Attribute-based Proxy Re-Encryption with Constant Size Ciphertexts

Arinjita Paul, S. Sharmila Deva Selvi and C. Pandu Rangan

Theoretical Computer Science Lab,
Department of Computer Science and Engineering,
Indian Institute of Technology Madras, Chennai, India.
{arinjita,sharmila,prangan}@cse.iitm.ac.in

Abstract. Attribute-based proxy re-encryption (ABPRE) allows a semi-trusted proxy to transform an encryption under an access-policy into an encryption under a new access policy, without revealing any information about the underlying message. Such a primitive facilitates fine-grained secure sharing of encrypted data in the cloud. In its key-policy flavor, the re-encryption key is associated with an access structure that specifies which type of ciphertexts can be re-encrypted. This paper proposes the first CCA secure key-policy attribute-based proxy re-encryption (KP-ABPRE) scheme allowing monotonic access structures with constant ciphertext size for both the original and re-encrypted ciphertexts. Prior to our work, only two attempts were made towards the construction of an RCCA secure and a CCA secure KP-ABPRE scheme in the literature. We show that both the systems are vulnerable to replayable chosen-ciphertext and chosen-ciphertext attack respectively.

When a user shares his data by delegating decryption towards an access-policy, the proxy can collude with a malicious delegatee to attempt to obtain the private keys of the delegator during the delegation period. If the private keys are exposed, the security of the delegator's data is completely compromised. The proxy or the delegatee can obtain all confidential data of the delegator at will at any time, even after the delegation period is over. Hence, achieving *collusion resistance* is indispensable to real-world applications. In this paper, we show that our construction satisfies collusion resistance. Our scheme is proven CCA secure in the random oracle model, based on Bilinear Diffie-Hellman exponent assumptions.

Keywords: proxy re-encryption, key-policy, attribute-based proxy re-encryption, unidirectional, bilinear map.

1 Introduction

In traditional proxy re-encryption systems [1,5], the communication model is one-to-one, in the sense that a message can be re-encrypted only towards a particular public key. In practice, however many scenarios require the re-encryption functionality without exact knowledge of the set of intended recipients. One such important application is data sharing in untrusted cloud storage. In a cloud storage system, a data owner may wish to share his encrypted data with users satisfying a specified access policy. To enable such expressive and fine-grained data sharing, Liang *et al*. [14] introduced the notion of attribute-based proxy re-encryption (ABPRE). ABPRE is an extension of the traditional proxy re-encryption (PRE) primitive to its attribute-based counterpart. The notion of PRE was introduced by Blaze, Bleumer and Strauss [5] to provide delegation of decryption rights. ABPRE designates a semi-trusted proxy to transform ciphertexts of users (delegators) satisfying an access policy into ciphertexts of users (delegatee) satisfying a new access policy. The proxy performs the conversion as a service upon receiving a special key construct called *re-encryption key* from the delegator, and should not learn anything about the underlying message.

ABPRE integrates the notion of proxy re-encryption (PRE) with attribute-based encryption (ABE) to effectively enhance the flexibility of delegation of decryption capability. ABE is a generalization of Identity-Based Encryption (IBE), introduced by Sahai and Waters [19] wherein a user's identity is generalized to a set of descriptive attributes instead of a single string. ABE has two variants: *Key-Policy ABE* (KP-ABE) and *Ciphertext-Policy ABE* (CP-ABE). In KP-ABE systems, the private key captures an access structure that specifies which type of ciphertexts the key can decrypt, while the ciphertexts are labeled with attribute sets. Its dual, CP-ABE associates a user's private key with a set of attributes, while ciphertexts are embedded with the access policy information. In light of the above, ABPRE can be classified into key-policy ABPRE (KP-ABPRE) and ciphertext-policy ABPRE (CP-ABPRE). Based on the direction of delegation, ABPRE systems are classified into unidirectional and bidirectional schemes. Additionally, based on the number of re-encryptions permitted, ABPRE systems are classified into single-hop and multi-hop schemes. In this work, we focus on unidirectional single-hop key-policy ABPRE schemes.

A cloud storage system may typically contain two kinds of encrypted data. The first kind termed *first-level ciphertext*, is the information that a user A encrypts under his attribute set W_1 and is likely to share with users identified by an attribute set W_2. Such an information is subject to re-encryption by the cloud, which performs the conversion upon getting the re-encryption key from user A. The second kind termed as *second-level ciphertext* is the re-encrypted data, converted by the cloud towards an access policy fulfilled by attribute set W_1 of user A, delegated by another user C. Such an encrypted file can not be further re-encrypted in a single hop scenario. The only way to illegally decrypt such ciphertexts is when a malicious user B with attributes W_2 and a cloud possessing a re-encryption key colludes to obtain the private key corresponding to an access structure satisfied by W_1. Again, the re-encryption rights are enabled for a bounded, fixed period and malicious parties may want to decrypt ciphertexts of A even beyond that period. *Collusion attack* [1] refers to such an act where a colluding delegatee and cloud extracts the private key of the delegator, causing harm to the delegator in every possible manner, such as unauthorised access to his sensitive data, identity theft and illegal delegation of decryption power. Thus, achieving collusion resistance is one of the major important problems in KP-ABPRE schemes. Collusion-resistant KP-ABPRE has many real-world applications such as blockchain based distributed cloud data storage and sharing, online medical service systems, online payment systems among others [13].

