SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

April 2016, Vol. 59 042701:1–042701:16
doi: 10.1007/s11432-015-5428-1

Adaptively secure ciphertext-policy attribute-based encryption with dynamic policy updating

Zuobin YING*, Hui LI, Jianfeng MA, Junwei ZHANG & Jiangtao CUI

School of Computer Science and Technology, Xidian University, Xi'an 710071*, China*

Received June 1, 2015; accepted July 7, 2015; published online February 26, 2016

Abstract Attribute-Based Encryption (ABE) is a promising new cryptographic technique which guarantees fine-grained access control of outsourced encrypted data in the cloud. With the help of ABE, the majority of security issues in accessing cloud data can be solved. However, a key limitation remains, namely policy updating. Whenever the access policy is updated, a common approach is to have the data owner retrieve the data and re-encrypt it with new policy, before sending the new ciphertext back to the cloud. This straight-forward approach will lead to heavy computation and communication overhead. Although a number of other approaches have been proposed in this regard, they suffer from two limitations; namely, supporting only limited update-policy types or having weak security models. In order to address these limitations, we propose a novel solution to the attribute-based encryption access control system by introducing a dynamic policy-updating technique which we call DPU-CP-ABE. The scheme is proved to be adaptively secure under the standard model and can support any type of policy updating. In addition, our scheme can significantly reduce the computation and communication costs of updating ciphertext.

Keywords attribute-based encryption, ciphertext-policy, dynamic policy updating, adaptive secure, standard model

Citation Ying Z B, Li H, Ma J F, et al. Adaptively secure ciphertext-policy attribute-based encryption with dynamic policy updating. Sci China Inf Sci, 2016, 59(4): 042701, doi: 10.1007/s11432-015-5428-1

1 Introduction

Since proposed in [1], Attribute-Based Encryption (ABE) [1], as an ingenious extension of Identity-based encryption (IBE) by introducing the notion of access policy, has been widely studied and applied in cloud computing systems. By defining an ABE access policy, data owners can easily determine which group of users are authorized to access encrypted data that is stored in the cloud. Thus, any data can be stored in the cloud without the owner needing to worry about the threats of privacy leakage, malicious users, or even the cloud itself. Traditional ABE has two variants according to the form of access policy; namely, Key-Policy ABE (KP-ABE) [2] and Ciphertext-Policy ABE (CP-ABE) [3]. CP-ABE, which offers the data owner a scalable method of encrypting the data, is more appropriate for constructing a data-outsourcing system than KP-ABE, since the data owner is able to define the access policy [4]. The core idea of using CP-ABE in a cloud environment can be summarized by Steps (1) and (2) in Figure 1.

* Corresponding author (email: zbying@mail.xidian.edu.cn)

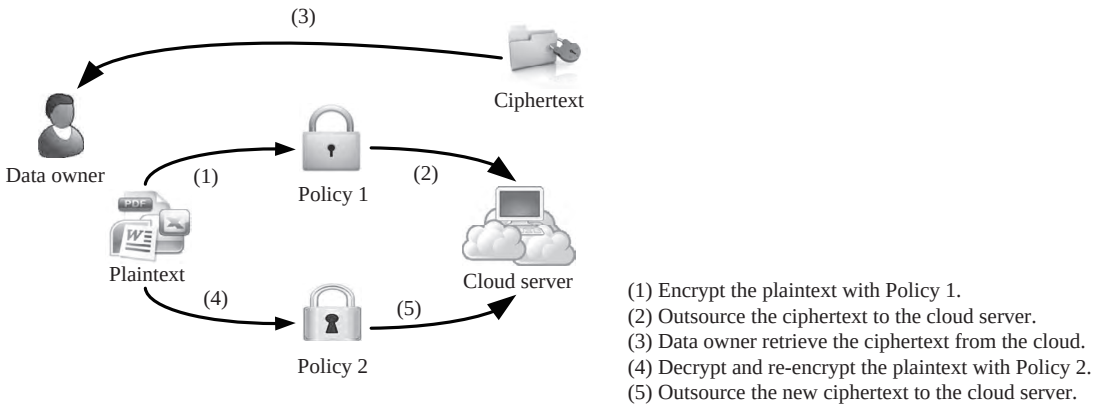

Figure 1 CP-ABE scheme with a straight-forward implementation of policy updating.

However, the traditional CP-ABE scheme suffers from a key limitation in that it does not support policy changing. In fact, access policies, outsourced along with big data to the cloud by millions of organizations are frequently changed or dynamically adjusted by the data owners because of personnel changes, emergency situations, etc. Consider the following examples: (1) Without an access-policy change, Bob, the former Chief Technology Officer of enterprise A, can still visit all sensitive files, which may be encrypted using ABE, because he was previously authorized to do so, even though he has since moved to enterprise B. This could lead to serious trade-secret or proprietary information leakages. (2) In nationwide joint plague-prevention operations, medical institutions and international medical organizations must cooperate in order to prevent the plagues proliferation. Therefore, some confidential investigation and research files as well as guidance, which may be encrypted using ABE under pre-defined access policies, may urgently need to be opened to these organizations for a period of time.

A straight-forward implementation of handling these scenarios is to execute Steps (3)–(5) in Figure 1. Step (3) allows the data owners to retrieve the encrypted data from the cloud, Step (4) re-encrypts the data under a new policy, and Step (5) sends the ciphertext back to the cloud. Unfortunately, this approach will cause tremendous network communication overhead, and a heavy computational burden on the data owner as well [5].

Therefore, developing an ABE access-control system with policy updating has emerged as a key issue in cloud computing. However, most existing access-control schemes based upon ABE [4, 6–8] do not take into account the policy-updating issue. The prototype of the policy-updating scheme originates from the delegation of private keys in [2] and the delegation of ciphertext in [9]. Nevertheless, these methods can only support a situation where the new policy is more restrictive than the current one. Yang et al. [5] proposed a new approach to reconstruct all types of policies and presented an efficient access-control system with a dynamic policy-updating mechanism. However, their work was not proved under a sufficiently secure model.

In this paper, we put forward a novel approach, namely Adaptively Secure Ciphertext-Policy Attribute-Based Encryption with Dynamic Policy Updating (DPU-CP-ABE), to solve the aforementioned policy-updating problems in a data-outsourcing system using CP-ABE access control under the standard security model. Our main idea is to maximize the usage of the existing ciphertext as well as the current policy, and minimize the data owners computation cost by delegating the ciphertext-updating task to the cloud server. Meanwhile, we ensure the security of our scheme, which is adaptively secure under the standard model. In our scheme, whenever data owners want to update the policy, they only need to submit a new policy-updating query; then, the cloud server updates the ciphertext without decrypting it. However, the updating procedure should meet the following requirements.

(1) The system should support any type of policy updating.

(2) The system should be fully secure under standard model, since the cloud is an open and complicated environment.

(3) The updating procedure should not lead to supererogatory information leakage.

1.1 Contributions

In this paper, we describe a novel data-outsourcing system using CP-ABE access control with efficient dynamic policy updating. Our contributions can be briefly outlined as follows.

(1) To our knowledge, our scheme is the first CP-ABE access control system with dynamic policy updating that is proved to be adaptively secure under the standard model. The complete security proof is presented in the appendix.

(2) Our policy-updating scheme supports the updating of any type of fine-grained policy, as compared to the ciphertext delegation method [9] which can only re-encrypt the ciphertext under a more restrictive policy. Moreover, although the policy-updating task is outsourced to the cloud, the updating procedure will leak no sensitive data to the cloud server. A detailed proof can be found in the appendix.

(3) We design a dynamic policy updating algorithm for LSSS (Linear Secret-Sharing Schemes) access structure using CP-ABE access control, which is more efficient than the straight-forward policy-updating implementation.

1.2 Related work

Sahai and Waters [1] pioneered the study of Attribute-Based Encryption (ABE). According to their original ABE scheme, data owners are able to encrypt the data under a set of attributes ω. The ciphertext can only be decrypted by attribute sets ω' where $|\omega \cap \omega'| \geqslant d$, where d is the preset threshold value. After that, by introducing an access-control mechanism, there are two branches of ABE, namely Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE). In a KP-ABE scheme [2], ciphertexts are annotated by attributes, and access polices over these attributes are associated with users' private keys. However, the access policies and the attributes trade places in a CP-ABE scheme [3], i.e., access policies are associated with the ciphertext while the users private key is combined with the attributes. Moreover, the decryption can be processed successfully iff the users attributes match the access policy set by the data owner. Waters redescribed the CP-ABE scheme using the LSSS access structure in [10].

Note that the schemes mentioned above are all single-authority ABEs. In reality, different attributes are organized and managed according to their characteristics by different authorities. Chase [11] was the first to present the multi-authority ABE system, in which multiple AAs (Attribute Authorities) and a single CA (Central Authority) could be found. The CA releases identity-related keys, while each AA is responsible for a subset of attributes and releases attribute-related keys. The correctness of reconstructing an attribute-related key of one user from different AAs is guaranteed by the *global identifier* (*gid*). In addition, the *gid* can prevent different users from colluding. However, the CA must be fully trusted, and the scheme only supports an "**AND**" policy between AAs. The first fully secure multi-authority ABE scheme was introduced by Lewko and Waters [6], which was proved to be secure under random oracle model. Liu et al. [7] developed an improved multi-authority CP-ABE scheme in which multiple CAs as well as AAs could be found. The keys, related to identity, are released by CAs. However, the CAs do not participate in any operations which are related to attributes. The scheme is adaptively secure under the standard model.

These studies enrich the ABE theory and solve some of the problems. However, policy updating remains a challenging issue. Goyal et al. [2] were the first to take into account the policy-updating issue in the key-policy structure. Recently, Sahai et al. [9] discussed the policy-updating problem under the ciphertext-policy structure. Unfortunately, the policy-updating methods of these systems were limited by the notion of delegation, which means that the new policy should be more restrictive than the current one. Yang et al. [5] proposed a new efficient dynamic policy-updating scheme which could update all types of policies. They proved their scheme was secure under the general group model, and stated that, by using composite-order groups, the secure model could be extended to the random oracle model. However, they failed to provide the related proof of security.

In summary, state-of-the-art approaches towards policy updating in ABE either support only limited update-policy types or work under a weak security model. In contrast, our work in this paper addresses both of these limitations simultaneously.

Organization. The remainder of the paper proceeds as follows. In Section 2, we outline some preliminaries, including the framework, security model, access policy, and number-theoretic assumptions. Section 3 presents our DPU-CP-ABE construction. The comparison between existing schemes and ours is presented in Section 4 and the conclusion of the paper is in Section 5. In the appendix, we provide the entire security proof of our DPU-CP-ABE scheme.

2 Preliminaries

Before describing DPU-CP-ABE scheme, we briefly outline the framework and technique preliminaries. For definitions of the composite order bilinear group and number-theoretic assumptions, see Appendix A.

2.1 Framework

Our DPU-CP-ABE system is a collection of the nine algorithms listed below:

GlobalSetup$(\lambda) \rightarrow \mathsf{GLPK}$. This algorithm inputs the security parameter λ and outputs the system global parameter GLPK.

CenterAuthSetup$(\mathsf{GLPK}, c) \rightarrow (\mathsf{CPK}_c, \mathsf{CVK}_c, \mathsf{CMK}_c)$. Each CA_c runs this algorithm with the input GLPK along with index c. The outcomes are master secret key CMK_c which is used by each CA_c, and public key pairs $(\mathsf{CPK}_c, \mathsf{CVK}_c)$. AAs will use CVK_c as the authentication key when communicating with the CAs.

