in the standard model ABE construction that enables attribute revocation. Hur and Noh [8] rely on a stateless group key distribution method based on binary trees to define a CP-ABE solution enabling attribute revocation. The authors claimed that the proposed scheme achieves backward secrecy and forward secrecy without providing formal security analysis. Yang et al. [28] 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 [29], Yang et al. propose a multi-authority CP-ABE supporting revocation process. The latter is performed mainly by attribute authorities which are responsible for computing an updated decryption key for each non-revoked user. Relying on binary trees, Cui et al. [6] propose a CP-ABE scheme that enables attribute revocation where most of the computations are delegated to an untrusted server. Similarly to [8], the authors of [6] claim that their construction is both backward and forward secure without providing any formal proofs. Liu et al. [18] propose a large universe CP-ABE construction enabling both user revocation and accountability. The authors show that the proposed construction is selectively secure in standard models. Li et al. [13] propose a new multi-authority 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. [22] design an adaptively secure CP-ABE scheme enabling attribute revocation. In the proposed scheme, the untrusted server is used to help non-revoked users to decrypt ciphertexts. Very recently, Xiong et al. [27] propose an adaptively secure CP-ABE scheme allowing attribute revocation. It uses monotonic span program [9] as an access structure to reduce the number of pairing and exponentiation operations required for encryption and decryption.

Outsourced Decryption-Based ABE. To mitigate the burden of decryption for data users, Yang et al. [29] propose a construction that outsources most part of the computations to the cloud server. The same idea was later used in [6,22,27,30].

Robust ABE. A common weakness of the previously mentioned ABE constructions is that they all include a single point of failure on 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. To mitigate the previous threat, Li et al. [14] propose a multi-authority CP-ABE called TMACS. In contrast to previously mentioned approaches, in TMACS, the set of attribute authorities are collaboratively managing the whole set of attributes and no one of them can have full control over any specific attribute. The construction relies on a (t, n) threshold secret sharing protocol (see Section 3.2 of the extended version of this paper [4]) to require the collaboration of at least t attribute authorities to issue a valid decryption key, which allows to prove that the proposed construction is selectively secure even when $t-1$ authorities are compromised. Unfortunately, neither the access revocation, nor the outsourced decryption has been addressed in this work.

Table 1 presents a comprehensive feature comparison of the (most) 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.

Table 1. Feature comparaison of (most) related CP-ABE constructions

Approaches	**Robustness**	**Revocation**	**Security Model**	**Outsourced Decryption**
[7,16,20,21,23,25]	✗	✗	Selective	✗
[8,10,12,15,19,28,31]	✗	✓	Selective	✗
[6,29,30]	✗	✓	Selective	✓
[14]	✓	✗	Selective	✗
[3,11,24]	✗	✗	Fully	✗
[13,18]	✗	✓	Fully	✗
[22,27]	✗	✓	Fully	✓
This work	✓	✓	Fully	✓

3 Preliminaries

Definition 1 (Bilinear Maps). *Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be three multiplicative cyclic groups of prime order p. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map having the following properties:*

- *Symmetric bilinearity: $\forall g_1 \in \mathbb{G}_1, \forall g_2 \in \mathbb{G}_2$ and $\forall a, b \in \mathbb{Z}_p$, 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 our construction holds 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 $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e(\cdot,\cdot))$ as a bilinear environment.

Definition 2 (independence [5]). *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 $T = \langle t_1, \cdots, t_t \rangle \in \mathbb{F}_p[X_1, \cdots, X_c]^t$ 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 3 (GDHE assumption [5]**).** *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*

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

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

4 System and Security Models

In this section, we introduce the system 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 providers, data users, and a cloud server provider.