Note that achieving this powerful functionality of fine-grained access control comes at a cost. In a typical implementation of any previous construction of KP-ABPRE in the literature, the size of the ciphertext grows linearly with the size of the attribute set embedded by the sender. Also, the re-encryption and decryption time is proportional to the number of attributes involved of the receiver. Reducing the ciphertext size and computation cost is highly beneficial in scenarios with low-bandwidth requirements and limited computing power. In this paper, we study KP-ABPRE in light of achieving collusion resistance and constant size ciphertexts. To this end, we propose the first CCA secure and efficient KP-ABPRE scheme in the random oracle model, whose security is based on the Decisional Bilinear Diffie Hellman Exponent (DBDHE) assumption.

Related Work and Contribution

In this work, we address the problem of designing collusion-resistant non-interactive attribute based PRE in the key-policy setting supporting rich access policies such as monotonic access structures, proposed in [12]. To integrate PRE to ABE setting, Liang *et al*. [14] first defined the notion of attribute-based PRE and proposed a CP-ABPRE scheme in the standard model, proven CPA-secure under Augment Decisional Bilinear Diffie-Hellman assumption. Chung *et al*. [7] gives a detailed study of the existing attribute based PRE schemes in their work.

Li *et al*. [12] proposed the first KP-ABPRE scheme wherein matrix access structure is used to provide key-policy. Their construction is unidirectional, multi-use, collusion-resistant and is proven CPA-secure based on the Bilinear Decisional Diffie Hellman (DBDH) assumption. Note that, their construction relies on a trusted authority for the generation of the re-encryption keys, making the scheme highly infeasible. Since the same trusted authority is responsible for the generation of private keys, achieving delegation with the involvement of the trusted authority is trivial but impractical. Their work is extended in [11] to achieve an RCCA-secure KP-ABPRE scheme with matrix access structure to realize key-policy. Their design is unidirectional, multi-use and is claimed to be adaptively RCCA-secure based on DBDH assumption, with the same drawback of entrusting the trusted authority with the generation of re-encryption keys. Note that RCCA (replayable chosen ciphertext attack) is a weaker variant of CCA tolerating a "harmless mauling" of the challenge ciphertext. In 2015, Ge *et al*. [10] designed a unidirectional single-hop CCA-secure KP-ABPRE scheme supporting monotonic access structures, without random oracles under the 3-weak decisional bilinear DiffieHellman inversion(3-wDBDHI) assumption. However, note that, their construction does not adhere to the standard definition of KP-ABPRE. In essence, their scheme is a variation of conditional proxy re-encryption [20]. In their design, a first-level ciphertext C is labelled with a set of descriptive conditions W, encrypted towards an individual public key pk_i, decryptable only using its corresponding private key sk_i. Here, every re-encryption key is generated using an access structure tree $\mathcal{T}$ associated with conditional keywords. That is, C can only be re-encrypted towards another public key pk_j specified in the re-encryption key $RK_{i,\mathcal{T},j}$ only if W (used to label C) satisfies the access tree $\mathcal{T}$ of the re-encryption key. Their system enables one-to-one communication subject to conditions specified via access trees in re-encryption key, rather than a many-to-many transformation enabled by KP-ABPRE. Recently, Ge *et al*. [9] proposed an adaptive CCA-secure collusion resistant KP-ABPRE scheme that supports any monotonic access structures on users' keys. Their scheme is claimed to be secure in the standard model based on subgroup decision problem for 3 primes and composite order bilinear pairings.

Our contribution in this work is twofold. Firstly, we demonstrate two attacks on the security of the existing KP-ABPRE schemes in the literature. We show that the recent CCA secure KP-ABPRE construction of Ge *et al*. [9] is vulnerable to CCA attack. We also demonstrate an RCCA attack on the KP-ABPRE scheme due to Li *et al*. [11]. Consequently, only one result due to Li *et al*. [12] achieves attribute based proxy re-encryption in the key-policy setting, which is CPA secure in the random oracle model. In [8], Cohen *et al* remarks on the inadequacy of CPA security in proxy re-encryption. Besides, the difficulty in achieving a CCA-secure KP-ABPRE scheme has been discussed in [11]. Our second contribution lies in designing the first construction of a CCA secure KP-ABPRE scheme, proven secure under the Decisional Bilinear Diffie Hellman Exponent assumption in the random oracle model. All the previous attempts to construct KP-ABPRE schemes admitting expressive re-encryption and decryption policies produce ciphertexts whose size grows linearly with the number of attributes embedded in the ciphertext for both levels of encryption. This paper proposes the first KP-ABPRE result allowing monotonic access structure with constant size ciphertext, based on the KP-ABE framework of Rao *et al*. [18] and BLS short signature [6]. This is achieved by increasing the private key size by a factor of $|W|$, where W is the set of distinct attributes associated with the access structure embedded in the private key. Also, the scheme enjoys the feature of constant number of exponentiations during encryption, and constant number of bilinear pairings during encryption, re-encryption and decryption. This is especially useful in applications that have low bandwidth requirements and limited computing power.