AuthSetup$(\mathsf{GLPK}, u, Q_u) \rightarrow (\mathsf{APK}_u, \mathsf{AVK}_u, \mathsf{AMK}_u)$. Each AA_u runs this algorithm with the inputs of GLPK, index u and the attributes domain Q_u. It generates the master secret key AMK_u and public key pairs $(\mathsf{APK}_u, \mathsf{AVK}_u)$. CAs will use AVK_u as the authentication key when communicating with the AAs.

CaKeyGen$(gid, \mathsf{GLPK}, \{\mathsf{AVK}_u | u \in \mathbb{U}\}, \mathsf{CMK}_c) \rightarrow (\mathsf{csk}_{\mathrm{gid},c}, \mathsf{cpk}_{\mathrm{gid},c})$. gid stands for the global identifier. CMK_c refers to the master secret key of CA_c. When a user with a gid accesses CA_c to request the corresponding key, CA_c runs this algorithm. By taking $gid, \mathsf{GLPK}, \{\mathsf{AVK}_u \mid u \in \mathbb{U}\}, \mathsf{CMK}_c$ as input, this algorithm outputs the user-ca-key$(csk_{\mathrm{gid},c}, cpk_{\mathrm{gid},c})$. Here, $cpk_{\mathrm{gid},c}$ is called user-ca-pub-key. and is the users public key. $csk_{\mathrm{gid},c}$ is the users private key.

AuthKeyGen$(att, \{\mathsf{cpk}_{\mathrm{gid},c} | c \in \mathbb{C}\}, \mathsf{GLPK}, \mathsf{AMK}_u, \{\mathsf{CVK}_c | c \in \mathbb{C}\}) \rightarrow \mathsf{attk}_{\mathrm{att,gid}}$ or $\square$. att defines an attribute. When a user queries AA_u to obtain a secret key for att, AA_u executes this algorithm. Through inputting $att, \{\mathsf{cpk}_{\mathrm{gid},c} | c \in \mathbb{C}\}, \mathsf{GLPK}, \{\mathsf{CVK}_c | c \in \mathbb{C}\}, \mathsf{AMK}_u$, it outputs an user-att-key $\mathsf{attk}_{\mathrm{att,gid}}$ iff all $cpk_{\mathrm{gid},c}$s are valid; otherwise $\square$ is output to indicate the users public key is invalid. When a user with a gid obtains the attribute set A_{gid}, the user's dec-key can be established as $\mathsf{UDK}_{\mathrm{gid}} = (\{csk_{\mathrm{gid},c}, cpk_{\mathrm{gid},c} | c \in \mathbb{C}\}, \{\mathsf{attk}_{\mathrm{att,gid}} | att \in A_{\mathrm{gid}}\})$.

Encrypt$(m, \mathbb{A}, \mathsf{GLPK}, \{\mathsf{CPK}_c | c \in \mathbb{C}\}, \{\mathsf{APK}_u\}) \rightarrow \mathsf{CT}, \mathsf{En\text{-}para}(m)$. This algorithm takes a message m, an access policy $\mathbb{A}$ established upon the global attribute sets Q, GLPK, CAs' public key sets $\{\mathsf{CPK}_c\}$, as well as the relevant AAs' public key sets $\{\mathsf{APK}_u\}$ as input. The ciphertext CT and its policy $\mathbb{A}$ as well as the encryption information $\mathsf{En\text{-}para}(m)$ are the outputs.

Decrypt$(\mathsf{CT}, \mathsf{GLPK}, \{\mathsf{APK}_u\}, \mathsf{UDK}_{\mathrm{gid}}) \rightarrow m$ or $\square$. This algorithm inputs a ciphertext CT as well as its policy $\mathbb{A}, \mathsf{GLPK}$, the relevant AAs' public parameters $\{\mathsf{APK}_u\}$, a user's dec-key and the related A_{gid}. It outputs either m when $_{\mathrm{gid}}$ fits $\mathbb{A}$ or $\square$ otherwise, which indicates the failure of decryption.

DPUKeyGen$(\{\mathsf{CPK}_c | c \in \mathbb{C}\}, \{\mathsf{APK}_u\}, \mathsf{En\text{-}para}(m), \mathbb{A}, \mathbb{A}') \rightarrow \mathsf{DPUK}_m$. The data owner runs this algorithm by inputting the related public key $\mathsf{APK}_u, \{\mathsf{CPK}_c | c \in \mathbb{C}\}$, the encryption information $\mathsf{En\text{-}para}(m)$ of m, the current policy $\mathbb{A}$ and the new policy $\mathbb{A}'$ to be updated to. It outputs the DPUK_m which will be used to update the ciphertext CT.

CTUpdate$(\mathsf{CT}, \mathsf{DPUK}_m) \rightarrow \mathsf{CT}'$. The cloud server runs this algorithm by taking as input ciphertext CT and the relevant access policy updating key DPUK_m. The CT along with its access policy $\mathbb{A}$ will be updated to CT' and $\mathbb{A}'$ correspondingly.

2.2 Security model

We describe the assumptions for the participants before the security model is given. The cloud is assumed to be curious about the data it stores, but will do exactly what the data owner requests and will not collude with the users. Thus, users cannot obtain ciphertexts encrypted under the previous access policies so as to derive the encryption component $e(g,g)^s$. The data owners are fully trusted.

Now we describe the security model of DPU-CP-ABE by defining the following game, taken place between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$. $\mathcal{A}$ is able to corrupt CAs and AAs after seeing the public parameters. The corrupted CA and AA are denoted as $\mathbb{C}_{\rm cr} \subset \mathbb{C}$ and $\mathbb{U}_{\rm cr} \subset \mathbb{U}$, respectively, where $\mathbb{C}\backslash\mathbb{C}_{\rm cr} \neq \emptyset$, $\mathbb{U}\backslash\mathbb{U}_{\rm cr} \neq \emptyset$. Without losing anything general, we presume that all CAs are corrupted by $\mathcal{A}$ except one (i.e., $\mathbb{C}\backslash\mathbb{C}_{\rm cr} = 1$).

Setup. Challenger $\mathcal{B}$ runs GlobalSetup, CenterAuthSetup and AuthSetup algorithms and sends GLPK, $\{\mathsf{CPK}_c, \mathsf{CVK}_c | c \in \mathbb{C}\}$, $\{\mathsf{APK}_u, \mathsf{AVK}_u | u \in \mathbb{U}\}$ to the adversary $\mathcal{A}$. After seeing the public parameters, $\mathcal{A}$ specifies an index $\bar{c}^* \in \mathbb{C}$ to be the only uncorrupted CA, so $\bar{c}^* = \mathbb{C}\backslash\mathbb{C}_{\rm cr}$. Then, $\mathcal{A}$ is given the master secret key of $\{\mathsf{AMK}_u | u \in \mathbb{U}_{\rm cr}\}$ and $\{\mathsf{CMK}_c | c \in \mathbb{C}_{\rm cr}\}$.

Key Query Phase 1. $\mathcal{A}$ can question the following oracles to obtain user-ca-key and user-att-key:

$\mathsf{CA\text{-}KQ}(gid, \bar{c}^*)$: $\mathcal{A}$ submits queries with a pair $(gid, \bar{c}^*)$ and acquires the corresponding user-ca-key as $(csk_{\rm gid,\bar{c}^*}, cpk_{\rm gid,\bar{c}^*})$.

$\mathsf{AA\text{-}KQ}(att, \{cpk_{\rm gid,c} | c \in \mathbb{C}\}, \bar{u^*})$: $\mathcal{A}$ submits queries with a tuple $(att, \{cpk_{\rm gid,c}\} | c \in \mathbb{C}\}, \bar{u^*})$ where att is an attribute in $Q_{\bar{u^*}}$, $\{cpk_{\rm gid,c}\} | c \in \mathbb{C}\}$ are user-ca-public-key of the user gid, $\bar{u^*} = \mathbb{U}\backslash\mathbb{U}_{\rm cr}$ is the index of the only uncorrupted AA. Oracle $\mathsf{AA\text{-}KQ}$ replies with the corresponding user-att-key or $\square$ depending on whether $\{cpk_{\rm gid,c}\}$ are valid or not.

Challenge. $\mathcal{A}$ submits two messages (m_0 and m_1) of equal length. Additionally, $\mathcal{A}$ also provides a set of challenge access structures $\{(M_1^*, \rho_1^*), \ldots, (M_k^*, \rho_k^*)\}$. The challenger $\mathcal{B}$ flips a random coin $b \in \{0,1\}$ and encrypts m_b under all access structures $\{(M_1^*, \rho_1^*), \ldots, (M_k^*, \rho_k^*)\}$, then sends the ciphertext $\{(\mathsf{CT}_1^*), \ldots, (\mathsf{CT}_k^*)\}$ to $\mathcal{A}$.

Key Query Phase 2. Moreover, the adversary may submit some other queries as it did in **Key Query Phase 1.** The adversary can also query for the update key by submitting (M_x^*, ρ_x^*), (M_y^*, ρ_y^*). $\mathcal{B}$ replies with update key DPUK_{m_b}.

Guess. The adversary outputs a guess b' of b.

For a gid, $\mathcal{A}$ defined the relevant attribute set as $A_{\rm gid} = \{att | \mathsf{AA\text{-}KQ}(att, cpk_{\rm gid,c} | c \in \mathbb{C}), \bar{u^*}\}$.

The adversary wins the above game if $b' = b$ under the constraint that there exists no $A_{\rm gid}$ so that $A_{\rm gid} \cup (\bigcup_{u_{\rm cr} \in \mathbb{U}_{\rm cr}} Q_{u_{\rm cr}})$ can make the challenge access policy set $\{(M_1^*, \rho_1^*), \ldots, (M_k^*, \rho_k^*)\}$ satisfied. The adversary $\mathcal{A}$'s advantage is defined as $|\Pr[b' = b] - \frac{1}{2}|$.

Definition 1. A DPU-CP-ABE system is secure given that no poly-time adversaries have a non-negligible advantage in the above security game[1)].

The main differences between DPU-CP-ABE security model and the state-of-the-art ones exist in the following two aspects. Firstly, in the **Challenge phase**, since the policy may be adjusted, $\mathcal{A}$ can release a set of challenge access policies instead of only one policy, under the restriction by which the decryption is not allowed. Moreover, all corresponding ciphertexts will be given to the adversary. Secondly, in **Key Query Phase 2**, the adversary is able to query and obtain the update key in order to gain more advantages. We further prove that this attempt is helpless in breaking our scheme.

Remark 1. To simplify the description, we make an assumption that a user with gid can only have a set of user-ca-keys at one time (i.e., a user can submit requests to each CA_c for the user-ca-key only once)[2)]. In the above model, since the adversary $\mathcal{A}$ has all the master secret keys $\{\mathsf{CMK}_c | c \in \mathbb{C}_{\rm corrupt}\}$, the user only needs to submit queries to the $\mathsf{CA\text{-}KQ}$ $(gid, \bar{c}^*)$ to get user-ca-key $(csk_{\rm gid,\bar{c}^*}, cpk_{\rm gid,\bar{c}^*})$, and they can query the $\mathsf{AA\text{-}KQ}$ oracle to obtain $(csk_{\rm gid,c}, cpk_{\rm gid,c} | c \in \mathbb{C}_{\rm corrupt})$ if they are needed.

1) We prove that our system is secure in the appendix.

2) In fact, we can remove this assumption by introducing a time stamp to form $cpk_{\rm gid,c,t}$ as well as $A_{\rm gid,c,t}$, in which case the description will be much more complex.