- The decentralized certificate authority (DCA) is a consortium blockchain-based PKI management system (e.g., [1]) which is responsible of setting up of the system by choosing its parameters such as, the set of attributes 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 for choosing the robustness level that should be satisfied, i.e., the number of attribute authorities that should be compromised to break the confidentiality of the shared data. 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.
- Cloud storage provider (CSP) provides data storage and computation capabilities such as 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 aims to access and use the shared data. A DU is labeled by a set of attributes. It is supposed to be able to download any encrypted data from the CSP. However, only DUs 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`, `DecKeyGen`, `PartialDecrypt`, `Decrypt`, and `Revoke`. 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 following 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 set of master public key shares $\{PK_i\}$ of the involved AAs and outputs a master public key MPK.
– `Encrypt`$(M, \mathbb{M}, MPK) \rightarrow \chi$ is a probabilistic algorithm that takes a message M and an access structure $\mathbb{M}$ and outputs an encrypted 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 outputs 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`$(\mathcal{SK}_{csp}, \overline{\mathcal{K}},\ ind) \rightarrow \overline{\chi}$ 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`$(\mathcal{K}, \overline{\chi}) \rightarrow M$ is a deterministic algorithm that takes as input a DU's secret decryption key $\mathcal{K}$ and a partially decrypted data item $\overline{\chi}$ and outputs a plaintext data item M.

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 a 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 aims 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 learn information about the shared data. Data consumers are considered 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 secrecy. The formalization of the following requirements is given in the full version of the paper [4].

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 attacks.

Robustness. According to the threat model we are considering, 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 long as no more than $t-1$ attribute authorities are compromised, with t denoting the robustness level.

Forward Secrecy. Forward secrecy is a mandatory property for enabling secure revocation. Forward secrecy requires that it should not be feasible for a data user to decrypt previous (non downloaded) and subsequent ciphertexts, if a subset of his/her attributes required for decryption are revoked.

5 Our Proposed Scheme

We now present the details of our construction. Since 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 former as $\mathcal{B}$.

5.1 System Initialization

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 two random elements of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Let $H : \mathbb{G}_T \rightarrow \{0,1\}^m$ be a cryptographic hash function for some $m(\geq 256)$ and Ξ be an unforgeable under adaptive chosen message attacks signature system. The DCA generates a signature key SMK and a verification key VMK and sends the pubic parameters $env = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e(\cdot,\cdot), g_1, g_2, H, \Xi, VMK)$ to $\mathcal{B}$ for storage.

System Initialization. The system is initialized using the following steps.

- **AA registration:** Each attribute authority AA$_i$ uses Ξ to generate 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 DCA. Then, the latter assigns a random identifier $cspid \in \mathbb{Z}_p$, issues a certificates $Cert_{cspid}$, and submits the couple $(cspid, Cert_{cspid})$ for storage in $\mathcal{B}$. Finally, CSP initializes its secret key as $\mathcal{SK}_{csp} = \emptyset$. The latter will be used to enforce access revocation by updating encrypted data items.
- **Robustness level selection:** Let us suppose that n attribute authorities $AA = \{A_1, \cdots, A_n\}$ are registered in the system. Using this process, DCA chooses the robustness level t $(t < n)$ that should be satisfied 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 (see [4], 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 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 for $k \in \{a, \alpha, \overline{\alpha}\}$: $sk_i^{\{k\}} = \sum_{j=1}^{n} s_{j,i}^{\{k\}}$, $pka_i = g_1^{sk_i^{\{a\}}}$, $\overline{pka_i} = g_2^{sk_i^{\{a\}}}$, $pke_i = e(g_1, g_2)^{sk_i^{\{\alpha\}}}$, and $pkr_i = \underline{g_2^{sk_i^{\{\overline{\alpha}\}}}}$. 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 of A_i is set as $SK_i = \{sk_i^{\{k\}}\}_{k \in \{a, \alpha, \overline{\alpha}\}}$.
  - 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 them 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 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.

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

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

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

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

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, DCA computes Θ_σ and $\overline{\Theta}_\sigma$ for all $\sigma \in \Sigma$ as $\Theta_\sigma = \prod_{i=1}^{n} \Theta_{\sigma,i}$ and $\overline{\Theta}_\sigma = \prod_{i=1}^{n} \overline{\Theta}_{\sigma,i}$. 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. The scalar π_p represents the timestep on which the public key is created/updated.