2 Preliminaries

2.1 Bilinear Maps and Hardness Assumptions

Definition 1. ***(Bilinear Maps)*** *Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two finite cyclic groups of prime order p. A function $\hat{e} : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ is called a bilinear map if it satisfies the following three properties:*

- *Bilinearity: For all $a,\ b \in \mathbb{Z}_p^*$ and $g,\ h \in \mathbb{G}_0$, $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$.*
- *Non-degeneracy: if $\mathbb{G}_0 = \langle g \rangle$, then $\mathbb{G}_1 = \langle \hat{e}(g, g) \rangle$.*
- *Efficiency: $\hat{e}$ is efficiently computable.*

Definition 2. ***(n-Decisional Bilinear Diffie-Hellman Exponentiation (n-DBDHE) assumption [18])*** *The n-decisional bilinear diffie-hellman exponentiation (n-DBDHE) assumption is, given the elements $\{g, g^b, g^a, g^{a^2}, \cdots, g^{a^n}, g^{a^{n+2}}, \cdots, g^{a^{2n}}\} \in \mathbb{G}_0$ and $T \in \mathbb{G}_1$, there exists no PPT adversary which can decide whether $T = \hat{e}(g, g)^{n(a^{n+1})}$ or a random element from $\mathbb{G}_1$ with a non-negligible advantage, where g is a generator of $\mathbb{G}_0$ and $a, b \in_R \mathbb{Z}_p^*$.*

2.2 Access Structure and Linear Secret Sharing Schemes

Definition 3. ***(Access Structure [3])*** *Let $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\mathcal{P}}$ is monotone if $\forall B, C$, if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\mathcal{P}$, i.e., $\mathbb{A} \subseteq 2^{\mathcal{P}} \backslash \{\phi\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.*

In the context of ABE, the role of parties is taken by attributes. Thus, an access structure $\mathbb{A}$ contains all the authorized set of attributes. In this paper, we work with monotone access structures. Next, we define Linear Secret-Sharing Scheme (LSSS)-realizable access structure, used to specify access control policies over user attributes.

Definition 4. ***(Linear Secret Sharing Scheme)*** *[4] Let $\mathcal{P}$ be a set of parties. Let M be a matrix of size $l \times k$, called the share generating matrix. $\rho : [l] \rightarrow \mathcal{P}$ is a row-labeling function that maps rows in M to attributes in $\mathbb{A}$. A secret sharing scheme Π for access structure $\mathbb{A}$ is called linear in $\mathbb{Z}_p$ represented as (M, ρ) if it consists of the following polynomial time algorithms:*

- *Share(M, ρ, s): To generate shares of a secret $s \in \mathbb{Z}_p$, it chooses $z_2, z_3, \cdots, z_k \xleftarrow{R} \mathbb{Z}_p$ and sets $v = (s, z_2, z_3, \cdots, z_k)^T$. It outputs $(M \cdot v)$ as the vector of l shares. The share $\lambda_i = (M_i \cdot v)$ belongs to an attribute $\rho(i)$, where M_i^T is the i^{th} row of M.*
- *Reconstruct(M, ρ, S): This algorithm accepts as input (M, ρ) and a set of attributes $S \in \mathbb{A}$. Let $I = \{i | \rho(i) \in S\}$. It outputs a set $\{\omega_i : i \in I\}$ of secret reconstruction constants such that $\sum_{i \in I} \omega_i \cdot \lambda_i = s$, if $\{\lambda_{\rho(i)} : i \in I\}$ is a valid set of secret shares of the secret s according to Π.*

3 Definition of KP-ABPRE

Definition 5. ***(Key-Policy Attribute-Based Proxy Re-Encryption (KP-ABPRE))*** *A single-hop unidirectional KP-ABPRE scheme consists of the following seven algorithms:*