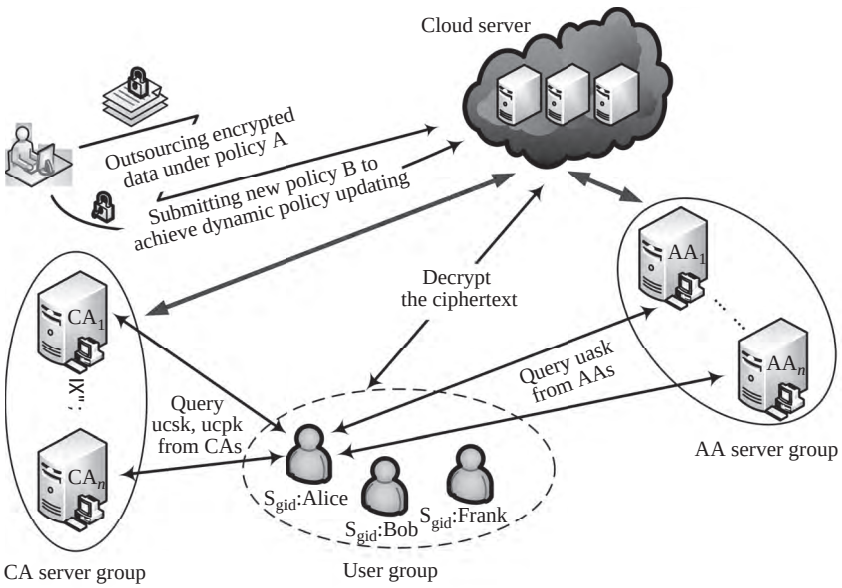

Figure 2 DPU-CP-ABE system model under consideration.

2.3 Access policy

Definition 2 (Access Structure [12]). Let $\mathbb{P} = \{P_i\}$ $(i = 1, \ldots, n)$ denote a set of participants. Given a collection $\mathbb{A} \subseteq 2^{\mathbb{P}}$, if $\forall B, C$, $B \in \mathbb{A}$ and $B \subseteq C$, we have $C \in \mathbb{A}$, then $\mathbb{A}$ is monotone. An access structure (resp., monotone access structure) is a collection (resp., monotone collection) $\mathbb{A}$ of non-empty subsets $\mathbb{P}$ (i.e., $\mathbb{A} \subseteq 2^{\mathbb{P}} \backslash \{\emptyset\}$). We designated the sets in $\mathbb{A}$ (resp., not in $\mathbb{A}$) as authorized (resp., unauthorized) sets.

In ABE schemes, attributes function as participants. Therefore, access structure $\mathbb{A}$ can only be satisfied in terms of authorized sets. In this context, we concentrate on monotone access structures. Previous work [10, 12] demonstrated how to convert a monotone access structure into linear secret sharing scheme. The definition of LSSS is provided as follows.

Definition 3 (Linear Secret-Sharing Schemes (LSSS) [10]). Linear (over $\mathbb{Z}_p$) in nature is a secret-sharing scheme Π defined by a set of participants $\mathcal{P}$ provided that

1. The shares for every single participant form a vector over $\mathbb{Z}_p$.

2. There exists a matrix M (with l rows and n columns), which is named as share-generating matrix for Π. For $i = 1, \ldots, l$, use function $\rho(i)$ to define the ith row of M as a party. As far as we reckon the column vector $\boldsymbol{v} = (s, r_2, \ldots, r_n)$, in which $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are selected at random, then $M\boldsymbol{v}$ is the vector of l shares of the secret s in terms of Π. The share $(M\boldsymbol{v})_i$ is a part of participant $\rho(i)$.

As illustrated in [12], every linear secret sharing-scheme which, in accordance with the definition mentioned above, exhibits a character of linear reconstruction, can be defined as follows. Presume that Π is an access structure $\mathbb{A}$. $S \in \mathbb{A}$ is an authorized set, making $I = \{i : \rho(i) \in S\}$ will ensure the existence constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ for which $\sum_{i \in I} \omega_i \lambda_i = s$, if $\{\lambda_i\}$ can be accounted as valid shares of any secret value s in respect of Π.

3 Our DPU-CP-ABE

In our DPU-CP-ABE scheme, DPUKeyGen and CTUpdate are specifically designed to solve the access policy updating problem. Since the remaining algorithms have been discussed in [7], our scheme can be considered as an extension from [5, 7]. The DPU-CP-ABE scheme is constructed in composite order bilinear groups of order $N = p_1 p_2 p_3$ (3 distinct primes). We demonstrate the system in Figure 2 and detail the phases as follows.

GlobalSetup$(\lambda) \rightarrow \mathsf{GLPK}$. Denote G as a bilinear group with order $N = p_1p_2p_3$, G_{p_i} function as the subgroup in G with the corresponding order p_i. Select $g, h \in G_{p_1}$ at random, where $h = g^a$. Denote the generator of G_{p_3} as I_3. The global public key is organized as $\mathsf{GLPK} = (g, h, N, I_3, \Omega_{\mathrm{Sign}})$, $\Omega_{\mathrm{Sign}} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ is a secure signature subsystem called unforgeable under adaptive chosen message attacks (UF-CMA) [13], which will be used to defend the potential collusion attack.

CenterAuthSetup$(\mathsf{GLPK}, c) \rightarrow (\mathsf{CPK}_c, \mathsf{CVK}_c, \mathsf{CMK}_c)$. CA_c runs KGen algorithm of Ω_{Sign} to get signature verifying key pairs $(SignKey_c, VerifyKey_c)$, randomly pick an exponent $\alpha_c \in \mathbb{Z}_N$. CA_c releases the public parameters $\mathsf{CPK}_c = e(g,g)^{\alpha_c}, \mathsf{CVK}_c = VerifyKey_c$ and the master secret key $\mathsf{CMK}_c = (\alpha_c, SignKey_c)$.

AuthSetup$(\mathsf{GLPK}, u, Q_u) \rightarrow (\mathsf{APK}_u, \mathsf{AVK}_u, \mathsf{AMK}_u)$. For each $att \in Q_u$, AA_u chooses $s_{\mathrm{att}} \in \mathbb{Z}_N$ randomly and sets $T_{\mathrm{att}} = g^{s_{\mathrm{att}}}$. Besides, for each $c \in \mathbb{C}$, AA_u randomly chooses $v_{u,c} \in \mathbb{Z}_N$ and let $V_{u,c} = g^{v_{u,c}}$. Finally, AA_u releases its public parameters $\mathsf{APK}_u = \{T_{\mathrm{att}} | att \in Q_u\}, \mathsf{AVK}_u = \{V_{u,c} | c \in \mathbb{C}\}$ and the master secret key $\mathsf{AMK}_u = (\{s_{\mathrm{att}} | att \in Q_u\}, \{v_{u,c} | c \in \mathbb{C}\})$.

CaKeyGen$(gid, \mathsf{GLPK}, \{\mathsf{AVK}_u | u \in \mathbb{U}\}, \mathsf{CMK}_c) \rightarrow (\mathsf{csk}_{\mathrm{gid},c}, \mathsf{cpk}_{\mathrm{gid},c})$. Whenever a user submits her gid to CA_c in order to get user-ca-key, CA_c selects $r_{\mathrm{gid},c} \in \mathbb{Z}_N$ at random and $P_{\mathrm{gid},c}, P'_{\mathrm{gid},c} \in G_{p_3}$, and sets

$$csk_{\mathrm{gid},c} = g^{\alpha_c} h^{r_{\mathrm{gid},c}} P_{\mathrm{gid},c}, \quad L_{\mathrm{gid},c} = g^{r_{\mathrm{gid},c}} P'_{\mathrm{gid},c}.$$

For $u = 1$ to U, CA_c randomly chooses $P_{\mathrm{gid},c,u} \in G_{p_3}$ and computes $\Upsilon_{\mathrm{gid},c,u} = V_{u,c}^{r_{\mathrm{gid},c}} P_{\mathrm{gid},c,u}$.

CA_c computes $\gamma_{\mathrm{gid},c} = \mathsf{Sign}(\mathsf{SignKey}_c, gid \| c \| L_{\mathrm{gid},c} \| \Upsilon_{\mathrm{gid},c,1} \| \cdots \| \Upsilon_{\mathrm{gid},c,U})$.

Then, let $cpk_{\mathrm{gid},c} = (gid, c, L_{\mathrm{gid},c}, \{\Upsilon_{\mathrm{gid},c,u} | u \in \mathbb{U}\}, \gamma_{\mathrm{gid},c})$.

AuthKeyGen$(att, \{\mathsf{cpk}_{\mathrm{gid},c} | c \in \mathbb{C}\}, \mathsf{GLPK}, \{\mathsf{CVK}_c | c \in \mathbb{C}\}, \mathsf{AMK}_u) \rightarrow \mathsf{attk}_{\mathrm{att,gid}}$ or $\square$. Whenever a user asks AA_u for a secret key for att, AA_u operates as follows.

(1) For $c = 1$ to C, AA_u splits $cpk_{\mathrm{gid},c}$ into $(gid, c, L_{\mathrm{gid},c}, \{\Upsilon_{\mathrm{gid},c,u} | u \in \mathbb{U}\}, \gamma_{\mathrm{gid},c})$:

$$valid \leftarrow Verify(VerifyKey_c, gid \| c \| L_{\mathrm{gid},c} \| \Upsilon_{\mathrm{gid},c,1} \| \cdots \| \Upsilon_{\mathrm{gid},c,U}, \gamma_{\mathrm{gid},c}), \tag{1}$$

$$e(g, \Upsilon_{\mathrm{gid},c,u}) = e(V_{u,c}, L_{\mathrm{gid},c}) \neq 1. \tag{2}$$

Alternatively, if the submitted $\{cpk_{\mathrm{gid},c} | c \in \mathbb{C}\}$ are invalid, AA_u outputs $\square$.

(2) For $c = 1$ to C, AA_u randomly chooses $P'_{\mathrm{att,gid},c} \in G_{p_3}$, and set $\mathsf{attk}_{\mathrm{att,gid},c} = (\Upsilon_{\mathrm{gid},c,u})^{s_{\mathrm{att}}/v_{u,c}} P'_{\mathrm{att,gid},c}$.

Note that

$$\begin{aligned}
\mathsf{attk}_{\mathrm{att,gid},c} &= (\Upsilon_{\mathrm{gid},c,u})^{s_{\mathrm{att}}/v_{u,c}} P'_{\mathrm{att,gid},c} \\
&= (V_{u,c}^{r_{\mathrm{gid},c}} P_{\mathrm{gid},c,u})^{s_{\mathrm{att}}/v_{u,c}} P'_{\mathrm{att,gid},c} \\
&= (g^{v_{u,c} \cdot r_{\mathrm{gid},c}} P_{\mathrm{gid},c,u})^{s_{\mathrm{att}}/v_{u,c}} P'_{\mathrm{att,gid},c} \\
&= T_{\mathrm{att}}^{r_{\mathrm{gid},c}} (P_{\mathrm{gid},c,u})^{s_{\mathrm{att}}/v_{u,c}} P'_{\mathrm{att,gid},c}.
\end{aligned}$$

As $(P_{\mathrm{gid},c,u})^{s_{\mathrm{att}}/v_{u,c}} P'_{\mathrm{att,gid},c}$ is in G_{p_3} and $P'_{\mathrm{att,gid},c}$ is randomly chosen, then $\mathsf{attk}_{\mathrm{att,gid},c} = T_{\mathrm{att}}^{r_{\mathrm{gid},c}} P_{\mathrm{att,gid},c}$.