We emphasize here that the master key $MK = \{a, \alpha, \overline{\alpha}\}$ is implicitly shared among the different AAs. However, 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 item to CSP, similarly to most ABE schemes, the data to be shared M is firstly encrypted using a secure symmetric key algorithm (e.g., AES). The used symmetric key is generated then encrypted as we describe 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 on the data to be outsourced/shared. Then he/she executes the `Encrypt` algorithm which 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 $\kappa = H(pke^s) = H\left(e(g_1, g_2)^{\alpha \cdot s}\right)$. κ is then used to encrypt the data item to be shared M to get $E_\kappa(M)$.
– The data provider chooses the access policy to be enforced and transforms it into a Linear Secret Sharing Scheme (LSSS) access structure $(\mathbb{M}, \rho)$ as described in [17], 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, to make sure that only the keys that satisfy $(\mathbb{M}, \rho)$ will be able to compute κ, we hide the random element s used

to generate κ by choosing a random vector $\boldsymbol{v} = \{s, v_2, \cdots, v_k\} \in \mathbb{Z}_p^k$. For each row vector $\mathbb{M}_i$ of $\mathbb{M}$, $\lambda_i = \mathbb{M}_i \cdot \boldsymbol{v}^\top$ 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 $\mathcal{C} = \{C' = g_2^s,\ C_i = \overline{pka}^{\lambda_i} \cdot \overline{\Theta}_{\rho(i)}^{-r_i},\ D_i = g_2^{r_i}\}_{i\in[1,l]}$. Finally, the data owner sends the encrypted data item bundle $\chi = (E_\kappa(M), \mathcal{C}, \pi_c = \pi_p)$ to CSP. Similarly, π_c is used to denote the timestep on which the data item has been encrypted. At encryption, the timestep π_c is the same as the timestep associated to 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, to generate a secret decryption key, u_i performs the following steps:

– First, 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 here 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. 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 signed using $Cert_{uid}$ to A_j. Once the query is recieved by A_j, the latter 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 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 uses the MPK to generate a secret decryption key share for u_i as $\mathcal{K}_{u_i,j} = \{K_j = g_1^{sk_j^{\{\alpha\}}} \cdot pka^{b_j},\ L_j = g_1^{b_j},\ K_{\sigma,j} = \Theta_\sigma^{b_j}\}_{\sigma\in\Sigma_{u_i}^{(j)}}$.
– 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 $\mathcal{K}_u = \{K = g_1^{\alpha} \cdot g_1^{a \cdot d}, L = g_1^d, K_\sigma = \Theta_\sigma^d, \pi_u = \pi_p\}_{\sigma \in \Sigma_{u_i}}$ where π_u is used to denote the timestep on which $\mathcal{K}_u$ is issued.

We note here that the queried attribute authorities may assign different attributes $\Sigma_{u_i}^{(j)}$ to u_i. If it is the case, $\mathcal{K}_u$ will involve only the set of attributes $\Sigma_{u_i} = \cap_{j=0}^{t} \Sigma_{u_i}^{(j)}$ that are assigned by all t attribute authorities.

5.4 Access Revocation

Our construction aims to achieve robust fine-grained and on-demand access revocation. That is, to prevent a compromised AA from prohibiting authorized users from accessing the outsourced data by revoking their attributes, our construction requires the collaboration of t out of the n ($t > n/2$) registered attribute authorities to perform an access revocation. The revocation process is performed according to the following three steps.

Step 1: Collaborative Revocation Request. The access revocation process is triggered by an attribute authority A_i that wants to revoke the access granted by specific set of attributes $\Sigma_r \subset \Sigma$ 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 $Cert_{aid_i}$, 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, $j \in [1, n] \backslash \{i\}$ authenticates then processes the access revocation request. If it is legitimate, it computes the revocation request share as following $\mathcal{R}_j = \{R_j = R_b^{sk_j^{\overline{\alpha}}}, R_v = pkr^t, \Sigma_r, \mathcal{U}_r\}$ then, signs $\mathcal{R}_j$ and sends it to $\mathcal{B}$ for storage. Afterwards, each of the attribute authorities that participates to the previous collaborative revocation request updates locally the list of revoked user for each attribute as $\forall \sigma \in \Sigma_r : \mathcal{U}_{r,\sigma} = \mathcal{U}_{r,\sigma} \cup \mathcal{U}_r$.