- ***Setup****$(1^\kappa, U)$: A PPT algorithm run by a certification authority CA that takes the unary encoding of the security parameter κ and an attribute universe description U as inputs. It outputs the public parameters as params and a master secret key MSK.*
- ***KeyGen****$(MSK, (M, \rho), params)$: A PPT algorithm run by CA that takes as input the master secret key MSK, an access structure (M, ρ) for attributes over U and the public parameters params. It outputs a private key $SK_{(M,\rho)}$ corresponding to the access structure (M, ρ).*
- ***Encrypt****$(m, W, params)$: A PPT algorithm that takes as inputs a message $m \in \mathcal{M}$, an attribute set W and params. It outputs a ciphertext C, termed as first-level ciphertext, which can be further re-encrypted.*
- ***Decrypt****$(C, SK_{(M,\rho)}, params)$: A deterministic algorithm that takes as input a first-level ciphertext C encrypted under attribute set W, a private key $SK_{(M,\rho)}$, and params. If $W \models (M, \rho)$, it outputs the message $m \in \mathcal{M}$, else output an error symbol $\perp$ indicating C is invalid.*
- ***ReKeyGen****$(SK_{(M,\rho)}, (M, \rho), (M', \rho'), params)$: A PPT algorithm run by the delegator that takes as input a private key $SK_{(M,\rho)}$ corresponding to an access structure (M, ρ), an access structure (M', ρ') and params. It outputs a re-encryption key $RK_{(M,\rho)\rightarrow(M',\rho')}$, that can perform re-encryption of ciphertexts under attribute set $W \models (M, \rho)$ towards attribute set $W' \models (M', \rho')$.*
- ***ReEncrypt****$(C, RK_{(M,\rho)\rightarrow(M',\rho')}, params)$: A PPT algorithm run by the proxy that takes as input a first-level ciphertext C encrypted under an attribute set W, a re-encryption key $RK_{(M,\rho)\rightarrow(M',\rho')}$ and params. It outputs a re-encrypted ciphertext D if $W \models (M, \rho)$ or an error symbol $\perp$ indicating C is invalid. The ciphertext D cannot be further re-encrypted, also termed as second-level ciphertext.*
- ***ReDecrypt****$(D, SK_{(M',\rho')}, params)$: A deterministic algorithm that takes as input a second-level ciphertext D encrypted under an attribute set W', a private key $SK_{(M',\rho')}$ and params. If $W' \models (M', \rho')$, it outputs the message $m \in \mathcal{M}$ or an error symbol $\perp$ indicating D is invalid.*

The consistency of a KP-ABPRE scheme for any given public parameters $params$, private keys $SK_{(M,\rho)} \leftarrow KeyGen(MSK, (M,\rho), params)$, $SK_{(M',\rho')} \leftarrow KeyGen(MSK, (M',\rho'), params)$ and re-encryption keys $RK_{(M,\rho)\rightarrow(M',\rho')} \leftarrow ReKeyGen(SK_{(M,\rho)}, (M,\rho), (M',\rho'), params)$ and $\forall m \in \mathcal{M}$, the following equations hold:

1. Consistency between encryption and decryption:
   $Decrypt(C, SK_{(M,\rho)}, params) = m$, where $C \leftarrow Encrypt(m, W, params)$.
2. Consistency between encryption, proxy re-encryption and decryption:
   $ReDecrypt((ReEncrypt(C, RK_{(M,\rho)\rightarrow(M',\rho')}, params)), SK_{(M',\rho')}, params) = m$, where $C = Encrypt(m, W, params)$ and $W \models (M,\rho)$.

4 Security Model

Our game-based definitions of selective security of a single-hop unidirectional KP-ABPRE against chosen ciphertext attack are adaptations of the definitions of CCA security for KP-ABPRE systems in [10]. A KP-ABPRE scheme is $IND\text{-}PRE\text{-}CCA$ secure if no PPT adversary has a non-negligible advantage in the below game defined next between the challenger $\mathcal{C}$ and adversary $\mathcal{A}$. In this model, the adversary $\mathcal{A}$ needs to fix the target access structure (M^*, ρ^*) beforehand in the Initialization phase.

Game Template of $IND\text{-}PRE\text{-}CCA$ security:

- **Initialization:** $\mathcal{A}$ outputs a target access structure (M^*, ρ^*) on which it wishes to be challenged. $\mathcal{C}$ runs $Setup(1^\kappa, U)$ and sends the public parameter $params$ to $\mathcal{A}$.
- **Phase 1:** $\mathcal{A}$ issues queries to the following oracles simulated by $\mathcal{C}$:
  - Private Key Extraction($\mathcal{O}_{SK}(M,\rho)$): On input of an access structure (M,ρ), compute its corresponding private key $SK_{(M,\rho)}$ and return to $\mathcal{A}$.
  - Re-encryption Key Generation($\mathcal{O}_{RK}((M,\rho), (M',\rho'))$): Given two access structures (M,ρ) and (M',ρ')) as input, compute the re-encryption key $RK_{(M,\rho)\rightarrow(M',\rho')}$ and return it to $\mathcal{A}$.
  - Re-Encryption($\mathcal{O}_{RE}(C, (M,\rho), (M',\rho'))$): On input of a first-level ciphertext C, access structures (M,ρ) and (M',ρ'), compute re-encryption key $RK_{(M,\rho)\rightarrow(M',\rho')}$ and the second level ciphertext $D \leftarrow$ ReEncrypt$(C, RK_{(M,\rho)\rightarrow(M',\rho')}, params)$. Return D to $\mathcal{A}$.
  - Decryption($\mathcal{O}_{Dec}(C, (M,\rho))$): Given a first level ciphertext C and an access structure (M,ρ) as input, decrypt the ciphertext to obtain $m \in \mathcal{M}$. Return m or $\perp$ if the ciphertext is invalid.
  - Re-Decryption($\mathcal{O}_{RD}(D, (M',\rho'))$): Given a second level ciphertext D and an access structure (M',ρ') as input, decrypt the ciphertext to obtain $m \in \mathcal{M}$. Return m or $\perp$ if the ciphertext is invalid.
- **Challenge:** $\mathcal{A}$ decides whether it wants to be challenged with a first level or a second level ciphertext. It outputs two equal length messages m_0 and m_1 in $\mathcal{M}$ to $\mathcal{C}$. On receiving $\{m_0, m_1\}$, $\mathcal{C}$ picks $\psi \in \{0,1\}$ at random, an attribute set $S^* \models (M^*, \rho^*)$ and generates a challenge ciphertext and returns to $\mathcal{A}$.
- **Phase 2:** $\mathcal{A}$ issues queries to the oracles similar to Phase 1, subject to constraints as discussed later.
- **Guess:** $\mathcal{A}$ outputs its guess $\psi' \in \{0,1\}$.