(3) AA_u releases user-att-key $\mathsf{attk}_{\mathrm{att,gid}}$ to user

$$\begin{aligned}
\mathsf{attk}_{\mathrm{att,gid}} &= \prod_{c=1}^{C} \mathsf{attk}_{\mathrm{att,gid},c} = \prod_{c=1}^{C} T_{\mathrm{att}}^{r_{\mathrm{gid},c}} P_{\mathrm{att,gid},c} \\
&= T_{\mathrm{att}}^{\sum_{c=1}^{C} r_{\mathrm{gid},c}} \prod_{c=1}^{C} P_{\mathrm{att,gid},c} \\
&= T_{\mathrm{att}}^{\sum_{c=1}^{C} r_{\mathrm{gid},c}} P_{\mathrm{att,gid}}.
\end{aligned} \tag{3}$$

Encrypt$(m, \mathbb{A}, \mathsf{GLPK}, \{\mathsf{CPK}_c | c \in \mathbb{C}\}, \{\mathsf{APK}_u\}) \rightarrow \mathsf{CT}, \mathsf{En\text{-}para}(m)$. $\mathbb{A}$ is the access policy which can be symbolized by matrix (M, ρ). (M, ρ) can be represented as an $l \times n$ matrix which uses ρ to map each row M_x of M to an attribute $\rho(x)$. Besides, m is a piece of plaintext to be encrypted.

The algorithm picks a vector $\boldsymbol{v} = (s, v_2, \ldots, v_n) \in Z_N^n$ at random, for every single $x \in \{1, \ldots, l\}$, selects $r_x \in \mathbb{Z}_N$ randomly. Let $M_x \cdot \boldsymbol{v}$ denote the inner product of the xth row of M. Chiphertext is constructed as follows:

$$\mathbb{A} = (M, \rho), C = m \cdot \prod_{c=1}^{C} e(g, g)^{\alpha_c \cdot s}, C' = g^s,$$
$$\{C_x = h^{M_x \cdot \boldsymbol{v}} T_{\rho(x)}^{-r_x}, C_x' = g^{r_x} | x \in \{1, \ldots, l\}\}.$$

$\mathsf{En\text{-}para}(m)$ (the encryption information) is composed of all random r_x (i.e., $\mathsf{En\text{-}para}(m) = \{r_1, \ldots, r_n\}$) and the first entry s of vector $\boldsymbol{v} = (s, v_2, \ldots, v_n) \in Z_N^n$. For each message m, s is randomly chosen at the very beginning of the whole scheme and will be applied as the first entry of the new vector $\boldsymbol{v}'$ which will be discussed later. In other words, s will not change whether the policy associated with m changes or not.

Decrypt$(\mathsf{CT}, \mathsf{GLPK}, \{\mathsf{APK}_u\}, \mathsf{UDK}_{\mathrm{gid}}) \to m$ or $\square$. CT is split and constructed as $\langle C', C, \{C_x, C_x' | x \in \{1, \ldots, l\}\}, (M, \rho)\rangle$, the dec-key $\mathsf{UDK}_{\mathrm{gid}}$ is split into $(\{csk_{\mathrm{gid},c}, cpk_{\mathrm{gid},c} | c \in \mathbb{C}\}, \{\mathsf{attk}_{\mathrm{att,gid}} | att \in A_{\mathrm{gid}}\})$.

The algorithm calculates:

$$1.csk_{\mathrm{gid}} = \prod_{c=1}^{C} csk_{\mathrm{gid},c} = g^{\alpha} h^{r_{\mathrm{gid}}} P_{\mathrm{gid}},$$
$$2.L_{\mathrm{gid}} = \prod_{c=1}^{C} L_{\mathrm{gid},c} = g^{r_{\mathrm{gid}}} P'_{\mathrm{gid}},$$
$$\forall att \in A_{\mathrm{gid}}, \mathsf{attk}_{\mathrm{att,gid}} = T_{\mathrm{att}}^{\sum_{c=1}^{C} r_{\mathrm{gid},c}} P_{\mathrm{att,gid}} = T_{\mathrm{att}}^{r_{\mathrm{gid}}} P_{\mathrm{att,gid}}.$$

If A_{gid} matches (M, ρ), recall that $\sum_{\rho(x) \in A_{\mathrm{gid}}} \omega_x M_x = (1, 0, \ldots, 0)$, then it calculates

$$e(C', csk_{\mathrm{gid}}) \,/ \prod_{\rho(x) \in A_{\mathrm{gid}}} (e(C_x, L_{\mathrm{gid}}) \cdot e(C_x', \mathsf{attk}_{\rho(x),\mathrm{gid}}))^{\omega_x} = e(g, g)^{\alpha s}.$$

The ciphertext C can then be decrypted to get m.

DPUKeyGen$(\{\mathsf{CPK}_c \mid c \in \mathbb{C}\}, \{\mathsf{APK}_u\}, \mathsf{En\text{-}para}(m), \mathbb{A}, \mathbb{A}'\}) \to \mathsf{DPUK}_m$. The algorithm inputs the public parameters $\{\mathsf{CPK}_c \mid c \in \mathbb{C}\}, \{\mathsf{APK}_u\}$, the ciphertext information $\mathsf{En\text{-}para}(m)$ of m, both the current access policy (M, ρ) and the new access policy which we symbolized by (M', ρ'). Assume that M' is a new $l' \times n'$ access matrix using ρ' to map the rows to attributes. Based on the fact that function ρ and ρ' are non-injective, we can define $num_{\rho_{(x)},M}$ and $num_{\rho_{(x)},M'}$ to be the number of attribute $\rho_{(x)}$ in M and M', respectively.

We initialize the DPUKeyGen procedure by calling a comparing algorithm named as PolicyComp (shown in Algorithm 1). This algorithm compares the new policy with the current one. It outputs the row indexes information which can be classified into three sets: $R_{1,M'}, R_{2,M'}, R_{3,M'}$. Let both $R_{1,M'}$ and $R_{2,M'}$ be denoted as the set of indexes y, which indicate that $\rho'(y)$ exists in the current matrix M. If $num_{\rho'(y),M'} \leqslant num_{\rho'(y),M}$, let $(\rho(x) = \rho'(y))$, the indexes are put in $R_{1,M'}$. If $num_{\rho'(y),M'} > num_{\rho'(y),M}$, the excess part (i.e., $num_{\rho'(y),M'} - num_{\rho'(y),M}$) will be recorded in $R_{2,M'}$. $R_{3,M'}$ is the set log that $\rho'(y)$ do not exist in M (i.e., $\rho'(y)$ appears to be a new attribute).

Then the algorithm picks a new random vector $\boldsymbol{v}' \in Z_p^{n'}$ and sets s as the first entry. Afterwards, it computes the following:

(1) $\delta_x = M_x \cdot \boldsymbol{v}$. M_x is the valid share of the vector corresponding to the xth row in M.

(2) $\delta_y' = M_y' \cdot \boldsymbol{v}'$. M_y' is the valid share of the vector corresponding to the yth row in M.

(3) $R_M = \{1, \ldots, l\}$ are the rows' index set in M.

Afterwards, the update key component can be generated in the following manner, which is in fact classified into three types depending on (y, x).

For each $y \in [1, \ldots, l']$:

Algorithm 1 PolicyComp

Input: current policy (M, ρ) with $l \times n$ matrix; new policy (M', ρ') with $l' \times n'$ matrix;
Output: $R_{1,M'}, R_{2,M'}, R_{3,M'} \leftarrow$ three row index sets of M';

```
initialize the value of R_{1,M'}, R_{2,M'}, R_{3,M'} to ∅;
R_M ← row index set of M;
for y = 1 to l' do
    if ρ'(y) is in M then
        if ∃x ∈ R_M s.t. ρ(x) == ρ'(y) then
            add (y, x) into R_{1,M'};
            delete x from R_M;
        else
            find any x ∈ [1, l] s.t. ρ(x) == ρ'(y);
            add (y, x) into R_{2,M'};
        end if
    else
        add (y, 0) into R_{3,M'};
    end if
end for
```

[TYPE1] $(y, x) \in R_{1,M'}$, the update key elements will be

$$\mathsf{DPUK}_{y,x,m} = \left(\mathsf{DPUK}_{y,x,m} = h^{\delta'_y - \delta_x}\right).$$

and then set $r'_y = r_x$.

[TYPE2] $(y, x) \in R_{2,M'}$, the algorithm randomly picks $r'_y, d_y \in \mathbb{Z}_p$, and generates the update key elements

$$\mathsf{DPUK}_{y,x,m} = \left(d_y, \mathsf{DPUK}_{y,x,m} = h^{\delta'_y - d_y \delta_x}\right).$$

[TYPE3] $(y, x) \in R_{3,M'}$, the algorithm randomly picks $r'_y \in \mathbb{Z}_p$, and generates the update key elements

$$\mathsf{DPUK}_{y,x,m} = \left(\mathsf{DPUK}^{(1)}_{y,x,m} = h^{\delta'_y} T^{-r'_y}_{\rho'(y)}, \mathsf{DPUK}^{(2)}_{y,x,m} = g^{-r'_y}\right).$$

Therefore, the update key DPUK_m is constructed as

$$\begin{aligned}\mathsf{DPUK}_m = ((&\mathsf{TYPE1}, \{\mathsf{DPUK}_{y,x,m}\}_{(y,x)\in R_{1,M'}})\\ &(\mathsf{TYPE2}, \{\mathsf{DPUK}_{y,x,m}\}_{(y,x)\in R_{2,M'}})\\ &(\mathsf{TYPE3}, \{\mathsf{DPUK}_{y,x,m}\}_{(y,x)\in R_{3,M'}})).\end{aligned}$$

Afterward, the data owner submits the update key DPUK_m to the cloud server.

CTUpdate$(\mathsf{CT}, \mathsf{DPUK}_m) \rightarrow \mathsf{CT}'$. Once receiving the update key DPUK_m, for each $y \in [1, \ldots, l']$, the cloud server will run this algorithm to update the ciphertext elements. The original elements will be updated to E' according to the three types mentioned above.

[TYPE1] $y \in R_{1,M'}$, the new ciphertext elements E' will be

$$E' = (C' = g^s, C_y = C_x \cdot \mathsf{DPUK}_{y,x,m} = h^{M'_y \cdot \boldsymbol{v}'} T^{-r'_y}_{\rho'(y)}, C'_y = g^{r'_y}),$$

where $r'_y = r_x$.

[TYPE2] $y \in R_{2,M'}$, the new ciphertext elements E' will be

$$E' = (C' = g^s, C_y = (C_x)^{d_y} \cdot \mathsf{DPUK}_{y,x,m} = h^{M'_y \cdot \boldsymbol{v}'} T^{-r'_y}_{\rho'(y)}, C'_y = g^{r'_y}),$$

where $r'_y = d_y r_x$.