Step 2: Revocation Enforcement. Once more than $t - 1$ shares for a revocation request are committed to $\mathcal{B}$, CSP computes

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

Thanks 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 $\overline{\alpha}$. Hence, if Formula (1) is equal to R_v, CSP will be sure that the access revocation is requested by at least t out of the n registered attribute authorities.

Then, CSP generates a random scalar $z \in \mathbb{Z}_p$ and updates the master public key MPK as $\forall \sigma \in \Sigma_r : \ \Theta_\sigma \leftarrow \Theta_\sigma^z, \overline{\Theta}_\sigma \leftarrow \overline{\Theta}_\sigma^z,$ and $\pi_p \leftarrow \pi_p + 1$ and updates its secret key $\mathcal{SK}_{csp}$ as $\mathcal{SK}_{csp} \leftarrow \mathcal{SK}_{csp} \cup \{\mu_{\pi_p} = z\}$ Finally, CSP sends the updated MPK for updation in $\mathcal{B}$.

Step 3: Secret Decryption Key Updating. Once an access revocation request is performed, each data user needs to request a fresh secret decryption key as described in Sect. 5.3.

5.5 Outsourced Pre-decryption and User Decryption

Our construction enables data users to shift expensive computations, i.e., pairing operations to CSP without disclosing any information about the encrypted data to CSP. Outsourced pre-decryption is performed according to the following steps.

```
    Input: $\mathcal{SK}_{csp}, \overline{\mathcal{K}_u}, MPK, ind$
    Output: $\overline{\kappa}, E_\kappa(M)$
 1  $(E_\kappa(M), \mathcal{C}, \pi_c) =$ get_item(ind)
 2  $\mathbb{I} = \{i : \rho(i) \in \Sigma_u\}$
 3  if $\pi_u < \pi_p$ then
 4      return exception(``outdated secret decryption key")
 5  end
 6  $\{C', C_i, D_i\}_{i \in [1,l]} = \mathcal{C}$
 7  if $\pi_c < \pi_p$ then
 8      z = 1
 9      for $i \in ]\pi_c, \pi_p]$ do
10          $z = z * \mu_i$
11      end
12      foreach $i \in [1, l]$ do
13          $D_i = D_i^z$
14      end
15      update_data_item(ind,$\mathcal{C}$, $\pi_p$))
16  end
17  $\overline{\kappa} = \frac{e(K', C')}{\prod_{i \in \mathbb{I}} \left( e(L', C_i) \cdot e(K_i', D_i) \right)^{w_i}}$
18  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 $\mathbb{M}$. Finally, let $\{w_i\}_{i \in \mathbb{I}}$ be constants such that $\sum_{i \in \mathbb{I}} w_i \cdot \mathbb{M}_i = (1, 0, \cdots, 0)$.

Step 1: Secret Decryption Key Randomization. A DU u starts by randomizing its secret decryption key as follows. He/She 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 item to be pre-decrypted to CSP.

Step 2: Outsourced Pre-decryption. After receiving ($\overline{\mathcal{K}_u}$, *ind*), CSP performs the `PartialDecrypt` algorithm which we detail in Algorithm 1. To be useful for pre-decrypting the data item, $\overline{\mathcal{K}_u}$ should include a set of attributes in 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, a 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 $E_\kappa(M)$.

Step 3: User Decryption. Once the user retrieves the pre-decrypted symmetric key $\overline{\kappa}$ and the ciphertext $E_\kappa(M)$, he/she recovers the symmetric key κ by computing $\kappa = H\left(\overline{\kappa}^{1/x}\right)$ 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 our construction provides a correct revocable fine-grained access control capabilities. Then, we prove the collusion resistance, the robustness and the forward secrecy properties in Theorems 2, 3, and 4 respectively. The proofs of the following theorems are provided in the full version of this paper [4].

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 $\mathcal{C}$.*

Theorem 2. *Our construction is collusion resistant under the GDHE assumption.*

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

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

7 The Complexity

In this section, we analyze the practicability of our construction regarding several properties. We evaluate the storage and communication overheads, as well as the computation cost of the different operations used by our construction. We also evaluate the sizes of the different cryptographic keys to be used by the

Table 2. Notations used in the conducted evaluation