Note that, the actual construction of the challenge ciphertext and the constraints on the queries of $\mathcal{A}$ simulated by $\mathcal{C}$ are defined based on the type of ciphertext that $\mathcal{A}$ opts for. Due to the existence of two levels of ciphertexts, namely *first level* and *second level* ciphertexts, the security game and different adversarial constraints at both levels are shown next.

CCA-Security Game 1 $Exp_{A,1}^{IND-PRE-CCA}(\kappa)$

$(M^*, \rho^*) \leftarrow \mathcal{A}$, params $\leftarrow$ Setup(κ);
$(m_0, m_1, St) \leftarrow \mathcal{A}^{\mathcal{O}_{SK}, \mathcal{O}_{RK}, \mathcal{O}_{RE}, \mathcal{O}_{Dec}, \mathcal{O}_{RD}}(params)$;
$\psi \in_R \{0,1\}$, $\mathbb{C}^* \leftarrow$Encrypt($m_\psi, W \models (M^*, \rho^*)$,params);
$\psi' \leftarrow \mathcal{A}^{\mathcal{O}_{RK}, \mathcal{O}_{RE}, \mathcal{O}_{Dec}, \mathcal{O}_{RD}}(\mathbb{C}^*, St)$; //query constraints are shown in Section 4.1
if $\psi = \psi'$ return 1, else return 0

Game 1: IND-PRE-CCA security game for first level ciphertext. Note that St is the state information maintained by $\mathcal{A}$.

CCA-Security Game 2 $Exp_{A,2}^{IND-PRE-CCA}(\kappa)$

$(M^*, \rho^*) \leftarrow \mathcal{A}$, params $\leftarrow$ Setup(κ);
$(m_0, m_1$,St) $\leftarrow \mathcal{A}^{\mathcal{O}_{SK}, \mathcal{O}_{RK}, \mathcal{O}_{Dec}, \mathcal{O}_{RD}}(params)$;
$\psi \in_R \{0,1\}$, $\mathbb{C}^* \leftarrow$Encrypt($W \models (M^*, \rho^*), m_\psi$,params);
$\psi' \leftarrow \mathcal{A}^{\mathcal{O}_{RK}, \mathcal{O}_{Dec}, \mathcal{O}_{RD}}(\mathbb{C}^*, St)$; //query constraints are shown in Section 4.2
if $\psi = \psi'$ return 1, else return 0

Game 2: IND-PRE-CCA security game for second level ciphertext.

4.1 First Level Ciphertext Security:

For the first-level ciphertext security, $\mathcal{C}$ interacts with $\mathcal{A}$ as per the game template shown above (Game 1), with the following adversarial constraints, where the challenge ciphertext is C^* =Encrypt($m_\psi, S, params$) and $S \models (M^*, \rho^*)$:

- $\mathcal{O}_{SK}(M^*, \rho^*)$ should not be queried by $\mathcal{A}$.
- $\mathcal{O}_{RK}((M^*, \rho^*), (M', \rho'))$ must be not be queried if $\mathcal{O}_{SK}(M', \rho')$ has already been queried for.
- $\mathcal{O}_{SK}(M', \rho')$ must be not be queried if $\mathcal{O}_{RK}((M^*, \rho^*), (M', \rho'))$ has already been queried for.
- $\mathcal{O}_{RE}(C^*, (M^*, \rho^*), (M', \rho'))$ must not be queried if $\mathcal{O}_{SK}((M', \rho'))$ has already been queried upon.
- $\mathcal{O}_{Dec}(C^*, (M^*, \rho^*))$ cannot be queried by $\mathcal{A}$.

– $\mathcal{O}_{RD}(D, (M', \rho'))$ cannot be queried for second level ciphertext D re-encrypted towards (M', ρ') where D is a *challenge derivative* (defined next) of C^*.