[TYPE3] $y \in R_{3,M'}$, the new ciphertext elements E' will be

$$E' = (C' = g^s, C_y = \mathsf{DPUK}^{(1)}_{y,x,m} = h^{\delta'_y} T^{-r'_y}_{\rho'(y)}, C'_y = \mathsf{DPUK}^{(2)}_{y,x,m} = g^{-r'_y}).$$

Table 1 Feature Comparison

	Multi-authority	KP/CP	Security model	Security	Ciphertext updating	Dynamic policy updating
Goyal [2]	×	KP	Standard	Selective	×	×
Yu [14]	√	KP	Standard	Selective	√	×
Lewko [15]	×	CP	Standard	Adaptive	×	×
Lewko [6]	√	CP	Random Oracle	Adaptive	×	×
Liu [7]	√	CP	Standard	Adaptive	√	×
Sahai [9]	×	KP+CP	Standard	Selective	√	×
Yang [5]	√	CP	Generic Group	Adaptive	√	√
Ours	√	CP	Standard	Adaptive	√	√

Table 2 Complexity

	MA-CP-ABE + straight-forward	Ours Type 1	Ours Type 2	Ours Type 3
Communication overhead	$2\|\Gamma\| + 2C' + 1$	$\|\Delta\|$	$\|L\| + \|\Delta\|$	$2\|\Delta\|$
Ciphertext updating	$\|D\| + 2C' + 2$	$C' + 2$	$2C' + 2$	$2C' + 2$

Then, the new ciphertext CT' is reconstructed as

$$CT' = \left(C = m \cdot \prod_{c=1}^{C} e(g,g)^{\alpha_c \cdot s}, E' | \forall y \in [1, \ldots, l'] \right).$$

During the entire update procedure, the cloud has neither transmitted the ciphertext back and forth nor decrypted it. Moreover, the data owner only needs to release a new access policy and submit it to the cloud server, which makes it much more safe and efficient (see Appendix B for the security proof).

Remark 2. Note that in the system above, an attribute could only appear at most once in an LSSS structure, which is a crucial restriction in security proof. That is also why we classify two types of indexes to denote the attributes that exist in current matrix M.

4 Comparison

In Table 1, we compare state-of-the-art work [2, 5–7, 9, 14, 15] and our DPU-CP-ABE system in a series of aspects, including Multi Authority, KP/CP, Security Model, etc.

The complexity comparison between the MA-CP-ABE [7] scheme and our DPU-CP-ABE is listed in Table 2. Since there is no policy updating module in the MA-CP-ABE system, we utilize the straight-forward implementation as demonstrated in Figure 1 to accomplish the updating task.

The symbols used in Table 2 are as follows: C' refers to the number of attributes involved in encryption under updated policy, which reflects how many rows have participate in the LSSS Matrix (M', ρ'). $|D|$ denotes the number of rows in (M, ρ) which will be used in decryption. $|\Gamma|$ is the length of the elements in G_T, $|\Delta|$ is the length of the elements in G, and $|L|$ is the length of $\mathbb{Z}_p$. We mainly focus on two aspects in the comparison: communication overhead[3)] and ciphertext updating. Firstly, in MA-CP-ABE, the data owner has to retrieve the entire ciphertext and send the re-encrypted data back to the cloud, along with the other ciphertext components. In contrast, in our scheme, data owner only needs to submit the components which need to be updated. Secondly, in MA-CP-ABE, the data owner have to do the decryption and re-encryption job all by himself in MA-CP-ABE. Whereas, in our scheme, the ciphertext updating task is outsourced to the cloud. However, for each data m, the data owner should provide extra secret information $\mathsf{En\text{-}para}(m)$ in order to update the current policy. The $\mathsf{En\text{-}para}(m)$ will increase linearly with the growth of the data, maintaining such additional information seems to be an efficiency

3) For ease of description, we discuss the communication situation in an ideal environment, since the data transfer rate can be influenced by a lot of factors in real world, which is beyond the scope of this paper.

drawback for the data owner. However, the policy update will result in the update of users' private key, correspondingly to other policy updating schemes such as [2, 9]. Meanwhile, ABE is a one-to-many encryption. Taking the scenario of nationwide joint plague-prevention operations in the introduction, for example, the medical institutions will join in continuously, which means that the number of users will scale enormously and that the policy will be updated frequently. If the users private keys must be updated with each policy update, the generation of a new private key for each user will certainly lead to an extremely heavy computation cost for the whole system. By applying our scheme, however, the users will not need to update their private keys whenever the data owner updates the policy. This is an obvious benefit for all the users in the system. The sacrifice of the data owner's extra storage will not only avoid the need to update the new policy by himself, but also save the computation cost of updating the keys of all users in the system. Therefore, our scheme is efficient on the whole.

5 Conclusion

In this paper, we propose a fully secure CP-ABE scheme specifically designed for dynamic updating issues in policy-changing circumstances under the standard model. State-of-the-art methods of policy-updatable ABE either exert restrictions on the policy to be updated or are designed for a weak security model. In comparison, our scheme, namely DPU-CP-ABE, solves both limitations simultaneously. Moreover, we show that our approach is more efficient compared to the straight-forward policy-updating implementation [7] and supports any type of fine-grained updating policy. More importantly, our scheme is much safer than the policy-updating method in [5]. Our DPU-CP-ABE scheme is proved to be adaptively secure under the standard model.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 61202179, 61173089, 61472298, 61472310, U1405255, 61502248), National High-Tech R&D Program (863) (Grant No. 2015AA016007), SRF for ROCS, SEM and Fundamental Research Funds for the Central Universities.

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Sahai A, Waters B. Fuzzy identity-based encryption. In: Proceedings of 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, 2005. 457–473
2 Goyal V, Pandey O, Sahai A, et al. Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of 13th ACM Conference on Computer and Communications Security, Alexandria, 2006. 89–98
3 Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. In: Proceedings of IEEE Symposium on Security and Privacy, Oakland, 2007. 321–334
4 Hur J, Noh D K. Attribute-based access control with efficient revocation in data outsourcing systems. IEEE Trans Parall Distrib Syst, 2011, 22: 1214–1221
5 Yang K, Jia X, Ren K, et al. Enabling efficient access control with dynamic policy updating for big data in the cloud. In: Proceedings of the IEEE International Conference on Infocom, Toronto, 2014. 2013–2021
6 Lewko A, Waters B. Decentralizing attribute-based encryption. In: Proceedings of 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, 2011. 568–588
7 Liu Z, Cao Z F, Huang Q, et al. Fully secure multi-authority ciphertext-policy attribute-based encryption without random oracles. In: Proceedings of 16th European Symposium on Research in Computer Security, Leuven, 2011. 278–297
8 Ruj S, Nayak A, Stojmenovic I. Dacc: distributed access control in clouds. In: Proceedings of the IEEE International Conference on Trustcom, Changsha, 2011. 91–98
9 Sahai A, Seyalioglu H, Waters B. Dynamic credentials and ciphertext delegation for attribute-based encryption. In: Proceedings of 32nd Annual Cryptology Conference, Santa Barbara, 2012. 199–217
10 Waters B. Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Proceedings of 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, 2011. 53–70
11 Chase M. Multi-authority attribute based encryption. In: Proceedings of 4th Theory of Cryptography Conference, Amsterdam, 2007. 515–534
12 Beimel A. Secure schemes for secret sharing and key distribution. Dissertation for the Doctoral Degree. Haifa: Technion-Israel Institute of Technology, Faculty of Computer Science, 1996

13 Goldwasser S, Micali S, Rivest R L. A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput, 1988, 17: 281–308
14 Yu S C, Wang C, Ren K, et al. Achieving secure, scalable, and fine-grained data access control in cloud computing. In: Proceedings of the IEEE International Conference on Infocom, San Diego, 2010. 1–9
15 Lewko A, Okamoto T, Sahai A, et al. Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Proceedings of 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, 2010. 62–91

Appendix A Number-theoretic assumptions

Composite order bilinear groups[4)] are a building block of our DPU-CP-ABE scheme. Let $\mathfrak{G}$ define the group generator, along with inputting security parameter λ while outputting $(p_1, p_2, p_3, G, G_T, e)$ as well. Here, p_1, p_2, p_3 are three different primes, $N = p_1p_2p_3$ are the orders of G and G_T, where $e : G \times G \to G_T$, then:

1. (Bilinear) $\forall g, h \in G$ and $x, y \in \mathbb{Z}_N, e(g^x, h^y) = e(g, h)^{xy}$;
2. (Non-degenerate) $\exists g \in G$ so that the order of $e(g, g)$ in G_T is N.

Suppose either the bilinear map e or the group operations in G, G_T related to λ are computable in terms of polynomial time. Let $G_{p_1}, G_{p_2}, G_{p_3}$ be G's three subgroups with p_1, p_2, p_3 as respective orders. Note that when $h_a \in G_{p_a}$ and $h_b \in G_{p_b}$ where $a \neq b$, $e(h_a, h_b) = 1$. 1 denotes the identity element in G_T.

Assumption A1 (Three primes attached to subgroup decision problem[5)])**.** Provided with a group generator $\mathfrak{G}$, the distributions are given as follows:

$$\mathcal{G} = (e, G, G_T, N = p_1p_2p_3) \xleftarrow{R} \mathfrak{G},$$
$$g \xleftarrow{R} G_{p_1}, I_3 \xleftarrow{R} G_{p_3},$$
$$V = (g, \mathcal{G}, I_3),$$
$$W_1 \xleftarrow{R} G_{p_1p_2}, W_2 \xleftarrow{R} G_{p_1}.$$

A poly-time algorithm $\mathcal{A}$'s advantage in terms of breaking Assumption A1 is defined as

$$Adv1_{\mathfrak{G},\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(V, W_1 = 1)] - \Pr[\mathcal{A}(V, W_2 = 1)]|.$$

For any poly-time algorithm $\mathcal{A}$, if $Adv1_{\mathfrak{G},\mathcal{A}}(\lambda)$ is a non-negligible function of λ, then $\mathfrak{G}$ fails to satisfy Assumption A1.

Assumption A2[5)]**.** Given the generator $\mathfrak{G}$, Distributions are given as follows:

$$\mathcal{G} = (e, G, G_T, N = p_1p_2p_3) \xleftarrow{R} \mathfrak{G},$$
$$g, I_1 \xleftarrow{R} G_{p_1}, I_2, J_2 \xleftarrow{R} G_{p_2}, I_3, J_3 \xleftarrow{R} G_{p_3},$$
$$V = (g, \mathcal{G}, I_1I_2, I_3, J_2J_3),$$
$$W_1 \xleftarrow{R} G, W_2 \xleftarrow{R} G_{p_1p_3}.$$

A poly-time algorithm $\mathcal{A}$'s advantage in terms of breaking Assumption A2 is defined as

$$Adv2_{\mathfrak{G},\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(V, W_1 = 1)] - \Pr[\mathcal{A}(V, W_2 = 1)]|.$$

For any polynomial time algorithm $\mathcal{A}$, if $Adv2_{\mathfrak{G},\mathcal{A}}(\lambda)$ is a non-negligible function of λ, then $\mathfrak{G}$ fails to satisfy Assumption A2.

Assumption A3[5)]**.** Given the generator $\mathfrak{G}$, Distributions are given as followed:

$$\mathcal{G} = (e, G, G_T, N = p_1p_2p_3) \xleftarrow{R} \mathfrak{G},$$
$$s, \alpha \xleftarrow{R} \mathbb{Z}_N, g \xleftarrow{R} G_{p_1}, I_2, J_2, K_2 \xleftarrow{R} G_{p_2}, I_3 \xleftarrow{R} G_{p_3},$$
$$V = (\mathcal{G}, g, g^\alpha I_2, I_3, g^s J_2, K_2),$$
$$W_1 = e(g, g)^{\alpha s}, W_2 \xleftarrow{R} G_T.$$

A poly-time algorithm $\mathcal{A}$'s advantage in terms of breaking Assumption A3 is defined as

$$Adv3_{\mathfrak{G},\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(V, W_1 = 1)] - \Pr[\mathcal{A}(V, W_2 = 1)]|.$$

For any polynomial time algorithm $\mathcal{A}$, if $Adv3_{\mathfrak{G},\mathcal{A}}(\lambda)$ is a non-negligible function of λ, then $\mathfrak{G}$ fails to satisfy Assumption A3.