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 attributes held by a user
$N_\mathcal{U}$	The number of users in the system
$N_\mathcal{O}$	The number of data owners 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 revocations
$\mathcal{X}$	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

entities involved in our construction. For a better understanding of the evaluation results, we conduct a comparative analysis between our construction and TMACS [14]. 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. In the sequel, we used (-) to indicate that no storage (resp. communication, resp. computation) is required by an entity, and (N/A) to indicate that a process is not supported by a scheme.

7.1 Storage Complexity

In Table 3, we compare the storage complexity of our construction and TMACS on the different involved entities. Specifically, we quantify the storage complexity in terms of the size of elements (e.g., group elements, certificates, etc.) that are

Table 3. Comparison of the storage complexity

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	$(1 + 2l)S_{\mathbb{G}_2}$	$(1 + 2l)S_{\mathbb{G}_2} + S_{\mathbb{G}_t}$

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 be stored compared to TMACS. When it comes to CSP, our construction requires the CSP to store a revocation key that grows linearly with 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 revocation key as it does not enable access revocation.

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 illustrated 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 permits 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.

Table 4. Comparison of the communication complexity

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

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 for the different entities of the system when performing the different processes. The processes' complexities are expressed 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.

Table 5. Comparison of the computation complexity

Process	**Entities**	**This work**	**TMACS**
Setup	DCA	$4t\mathcal{E}$	$t\mathcal{E}$
	AA	$(4+2N_{\Sigma})\mathcal{E}$	$\mathcal{E}$
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	$\mathcal{M}$	$(1+2l)\mathcal{E}$
	CSP	$(1+2l)\mathcal{E}+(\pi-\pi_{\mathcal{C}})\mathcal{M}$	-
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 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, it requires linearly in the number of attributes more group exponentiations than TMACS to perform the setup process. Again, we stress that this latter 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 enables 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 security in standard models.

References

1. Ahmed, A.S., Aura, T.: Turning trust around: smart contract-assisted public key infrastructure. In: 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), pp. 104–111. IEEE (2018)
2. Al-Dahhan, R.R., Shi, Q., Lee, G.M., Kifayat, K.: Survey on revocation in ciphertext-policy attribute-based encryption. Sensors **19**(7), 1695 (2019)
3. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (SP 2007), pp. 321–334. IEEE (2007)
4. Bkakria, A.: Robust, revocable and adaptively secure attribute-based encryption with outsourced decryption. Cryptology ePrint Archive, Report 2022/456 (2022), https://ia.cr/2022/456
5. Boyen, X.: The uber-assumption family. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 39–56. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85538-5_3
6. Cui, H., Deng, R.H., Li, Y., Qin, B.: Server-aided revocable attribute-based encryption. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9879, pp. 570–587. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45741-3_29
7. Han, J., Susilo, W., Mu, Y., Zhou, J., Au, M.H.: PPDCP-ABE: privacy-preserving decentralized ciphertext-policy attribute-based encryption. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8713, pp. 73–90. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11212-1_5
8. Hur, J., Noh, D.K.: Attribute-based access control with efficient revocation in data outsourcing systems. IEEE Trans. Parallel Distrib. Syst. **22**(7), 1214–1221 (2011)