Definition 6. ***Challenge Derivative***[9] *A challenge derivative of C^* in the CCA setting is inductively defined as below:*

– *Reflexivity: C^* is a challenge derivative of itself.*
– *Derivative by re-encryption: D is a challenge derivative of C^* if $D \leftarrow \mathcal{O}_{RE}(C^*, (M^*, \rho^*), (M', \rho'))$.*
– *Derivative by re-encryption key: D is a challenge derivative of C^* if $RK_{(M^*,\rho^*)\rightarrow(M',\rho')} \leftarrow \mathcal{O}_{RK}((M, \rho), (M', \rho'))$ and $D = ReEncrypt(C^*, RK_{(M,\rho) \rightarrow(M',\rho')}, params)$.*

Definition 7. *The advantage of any PPT adversary $\mathcal{A}$ denoted by $Adv_{\mathcal{A}}$ in winning the above IND-PRE-CCA game (Game 1) for first level ciphertext which we term as $Exp_{A,1}^{IND-PRE-CCA}(\kappa)$ is shown as*

$$Adv_{\mathcal{A},1}^{IND-PRE-CCA} := \left| Pr\left[Exp_{A,1}^{IND-PRE-CCA}(\kappa) \right] - \frac{1}{2} \right|$$

where the probability is over the coin tosses of challenger $\mathcal{C}$ and adversary $\mathcal{A}$.

The scheme is *IND-PRE-CCA* secure for the first level ciphertext against any t-time adversary $\mathcal{A}$ that makes atmost q_{SK}, q_{RK}, q_{ReEnc}, q_{Dec} and q_{RD} queries to $\mathcal{O}_{SK}$, $\mathcal{O}_{RK}$, $\mathcal{O}_{RE}$, $\mathcal{O}_{Dec}$ and $\mathcal{O}_{RD}$ oracles respectively, if the advantage of $\mathcal{A}$ is negligibly small: $Adv_{\mathcal{A},1}^{IND-PRE-CCA} \leq \epsilon$.

4.2 Second Level Ciphertext Security:

For the second-level ciphertext security game (Game 2), the adversarial constraints on $\mathcal{A}$ are given below, where the challenge ciphertext is D^* =ReEncrypt$(C, RK_{(M',\rho')\rightarrow(M^*,\rho^*)}, params)$ and C is a first level encryption of m_ψ under the delegator's attribute set S' satisfying access structure (M', ρ'). Note that for single-hop KP-ABPRE schemes, $\mathcal{A}$ is given access to all possible re-encryption keys. As a result, there is no need to provide $\mathcal{A}$ with the re-encryption oracle.

– $\mathcal{O}_{SK}((M^*, \rho^*))$ should not be queried by $\mathcal{A}$.
– $\mathcal{O}_{RD}(D^*, (M^*, \rho^*))$ must not be queried.

Definition 8. *The advantage of any PPT adversary $\mathcal{A}$ denoted by $Adv_{\mathcal{A}}$ in winning the above IND-PRE-CCA game (Game 2) for second level ciphertext which we term $Exp_{A,2}^{IND-PRE-CCA}(\kappa)$ is shown as*

$$Adv_{\mathcal{A},2}^{IND-PRE-CCA} := \left| Pr\left[Exp_{A,2}^{IND-PRE-CCA}(\kappa) \right] - \frac{1}{2} \right|$$

where the probability is over the coin tosses of challenger $\mathcal{C}$ and adversary $\mathcal{A}$.

The scheme is *IND-PRE-CCA* secure for the second level ciphertext against any t-time adversary $\mathcal{A}$ that makes atmost q_{SK}, q_{RK}, q_{ReEnc}, q_{Dec} and q_{RD} queries to $\mathcal{O}_{SK}$, $\mathcal{O}_{RK}$, $\mathcal{O}_{Dec}$ and $\mathcal{O}_{RD}$ oracles respectively, if the advantage of $\mathcal{A}$ is negligibly small: $Adv_{\mathcal{A},2}^{IND-PRE-CCA} \leq \epsilon$.

DSK-Security Game $Exp_{A}^{DSK}(\kappa)$

$(M^*, \rho^*) \leftarrow \mathcal{A}$, params $\leftarrow$ Setup(κ);

$(SK_{(M^*,\rho^*)}, St) \leftarrow \mathcal{A}^{\mathcal{O}_{SK},\mathcal{O}_{RK}}(params)$; //query constraints shown in Section 4.3

if $SK_{(M^*,\rho^*)}$ is a valid private key of (M^*, ρ^*), return 1, else return 0

Game 3: DSK security game for KP-ABPRE schemes.

4.3 Collusion Resistance:

Collusion-resistance, also termed as delegator secret key security (DSK security) prevents a colluding proxy and delegatee to recover the private key of the delegator in full. The game template for DSK security is shown below, adapted from [17], illustrated in Game 3.