4) Boneh D, Goh E J, Nissim K. Evaluating 2-DNF formulas on ciphertexts. In: Proceedings of 2nd Theory of Cryptography Conference, Cambridge, 2005. 325–341.

5) Lewko A, Waters B. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Proceedings of 7th Theory of Cryptography Conference, Zurich, 2010. 455–479.

Appendix B Proof of security

Our DPU-CP-ABE system is an extension of the Multi-authority CP-ABE scheme (MA-CP-ABE) with composite order bilinear groups in [7], which is proved to be adaptively secure on the basis of the standard model. The difference between our security game and [7] is that the adversary returns two messages (m_0, m_1) along with a tuple of policies $\{(M_1^*, \rho_1^*), \ldots, (M_k^*, \rho_k^*)\}$. Further, the adversary will receive ciphertexts of m_b encrypted under these policies one by one. In addition, the update key query is also allowed for the challenging messages (m_0, m_1) between (M_x^*, ρ_x^*) and (M_y^*, ρ_y^*) in our security game.

In our security model, $CA_{\bar{c}^*}$ is defined as the only uncorrupted CA and no $A_{\mathrm{gid}} \cup (\bigcup_{u_{\mathrm{cr}} \in \mathbb{U}_{\mathrm{cr}}} Q_{u_{\mathrm{cr}}})$ can satisfy the set of of challenge access structures, which implies that no polynomial time adversary is able to form a $\mathsf{usersk}_{\mathrm{gid},\bar{c}^*}$ by requesting keys in order to rebuild $e(g,g)^{\alpha_{\bar{c}^*} s}$. Note that in our security game, the challenger $\mathcal{B}$ will return all the key queries from the adversary, except $c = \bar{c}^*$. Similar to [15], we can also provide the answers to key queries related to $\bar{c}^*$.

First, we need some modifications before the proof. Let Ψ be the primary structure. We covert Ψ into Ψ' as follows:

(1) The $\mathsf{AuthkeyGen}$ algorithm generates $\mathsf{attk}_{\mathrm{att,gid}} = \{\mathsf{attk}_{\mathrm{att,gid},c} | c \in \mathbb{C}\}$, i.e., the decryption key of gid is constructed as

$$\begin{aligned}\mathsf{UDK}_{\mathrm{gid}} &= (\{csk_{\mathrm{gid},c}, cpk_{\mathrm{gid},c} | c \in \mathbb{C}\}, \{\mathsf{attk}_{\mathrm{att,gid}} | att \in A_{\mathrm{gid}}\}) \\ &= (\{csk_{\mathrm{gid},c}, cpk_{\mathrm{gid},c} | c \in \mathbb{C}\}, \{\{\mathsf{attk}_{\mathrm{att,gid},c} | c \in \mathbb{C}\} | att \in A_{\mathrm{gid}}\}) \\ &= (\{csk_{\mathrm{gid},c}, cpk_{\mathrm{gid},c} | c \in \mathbb{C}\}, \{\{\mathsf{attk}_{\mathrm{att,gid},c} | att \in A_{\mathrm{gid}}\} | c \in \mathbb{C}\}) \\ &= \{(csk_{\mathrm{gid},c}, cpk_{\mathrm{gid},c}, \{\mathsf{attk}_{\mathrm{att,gid},c} | att \in A_{\mathrm{gid}}\}) | c \in \mathbb{C}\} \\ &= \{\mathsf{usersk}_{\mathrm{gid},c} | c \in \mathbb{C}\},\end{aligned}$$

where $\mathsf{usersk}_{\mathrm{gid},c} = (csk_{\mathrm{gid},c}, cpk_{\mathrm{gid},c}, \{\{\mathsf{attk}_{\mathrm{att,gid},c} | att \in A_{\mathrm{gid}}\}\})$ is gid's user-key related to c.

(2) In the $\mathsf{Decrypt}$ algorithm, for $c = 1$ to $\mathbb{C}$, $e(g,g)^{\alpha_c s}$ is rebuilt by the using of $\mathsf{usersk}_{\mathrm{gid}}$:

$$e(C', csk_{\mathrm{gid}}) \,/ \prod_{\rho(x) \in A_{\mathrm{gid}}} (e(C_x, L_{\mathrm{gid}}) \cdot e(C'_x, \mathsf{attk}_{\rho(x),\mathrm{gid}}))^{\omega_x} = e(g,g)^{\alpha_c s}, \tag{B1}$$

then m can be regained by

$$m = C / \prod_{c=1}^{\mathbb{C}} e(g,g)^{\alpha_c s}. \tag{B2}$$

Notably, both the attacker and the user will obtain more information in Ψ'; thus, the security of Ψ' implicitly reflects that of Ψ.

Secondly, we define two extra structures: semi-functional keys (sf-key) and semi-functional ciphertexts (sf-ciphertext). We select $v_{\mathrm{att}} \in \mathbb{Z}_N$ randomly to be associated with the attributes.

Semi-functional Key (sf-key). Note that, given a gid, an (sf-key) $\mathsf{usersk}_{\mathrm{gid},\bar{c}^*}$ may be in two different forms. We randomly choose exponents $b, \pi, r_{\mathrm{gid},\bar{c}^*} \in \mathbb{Z}_N, \{f_{u,\bar{c}^*} \in \mathbb{Z}_N | u \in \mathbb{U}\}$, and components $P_{\mathrm{gid},\bar{c}^*}, P'_{\mathrm{gid},\bar{c}^*} \in G_{p_3}, \{P_{\mathrm{att,gid},\bar{c}^*} \in G_{p_3} | att \in A_{\mathrm{gid}}\}, \{P_{\mathrm{gid},\bar{c}^*,u} \in G_{p_3} | u \in \mathbb{U}\}$.

[TYPE1]: user-ca-key$(csk_{\mathrm{gid},\bar{c}^*}, cpk_{\mathrm{gid},\bar{c}^*})$ is constructed as

$$\begin{aligned}csk_{\mathrm{gid},\bar{c}^*} &= g^{\alpha_{\bar{c}^*}} h^{r_{\mathrm{gid},\bar{c}^*}} P_{\mathrm{gid},\bar{c}^*} g_2^{\pi}, \\ L_{\mathrm{gid},\bar{c}^*} &= g^{r_{\mathrm{gid},\bar{c}^*}} P'_{\mathrm{gid},\bar{c}^*}, \\ \Upsilon_{\mathrm{gid},\bar{c}^*,u} &= V_{u,\bar{c}^*}^{r_{\mathrm{gid},\bar{c}^*}} P_{\mathrm{gid},\bar{c}^*,u} (u = 1, \ldots, \mathbb{U}), \\ \gamma_{\mathrm{gid},\bar{c}^*} &= \mathsf{Sign}(\mathsf{SignKey}_{\bar{c}^*}, gid \| \bar{c}^* \| L_{\mathrm{gid},\bar{c}^*} \| \Upsilon_{\mathrm{gid},\bar{c}^*,1} \| \cdots \| \Upsilon_{\mathrm{gid},\bar{c}^*,U}), \\ cpk_{\mathrm{gid},\bar{c}^*} &= (gid, \bar{c}^*, L_{\mathrm{gid},\bar{c}^*}, \{\Upsilon_{\mathrm{gid},\bar{c}^*,u} | u \in \mathbb{U}\}, \gamma_{\mathrm{gid},\bar{c}^*}).\end{aligned}$$

$\forall att \in A_{\mathrm{gid}}$, the descendent $\mathsf{attk}_{\mathrm{att,gid},\bar{c}^*}$ is bulit as $\mathsf{attk}_{\mathrm{att,gid},\bar{c}^*} = T_{\mathrm{att}}^{r_{\mathrm{gid},\bar{c}^*}} P_{\mathrm{att,gid},\bar{c}^*}$.

[TYPE2]: user-ca-key$(csk_{\mathrm{gid},\bar{c}^*}, cpk_{\mathrm{gid},\bar{c}^*})$ is constructed as

$$\begin{aligned}csk_{\mathrm{gid},\bar{c}^*} &= g^{\alpha_{\bar{c}^*}} h^{r_{\mathrm{gid},\bar{c}^*}} P_{\mathrm{gid},\bar{c}^*} g_2^{\pi}, \\ L_{\mathrm{gid},\bar{c}^*} &= g^{r_{\mathrm{gid},\bar{c}^*}} P'_{\mathrm{gid},\bar{c}^*} g_2^{b}, \\ \Upsilon_{\mathrm{gid},\bar{c}^*,u} &= V_{u,\bar{c}^*}^{r_{\mathrm{gid},\bar{c}^*}} P_{\mathrm{gid},\bar{c}^*,u} g_2^{bf_{u,\bar{c}^*}} (u = 1, \ldots, \mathbb{U}), \\ \gamma_{\mathrm{gid},\bar{c}^*} &= \mathsf{Sign}(\mathsf{SignKey}_{\bar{c}^*}, gid \| \bar{c}^* \| L_{\mathrm{gid},\bar{c}^*} \| \Upsilon_{\mathrm{gid},\bar{c}^*,1} \| \cdots \| \Upsilon_{\mathrm{gid},\bar{c}^*,U}), \\ cpk_{\mathrm{gid},\bar{c}^*} &= (gid, \bar{c}^*, L_{\mathrm{gid},\bar{c}^*}, \{\Upsilon_{\mathrm{gid},\bar{c}^*,u} | u \in \mathbb{U}\}, \gamma_{\mathrm{gid},\bar{c}^*}).\end{aligned}$$

$\forall att \in A_{\mathrm{gid}}$, the descendent $\mathsf{attk}_{\mathrm{att,gid},\bar{c}^*}$ is built as $\mathsf{attk}_{\mathrm{att,gid},\bar{c}^*} = T_{\mathrm{att}}^{r_{\mathrm{gid},\bar{c}^*}} P_{\mathrm{att,gid},\bar{c}^*} g_2^{bv_{\mathrm{att}}}$.

It can be inferred that both Type 1 and Type 2 sf-keys can satisfy (1) and (2). Especially, when $b = 0$, Type 2 degenerates to Type 1.

Semi-functional Ciphertext (sf-ciphertext). Let g_2 be the generator of G_{p_2}, and d modulo N be a random exponent. Additionally, we choose another random vector $\boldsymbol{w} = (o, w_2, \ldots, w_n) \in Z_N^n$, as well as random numbers $\{\tau_x \in \mathbb{Z}_N | x \in \{1, 2, \ldots, l\}\}$. Then, sf-ciphertext will be:

$$C' = g^s g_2^d, \left\{C_x = h^{M_x \cdot \boldsymbol{v}} T_{\rho(x)}^{-r_x} g_2^{M_x \cdot \boldsymbol{w} + \tau_x v_{\rho(x)}}, C'_x = g^{r_x} g_2^{-\tau_x} | x \in \{1, 2, \ldots, l\}\right\}.$$

Considering when (a) a normal $\mathsf{usersk}_{\mathrm{gid},\bar{c}*}$ and a sf-ciphertext or (b) a normal ciphertext and a sf-key $\mathsf{usersk}_{\mathrm{gid},\bar{c}*}$, are applied in (B1), then $e(g,g)^{\alpha_{\bar{c}*}s}$ can be obtained and will be used in (B2). However, the use of both sf-key $\mathsf{usersk}_{\mathrm{gid},\bar{c}*}$ and sf-ciphertext in (B1) will produce $e(g,g)^{\alpha_{\bar{c}*}s} \cdot e(g_2,g_2)^{d\pi-bo}$, in which $e(g_2,g_2)^{d\pi-bo}$ will interfere with the computation in (B2). Nevertheless, if $d\pi = bo$, the sf-key will be able to decrypt the sf-ciphertext, thus, we treat this kind of keys as *nominally* semi-functional.