9. Karchmer, M., Wigderson, A.: On span programs. In: [1993] Proceedings of the Eigth Annual Structure in Complexity Theory Conference, pp. 102–111. IEEE (1993)
10. Lewko, A., Sahai, A., Waters, B.: Revocation systems with very small private keys. In: 2010 IEEE Symposium on Security and Privacy, pp. 273–285. IEEE (2010)
11. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_31
12. Li, J., Yao, W., Han, J., Zhang, Y., Shen, J.: User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage. IEEE Syst. J. **12**(2), 1767–1777 (2017)
13. Li, Q., Ma, J., Li, R., Liu, X., Xiong, J., Chen, D.: Secure, efficient and revocable multi-authority access control system in cloud storage. Comput. Secur. **59**, 45–59 (2016)
14. Li, W., Xue, K., Xue, Y., Hong, J.: TMACS: a robust and verifiable threshold multi-authority access control system in public cloud storage. IEEE Trans. Parallel Distrib. Syst. **27**(5), 1484–1496 (2015)
15. Liang, X., Cao, Z., Lin, H., Shao, J.: Attribute based proxy re-encryption with delegating capabilities. In: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, pp. 276–286 (2009)
16. Liang, X., Cao, Z., Lin, H., Xing, D.: Provably secure and efficient bounded ciphertext policy attribute based encryption. In: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, pp. 343–352 (2009)

17. Liu, Z., Cao, Z.: On efficiently transferring the linear secret-sharing scheme matrix in ciphertext-policy attribute-based encryption. IACR Cryptol. ePrint Arch. **2010**, 374 (2010)
18. Liu, Z., Wong, D.S.: Practical attribute-based encryption: traitor tracing, revocation and large universe. Comput. J. **59**(7), 983–1004 (2016)
19. Luo, S., Hu, J., Chen, Z.: Ciphertext policy attribute-based proxy re-encryption. In: Soriano, M., Qing, S., López, J. (eds.) ICICS 2010. LNCS, vol. 6476, pp. 401–415. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17650-0_28
20. Nishide, T., Yoneyama, K., Ohta, K.: Attribute-based encryption with partially hidden Encryptor-specified access structures. In: Bellovin, S.M., Gennaro, R., Keromytis, A., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 111–129. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68914-0_7
21. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 195–203 (2007)
22. Qin, B., Zhao, Q., Zheng, D., Cui, H.: (dual) server-aided revocable attribute-based encryption with decryption key exposure resistance. Inf. Sci. **490**, 74–92 (2019)
23. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
24. Tomida, J., Kawahara, Y., Nishimaki, R.: Fast, compact, and expressive attribute-based encryption. Des. Codes Cryptogr. **89**(11), 2577–2626 (2021). https://doi.org/10.1007/s10623-021-00939-8
25. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_4
26. White, D.: Top 5 cloud computing predictions for 2020 (2020). https://www.techfunnel.com/information-technology/top-5-cloud-computing-predictions-for-2020/.Accessed 11 Apr 2022
27. Xiong, H., Huang, X., Yang, M., Wang, L., Yu, S.: Unbounded and efficient revocable attribute-based encryption with adaptive security for cloud-assisted internet of things. IEEE Internet Things J. **9** (2021)
28. Yang, K., Jia, X., Ren, K.: 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, pp. 523–528 (2013)
29. Yang, K., Jia, X., Ren, K., Zhang, B., Xie, R.: DAC-MACS: effective data access control for multiauthority cloud storage systems. IEEE Trans. Inf. Foren. Secur. **8**(11), 1790–1801 (2013)
30. Yeh, L.Y., Chiang, P.Y., Tsai, Y.L., Huang, J.L.: Cloud-based fine-grained health information access control framework for LightweightIoT devices with dynamic auditing andattribute revocation. IEEE Trans. Cloud Comput. **6**(2), 532–544 (2018). https://doi.org/10.1109/TCC.2015.2485199
31. Yu, S., Wang, C., Ren, K., Lou, W.: Attribute based data sharing with attribute revocation. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, pp. 261–270 (2010)
32. Zhang, Y., Deng, R.H., Xu, S., Sun, J., Li, Q., Zheng, D.: Attribute-based encryption for cloud computing access control: a survey. ACM Comput. Surv. **53**(4), 1–41 (2020)