- **Setup**: $\mathcal{A}$ outputs a challenge access structure (M^*, ρ^*). $\mathcal{C}$ generates the public parameters *params* using the *Setup* algorithm and returns it to $\mathcal{A}$.
- **Queries**: $\mathcal{A}$ issues queries to the Private Key Extraction($\mathcal{O}_{SK}(M, \rho)$) and Re-encryption Key Generation($\mathcal{O}_{RK}$
  $((M, \rho), (M', \rho'))$) oracle adaptively. It cannot query for the private key of the target access structure (M^*, ρ^*).
- **Output**: $\mathcal{A}$ returns $SK_{(M^*,\rho^*)}$ as the private key of the target access structure (M^*, ρ^*). $\mathcal{A}$ wins the game if $SK_{(M^*,\rho^*)}$ is a valid private key of (M^*, ρ^*).

Definition 9. *The advantage of any PPT adversary $\mathcal{A}$ denoted by $Adv_{\mathcal{A}}$ in winning the DSK-security game (Game 3) given above which we term $Exp_{A}^{DSK}(\kappa)$ is shown as*

$$Adv_{\mathcal{A}}^{DSK} := \Pr[\mathcal{A}\ wins]$$

where the probability is over the coin tosses of challenger $\mathcal{C}$ and adversary $\mathcal{A}$.

The scheme is DSK secure against any t-time adversary $\mathcal{A}$ that makes atmost q_{SK} and q_{RK} queries to $\mathcal{O}_{SK}$ and $\mathcal{O}_{RK}$ oracles respectively, if the advantage of $\mathcal{A}$ is negligibly small: $Adv_{\mathcal{A}}^{DSK} \leq \epsilon$.

5 Analysis of a CCA-secure KP-ABPRE scheme

5.1 Review of the Scheme due to Ge *et al.* [9]

The adaptively CCA-secure KP-ABPRE scheme due to Ge *et al* [9] consists of the following algorithms. It is based on composite order bilinear pairing.

- **Setup($1^\kappa, U$)**: The setup algorithm chooses a bilinear group $\mathbb{G}_0$ of order $N = p_1p_2p_3$. Let $\mathbb{G}_{p_i}$ denote the subgroup of $\mathbb{G}_0$ of order p_i. It chooses $\alpha, \phi \in \mathbb{Z}_N$, $g, \hat{g_1}, \hat{g_2} \in \mathbb{G}_{p_1}$, $X_3 \in \mathbb{G}_{p_3}$. For each attribute i, it picks $s_i \in \mathbb{Z}_N$ and computes $T_i = g^{s_i}$. Let SYM be a CCA-secure one-time symmetric encryption scheme, and Sig $= (\mathcal{G}, \mathcal{S}, \mathcal{V})$ be a strongly unforgeable one-time signature scheme. $H_1 : \mathbb{G}_1 \to \mathbb{Z}_N^*$ and $H_2 : \mathbb{G}_1 \to \{0, 1\}^*$ are collision-resistant hash functions. The master secret key is $MSK = (\alpha, X_3)$. The public parameters returned are $params = (N, g, \hat{g_1}, \hat{g_2}, \hat{g_2}^{\phi}, \hat{e}(g, g)^\alpha, T_i, SYM, (\mathcal{G}, \mathcal{S}, \mathcal{V}), \mathcal{H}_1, \mathcal{H}_2)$.

– **KeyGen(**$MSK, (M,\rho), params$**)**: Given as input an access structure (M,ρ), the trusted authority picks a random vector μ such that $\mu \cdot \mathbf{1} = \alpha$. For each row M_x of the matrix M, it chooses $r_x \in \mathbb{Z}_N$ and $W_x, V_x \in \mathbb{G}_{p_3}$ and computes the private key as: $\forall M_x \in \{M_1, \cdots M_l\}: \ K_{x,1} = g^{M_x \cdot \mu} T_{\rho(x)}^{r_x} W_x, \ K_{x,2} = g^{r_x} V_x$.

– **Encrypt(**$m, W, params$**)**: To encrypt a message $m \in \mathbb{G}_1$ under an attribute set W, the sender encrypts as shown below:
1. Set $C_1 = W$.
2. Select a one-time signature pair $(svk, ssk) \leftarrow \mathcal{G}$.
3. Pick $s \in \mathbb{Z}_N$ and compute $C_0 = m \cdot \hat{e}(g,g)^{\alpha s}, C_2 = g^s, \forall i \in W : C_i = T_i^s, C_3 = (\hat{g_1}^{svk} \hat{g_2}^{\phi})^s$.
4. Run the sign algorithm $\sigma \leftarrow \mathcal{S}(ssk, (C_0, C_2, C_i, C_3))$.
5. Return the original ciphertext $C = (svk, C_0, C_1, C_2, C_i, C_3, \sigma)$.