The security of Ψ' is established on the basis of Assumption A1, A2, and A3. For ease of expression, we utilize a hybrid statement upon a series of games to prove the security of our game. The first $\mathbf{Game_{real}}$ is the real one. In the last game $\mathbf{Game_{final}}$, all the $\{\mathsf{usersk}_{gic,\bar{c}*}\}$ are semi-functional of Type 1. It also includes a randomly distributed message encrypted under the Semi-functional Ciphertext algorithm, which is not the same as the challenge messages $\mathcal{A}$ provides.

$\mathbf{Game_{real}}$. The challenge ciphertext in this game is normal. Normal user-ca-key are used to reply to all the CA-KQs while all the AA-KQs are replied with user-att-key which are generated by running the normal AuthKeyGen in the real scheme.

$\mathbf{Game}_0$. The challenge ciphertext in this game is the sf-ciphertext. Normal user-ca-key are used to reply to all the CA-KQs while all the AA-KQs are replied with user-att-key which are generated by running the normal AuthKeyGen in the real scheme.

Let t be the number of CA-KQs made by the adversary $\mathcal{A}$. For $q = 1$ to t, we design the games as follows:

$\mathbf{Game}_{q,1}$. The challenge ciphertext in this game is the sf-ciphertext. Type 1 sf-key user-ca-keys are used to reply to the first $q-1$ CA-KQs; while Type 2 sf-key user-ca-key is used to reply to the qth CA-KQ. Normal user-ca-keys are used for replying the remaining CA-KQs. All the AA-KQs are replied with user-att-key which are generated by running the normal AuthKeyGen in the real scheme.

$\mathbf{Game}_{q,2}$. The challenge ciphertext in this game is the sf-ciphertext. Type 1 sf-key user-ca-keys are used to reply to the first qth CA-KQs. Normal user-ca-keys are used for replying the remaining CA-KQs. All the AA-KQs are replied with user-att-keys which are generated running the normal AuthKeyGen in the real scheme.

$\mathbf{Game_{final}}$. The challenge ciphertext in this game is a random message encrypted under **Semi-functional Ciphertext** algorithm, which is not the same as the challenge messages $\mathcal{A}$ provides. Type 1 sf-key user-ca-keys are used to reply all the CA-KQs. All the AA-KQs are replied with user-att-keys which are produced by executing the normal AuthKeyGen in the real scheme.

Notably, all AA-KQs in the above games are replied to with user-att-keys which are produced by executing the normal AuthKeyGen in the real scheme. It can be inferred that $\mathsf{attk}_{\mathrm{att},\mathrm{gid},\bar{c}*}$ is determined by user-ca-key $(csk_{\mathrm{gid},\bar{c}*}, cpk_{\mathrm{gid},\bar{c}*})$ correspondingly. Consequently, $\boldsymbol{user\text{-}key}_{\mathrm{gid},\bar{c}*}$ is also decided by $(csk_{\mathrm{gid},\bar{c}*}, cpk_{\mathrm{gid},\bar{c}*})$ correspondingly. Note that all user-ca-keys related to $\bar{c}*$ in $\mathbf{Game}_0$ are normal, while all user-ca-key related to $\bar{c}*$ in $\mathbf{Game}_{t,2}$ are sf-keys of Type 1. In other words, all user-key$_{\mathrm{gid},\bar{c}*}$s are normal in $\mathbf{Game}_0$ while all user-key$_{\mathrm{gid},\bar{c}*}$s in $\mathbf{Game}_{t,2}$ are the keys of Type 1 with semi-function. The indistinguishable quality of these games is demonstrated in the following lemmas. In the sequel, we will use $\mathbf{Game}_{0,2}$ to denote $\mathbf{Game}_0$, for simplicity.

Lemma B1. In the case of a UF-CMA signature algorithm Ω_{Sign}, presume that a poly-time algorithm $\mathcal{A}$ exists, provided that $Game_{\mathrm{real}}Adv\mathcal{A} - Game_0Adv\mathcal{A} = \epsilon$, and algorithm $\mathcal{B}$ with polynomial time can be constructed to break Assumption A1 with advantage ϵ.

Proof. $\mathcal{B}$ is given g, I_3, T. It will start the simulation of $\mathbf{Game}_{\mathrm{real}}$ or $\mathbf{Game}_0$ with $\mathcal{A}$. $\mathcal{B}$ runs the setup algorithms **GlobalSetup**, **CenterAuthSetup** and **AuthSetup** just the same as in the real scheme. Then, it randomly chooses exponents $a, \alpha_c \in \mathbb{Z}_N$, for each attribute $att \in Q_u$, as well as $s_{\mathrm{att}} \in \mathbb{Z}_N$. Afterwards, $\mathcal{B}$ sends the public parameters to $\mathcal{A}$, which contain $\mathsf{GLPK} = (N, g, g^a, I_3, \Omega_{\mathrm{Sign}})$, $\mathsf{CPK}_c = e(g,g)^{\alpha_c}, \mathsf{CVK}_c = VerifyKey_c$ and $\mathsf{APK}_u = \{T_{\mathrm{att}} | att \in Q_u\}, \mathsf{AVK}_u = \{V_{u,c} | c \in \mathbb{C}\}$.

Recall that $\Omega_{\mathrm{Sign}}, \mathsf{CVK}_c$ and AVK_u will be used in another signature subsystem to defend the collusion attack, which is independent from this proof. Hence, the public parameters that will be used in this proof is constructed as $PK = \{g, g^a, N, e(g,g)^{\alpha_c}, T_{\mathrm{att}} = g^{s_{\mathrm{att}}}\ \forall att\}$.

The adversary $\mathcal{A}$ specifies the only uncorrupted $CA_{\bar{c}^*}$ and $AA_{\bar{u}^*}$, then queries the normal keys from $\mathcal{B}$, since $\mathcal{B}$ knows $\mathsf{MSK} = \{\alpha_c, s_{\mathrm{att}}\}$.

$\mathcal{A}$ submits two equal length messages m_0, m_1 to $\mathcal{B}$, as well as a set of challenge access matrices $\{(M_1^*, \rho_1^*), \ldots, (M_k^*, \rho_k^*)\}$. In order to generate the challenge ciphertexts, $\mathcal{B}$ will let g^s implicitly be the G_{p_1} section of T, which means T is produced from $g^s \in G_{p_1}$ and probably has G_{p_2} as an element. $\mathcal{B}$ randomly selects $b \in \{0,1\}$ and sets $C = m_b e(g^{\alpha_c}, T),\ C' = T$.

Constructing C_x for each row x of M_x^*, $\mathcal{B}$ will first choose random values $v_2', \ldots, v_n' \in \mathbb{Z}_N$ and then create vector $\boldsymbol{v}' = (1, v_2', \ldots, v_n')$. It will likewise randomly select a $r_x' \in \mathbb{Z}_N$ and set $C_x = T^{aM_x^* \cdot \boldsymbol{v}'} T^{-r_{x'} s_{\rho(x)}},\ C_x' = T^{r_{x'}}$.

This means that $\boldsymbol{v} = (s, sv_2', \ldots, sv_n')$ and $r_x = r_x' s$. By modulo p_1, we find that $\boldsymbol{v}$ is a random vector with s as the first coordinate; r_x is also random. Thus, if $T \in G_{p_1}$, these are properly dispensed normal ciphertexts.

If $T \in G_{p_1 p_2}$, we refer g_2^d to the G_{p_2} section in T (i.e., $T = g^s g_2^d$). Hence, we get access to sf-ciphertext with $\boldsymbol{\omega} = da\boldsymbol{v}', \tau_x = -dr_x'$ and $v_{\rho_x} = s_{\rho(x)}$. Although the values of G_{p_1} parts are reused here, this would not cause unexpected correlations. By Chinese Remainder Theorem[6)], the results of $v_2', \ldots, v_n', r_x', s_{\rho_x}$ modulo p_1 are irrelevant from modulo p_2, so these are properly distributed sf-ciphertexts. $\mathcal{A}$ may query the DPUKeyGen oracle to gain more advantage. Let us consider the following two update key queries $\mathsf{DPUK}(m_0, (M_x^*, \rho_x^*), (M_y^*, \rho_y^*))$ and $\mathsf{DPUK}(m_1, (M_x^*, \rho_x^*), (M_y^*, \rho_y^*))$. We assume that the random numbers utilized in encrypting m_0 and m_1 are the same (since the simulator only chooses one

6) Ding C S, Pei D Y, Salomaa A. Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography. River Edge: World Scientific Publishing Co., Inc., 1996. 1–213.

challenge message by flipping a coin in the security game). The DPUKeyGen oracle returns the same update keys, which do not contain the challenge data. That means no information of the chosen challenging message is revealed to the adversary. Therefore, $\mathcal{B}$ is enabled to break Assumption A1 because of the advantage of ϵ using $\mathcal{A}$'s outputs.

Lemma B2. In the case of a UF-CMA signature algorithm Ω_{Sign}, presume that a poly-time algorithm $\mathcal{A}$ exists, provided that $Game_{q-1,2}Adv\mathcal{A} - Game_{q,1}Adv\mathcal{A} = \epsilon$, and algorithm $\mathcal{B}$ with polynomial time can be constructed to break Assumption A2 with advantage ϵ.

Proof. $\mathcal{B}$ is given $g, I_1I_2, I_3, J_2J_3, T$. It will start the simulation of $\mathbf{Game}_{q-1,2}$ or $\mathbf{Game}_{q,1}$ with $\mathcal{A}$. $\mathcal{B}$ randomly chooses exponents $a, \alpha_c \in \mathbb{Z}_N$, and for every single $att \in Q_u$ select $s_{\text{att}} \in \mathbb{Z}_N$ at random. Then $\mathcal{B}$ sends the public parameters to $\mathcal{A}$. The public parameters that will be used in this proof is constructed as $PK = \{g, g^a, N, e(g,g)^{\alpha_c}, T_{\text{att}} = g^{s_{\text{att}}}\ \forall att\}$.

$\mathcal{A}$ specifies the only uncorrupted $CA_{\bar{c}^*}$ and $AA_{\bar{u}^*}$, then queries the normal keys from $\mathcal{B}$, who knows $\mathsf{MSK} = \{\alpha_c, s_{\text{att}}\}$.

In order to generate the first $q-1$ sf-keys of Type 1, $\mathcal{B}$ selects $r_{\text{gid},\bar{c}*} \in \mathbb{Z}_N$ randomly and pick $P_{\text{gid},\bar{c}*}, P'_{\text{gid},\bar{c}*}, P_{\text{att,gid},\bar{c}*}$ from G_{p_3} to respond to each key request. Then it sets:

$$\begin{aligned}
&csk_{\text{gid},\bar{c}^*} = g^{\alpha_{\bar{c}^*}} g^{a r_{\text{gid},\bar{c}^*}} (J_2J_3)^{r_{\text{gid},\bar{c}^*}},\\
&L_{\text{gid},\bar{c}^*} = g^{r_{\text{gid},\bar{c}^*}} P'_{\text{gid},\bar{c}*},\\
&\mathsf{attk}_{\text{att,gid},\bar{c}^*} = T_{\text{att}}^{r_{\text{gid},\bar{c}^*}} P_{\text{att,gid},\bar{c}^*}\ \forall att \in A_{\text{gid}}.
\end{aligned}$$

It can be inferred that $csk_{\text{gid},\bar{c}^*}$ is dispensed properly, because the results of $r_{\text{gid},\bar{c}^*}$ modulo p_2 and p_3 are unrelated to that modulo modulo p_1. For key requests $> q$, as $\mathcal{B}$ knows MSK, it simply utilizes the key generation algorithm in order to make normal keys. $\mathcal{B}$ will then implicity set $g^{r_{\text{gid},\bar{c}^*}}$ to be the G_{p1} part of T, and set

$$\begin{aligned}
&csk_{\text{gid},\bar{c}^*} = g^{\alpha_{\bar{c}^*}} T^a P_{\text{gid},\bar{c}*},\\
&L_{\text{gid},\bar{c}^*} = T P'_{\text{gid},\bar{c}*},\\
&\mathsf{attk}_{\text{att,gid},\bar{c}^*} = T^{s_{\text{att}}} P_{\text{att,gid},\bar{c}^*}\ \forall att \in A_{\text{gid}}.
\end{aligned}$$

Note that when $T \in G_{p_1p_3}$, this appears to be a normal reasonably dispensed key. When $T \in G$, it becomes a Type 2 sf-key. It means that $v_{\text{att}} = s_{\text{att}}$. Let g_2^{π} be the G_{p2} section of T, we can get $\pi = a$ modulo p_2. (i.e., the G_{p2} part of csk and $L_{\text{gid},\bar{c}^*}$ are g_2^a, the G_{p2} part of $\mathsf{attk}_{\text{att,gid},\bar{c}^*}$ is $g_2^{a v_{\text{att}}}$). The value of v_{att} modulo p_2 is unrelated to s_{att} modulo p_1.

$\mathcal{A}$ submits two equal length messages m_0, m_1 to $\mathcal{B}$ as well as a set of challenge access matrices $\{(M_1^*, \rho_1^*), \ldots, (M_k^*, \rho_k^*)\}$. To generate challenge sf-ciphertexts, $\mathcal{B}$ will implicitly let $g^s = I_1$ and $g_2^d = I_2$. It randomly chooses $\omega_2, \ldots, \omega_n \in \mathbb{Z}_N$ and defines $\boldsymbol{\omega'} = (a, \omega_2, \ldots, \omega_n)$. It will also randomly choose a $r'_x \in \mathbb{Z}_N$ and sets the ciphertexts as follows:

$$\begin{aligned}
&C = m_b e(g^{\alpha_c}, I_1I_2),\ C' = I_1I_2,\\
&C_x = (I_1I_2)^{M_x^* \cdot \boldsymbol{\omega'}} (I_1I_2)^{-r_{x'} s_{\rho(x)}},\ C'_x = (I_1I_2)^{r_{x'}}.
\end{aligned}$$

Note that this will implicity set $\boldsymbol{v} = sa^{-1}\boldsymbol{\omega'}$, $\boldsymbol{\omega} = d\boldsymbol{\omega'}$, $r_x = r'_x s$ and $\tau_x = -dr'_x$. It can be inferred that s is shared in G_{p_1} and da in G_{p_2}. If $v_{\rho_x} = s_{\rho(x)}$ are of Type 1 semi-functional, they match those in kth key as required.

Consequently, suppose $T \in G_{p_1p_3}$, and $Game_{q-1,2}$ has simulated properly by $\mathcal{B}$. If $T \in \mathfrak{G}$ and all τ_x modulo p_2 are non-zero, $\mathcal{B}$ has simulated $Game_{q,1}$ properly. Therefore, $\mathcal{B}$ is able to break Assumption A2 with a non-negligible advantage ϵ by using $\mathcal{A}$'s outputs.

Lemma B3. In the case of a UF-CMA signature algorithm Ω_{Sign}, presume that a poly-time algorithm $\mathcal{A}$ exists, provided that $Game_{q,1}Adv\mathcal{A} - Game_{q,2}Adv\mathcal{A} = \epsilon$, and algorithm $\mathcal{B}$ with polynomial time can be constructed to break Assumption A2 with advantage ϵ.

Proof. $g, I_1I_2, I_3, J_2J_3, T$ are given to $\mathcal{B}$. It will start the simulation of $\mathbf{Game}_{q,1}$ or $\mathbf{Game}_{q,2}$ with $\mathcal{A}$. $\mathcal{B}$ selects exponents $a, \alpha_c \in \mathbb{Z}_N$ at random, and for every single $att \in Q_u$ chooses $s_{\text{att}} \in \mathbb{Z}_N$ randomly. Then, $\mathcal{B}$ sends the public parameters to $\mathcal{A}$. The public parameters that will be used in this proof are constructed as $PK = \{g, g^a, N, e(g,g)^{\alpha_c}, T_{\text{att}} = g^{s_{\text{att}}} \forall att\}$.

We apply the same method in constructing $q-1$ Type 1 sf-keys, $> q$ normal keys, as well as the challenge ciphertexts, as in the proof of the previous Lemma. This indicates that ciphertext is sharing the value ad in G_{p2}. It has no relationship with key q, so the value modulo $p2$ is random.

In order to make key q, $\mathcal{B}$ randomly chooses an exponent $k \in \mathbb{Z}_N$ and repeats the procedure it did before, then sets

$$\begin{aligned}
&csk_{\text{gid},\bar{c}^*} = g^{\alpha_{\bar{c}^*}} T^a P_{\text{gid},\bar{c}*} (J_2J_3)^k,\\
&L_{\text{gid},\bar{c}^*} = T P'_{\text{gid},\bar{c}*},\\
&\mathsf{attk}_{\text{att,gid},\bar{c}^*} = T^{s_{\text{att}}} P_{\text{att,gid},\bar{c}^*}\ \forall att \in A_{\text{gid}}.
\end{aligned}$$

The only difference lies in $csk_{\text{gid},\bar{c}^*}$, where another factor $(J_2J_3)^k$ is attached. This will randomize the G_{p_2} part of $csk_{\text{gid},\bar{c}^*}$ which makes the key no longer nominally semi-functional. Any attempt to decrypt the sf-ciphertext with $csk_{\text{gid},\bar{c}^*}$ will end in failure.

If $T \in G_{p_1p_3}$, this is a properly distributed Type 1 sf-key. If $T \in G$, this is a properly distributed Type 2 sf-key. Therefore, $\mathcal{B}$ can gain a non-negligible advantage of ϵ by using $\mathcal{A}$'s output to break Assumption A2.

Lemma B4. In the case of a UF-CMA signature algorithm Ω_{Sign}, presume that a poly-time algorithm $\mathcal{A}$ exists, provided that $Game_{t,2}Adv\mathcal{A} - Game_{\text{final}}Adv\mathcal{A} = \epsilon$, and algorithm $\mathcal{B}$ with polynomial time can be constructed to break Assumption A3 with advantage $\frac{\epsilon}{2}$.

Proof. $g, g^{\alpha_c} I_2, I_3, g^s J_2, K_2, T$ are given to $\mathcal{B}$. It will start the simulation of $\mathbf{Game}_{t,2}$ or $\mathbf{Game}_{\text{final}}$ with $\mathcal{A}$. $\mathcal{B}$ selects an exponent $a \in \mathbb{Z}_N$ at random and, for every single attribute $att \in Q_u$, randomly chooses $s_{\text{att}} \in \mathbb{Z}_N$. It obtains α_c from $g^{\alpha_c} I_2$. Then $\mathcal{B}$ sends the public parameters to $\mathcal{A}$. The public parameters that will be used in this proof are constructed as $PK = \{g, g^a, N, e(g,g)^{\alpha_c} = e(g, g^{\alpha_c} I_2), T_{\text{att}} = g^{s_{\text{att}}} \forall att\}$.

$\mathcal{A}$ specifies the only uncorrupted $CA_{\bar{c}^*}$ and $AA_{\bar{u}^*}$, then queries the normal keys from $\mathcal{B}$, who knows $\mathsf{MSK} = \{\alpha_c, s_{\text{att}}\}$.

In order to generate the Type 1 sf-keys, $\mathcal{B}$ randomly chooses $r_{\text{gid},\bar{c}*} \in \mathbb{Z}_N$ as well as elements $P_{\text{gid},\bar{c}*}, P'_{\text{gid},\bar{c}*}, P_{\text{att,gid},\bar{c}*}$ from G_{p3} to respond to each key request. Then, it sets

$$
\begin{aligned}
&csk_{\text{gid},\bar{c}^*} = g^{\alpha_{\bar{c}^*}} g^{a r_{\text{gid},\bar{c}^*}} K_2^{r_{\text{gid},\bar{c}^*}} P_{\text{gid},\bar{c}*},\\
&L_{\text{gid},\bar{c}^*} = g^{r_{\text{gid},\bar{c}^*}} P'_{\text{gid},\bar{c}*},\\
&\mathsf{attk}_{\text{att,gid},\bar{c}^*} = T_{\text{att}}^{r_{\text{gid},\bar{c}^*}} P_{\text{att,gid},\bar{c}^*}\ \forall att \in A_{\text{gid}}.
\end{aligned}
$$

$\mathcal{A}$ submits two equal length messages m_0, m_1 to $\mathcal{B}$ along with a set of challenge access matrices $\{(M_1^*, \rho_1^*), \ldots, (M_k^*, \rho_k^*)\}$. With the purpose of generating the challenge sf-ciphertexts, $\mathcal{B}$ obtains s from $g^s J_2$, randomly chooses $\omega_2, \ldots, \omega_n \in \mathbb{Z}_N$ and defines $\boldsymbol{\omega}' = (a, \omega_2, \ldots, \omega_n)$. It will also randomly chooses $r'_x \in \mathbb{Z}_N$ and set the ciphertexts as follows:

$$
\begin{aligned}
&C = m_b \cdot \prod_{c=1}^{C} T,\ C' = g^s J_2,\\
&C_x = (g^s J_2)^{M_x^* \cdot \boldsymbol{\omega}'} (g^s J_2)^{-r_{x'} s_{\rho(x)}},\ C'_x = (g^s J_2)^{r_{x'}}.
\end{aligned}
$$

Note that this will implicity set $\boldsymbol{v} = sa^{-1}\boldsymbol{\omega}'$, $\boldsymbol{\omega} = d\boldsymbol{\omega}'$, $r_x = r'_x s$ and $\tau_x = -dr'_x$. It can be inferred that s is shared in G_{p1} and da in G_{p2}.

When $T = e(g,g)^{\alpha_c s}$, this becomes a properly distributed semi-functional encryption of m_b. Elsewise, it is a random message in group G_T of properly distributed semi-functional encryption. As C is the number of CAs, $\mathcal{B}$ can break Assumption A3 with the advantage of $\frac{\epsilon}{C}$ using $\mathcal{A}$'s outputs.

Theorem B1. If Ω_{Sign} is proved to be UF-CMA secure, Assumptions A1–A3 work, DPU-CP-ABE scheme is therefore secure.

Proof. If the scheme of signature Ω_{Sign} is proved to be UF-CMA secure and Assumptions A1–A3 work, then we have demonstrated that the real security game and $Game_{\text{final}}$ are indistinguishable through the previous lemmas, in which value b is information-theoretically hidden to the attacker. Therefore, it can be concluded that the adversary fails to obtain a non-negligible advantage in terms of breaking Ψ', which means that the adversary is unable to break Ψ in our DPU-CP-ABE system with a non-negligible advantage.