– **ReKeyGen(**$SK_{(M,\rho)}, (M,\rho), (M',\rho'), params$**)**: The delegator generates a re-encryption key from access structure (M,ρ) to (M',ρ'), as shown below:
1. Choose $\theta \in \mathbb{Z}_p$ and $\delta \in \mathbb{G}_1$. For each row M_x of the matrix M, compute:
$$rk_1 = K_{x,1}^{H_1(\delta)} \cdot T_{\rho(x)}^{\theta}, rk_2 = K_{x,2}^{H_1(\delta)}, rk_3 = g^{\theta}.$$
2. Select an attribute set W' where $W' \models (M',\rho')$.
3. Select a one-time signature pair $(svk', ssk') \leftarrow \mathcal{G}$.
4. Pick $s' \in \mathbb{Z}_N$ and compute $rk_4 = \delta \cdot \hat{e}(g,g)^{\alpha s'}, rk_5 = g^{s'}, \forall i \in W' : rk_{6,i} = T_i^{s'}, rk_7 = (\hat{g_1}^{svk'} \hat{g_2}^{\phi})^{s'}$.
5. Run the sign algorithm $\sigma' \leftarrow \mathcal{S}(ssk', (rk_4, rk_5, rk_{6i}, rk_7))$.
6. Return $RK_{(M,\rho)\to(M',\rho')} = (rk_1, rk_2, rk_3, S', svk', rk_4, rk_5, rk_{6,i}, rk_7, \sigma')$.

– **ReEncrypt(**$C, RK_{(M,\rho)\to(M',\rho')}, params$**)**: On input of an original ciphertext C, the proxy re-encrypts C towards access structure (M',ρ') as below:
1. Check if the following equations hold:
$$\mathcal{V}(svk, \sigma, (C_0, C_2, C_i, C_3)) \stackrel{?}{=} 1, \quad (1)$$
$$\hat{e}(C_2, \hat{g_1}^{svk} \hat{g_2}^{\phi}) \stackrel{?}{=} \hat{e}(g, C_3). \quad (2)$$
2. If the above check fails, return $\perp$. Else, if $W \models (M,\rho)$, compute reconstruction constants ω_x such that $\sum_{\rho(x)\in W} \omega_x M_x = 1$. Next, compute:
$$Q = \prod \frac{\hat{e}(C_2, rk_1)^{\omega_x}}{(\hat{e}(C_{\rho_x}, rk_2)\hat{e}(C_{\rho(x)}, rk_3))^{\omega_x}}.$$
3. Pick a random $key \in \mathbb{G}_1$ and compute $\varPhi_1 = SYM.Enc(H_2(key, G))$, where $G = (C||(W', svk', rk_4, rk_5, rk_{6,i}, rk_7, \sigma')||Q)$.
4. Select an attribute set W'' such that $W'' \models (M',\rho')$ and a one-time signature pair $(svk'', ssk'') \leftarrow \mathcal{G}$. Choose $s'' \in \mathbb{Z}_N$ and compute $C_0'' = key \cdot \hat{e}(g,g)^{\alpha s''}$, $C_2'' = g^{s''}$, $\forall i \in W'' : C_i'' = T_i^{s''}, C_3'' = (\hat{g}_1^{svk''} \hat{g}_2^{\phi})^{s''}$. Run the sign algorithm to generate $\sigma'' \leftarrow \mathcal{S}(ssk'', (C_0'', C_2'', C_i'', C_3''))$. Denote $\varPhi_2 = (W'', svk'', C_0'', C_2'', C_i'', C_3'', \sigma'')$.
5. Return the re-encrypted ciphertext $D = (\varPhi_1, \varPhi_2)$.

– **Decrypt**$(C, SK_{(M,\rho)}, params)$: In order to decrypt a first-level ciphertext C, the decryption algorithm proceeds as below:
1. Check the validity of the ciphertext using equations (1) and (2).
2. If the check fails, output $\perp$ and aborts. Otherwise if $W \not\models (M, \rho)$, output $\perp$ and aborts. Else, compute reconstruction constants ω_x such that $\sum_{\rho(x)\in W} \omega_x M_x = 1$. Next, compute the plaintext as below:

$$m = C_0 \Big/ \prod_{\rho(x)\in S} \frac{\hat{e}(C_2, K_{x,1})^{\omega_x}}{\hat{e}(C_{\rho(x)}, K_{x,2})^{\omega_x}}.$$

– **ReDecrypt**$(D, sk_{(M',\rho')}, params)$: To decrypt a second-level ciphertext $D = (\Phi_1, \Phi_2)$, the decryption algorithm proceeds as below:
- Check if the following equations hold:

$$\mathcal{V}(svk'', \sigma'', (C''_0, C''_2, C''_i, C''_3)) \stackrel{?}{=} 1,$$
$$\hat{e}(C''_2, \hat{g}_1^{svk''} \hat{g}_2^{\phi}) \stackrel{?}{=} \hat{e}(g, C''_3).$$

  If the checks fail, output $\perp$ and abort. Further, if $W'' \not\models M''$, output $\perp$ and abort. Compute the reconstruction constants ω'' such that $\sum_{\rho(x')\in W''} \omega''_x M''_x = 1$. Compute key as below:

$$key = C''_0 \Big/ \prod_{\rho(x')\in W''} \frac{\hat{e}(C''_2, K_{x',1})^{\omega''_x}}{\hat{e}(C''_{\rho x'}, K_{x',2})^{\omega''_x}}.$$

- Run the decryption algorithm $G = SYM.Dec(H_2(key), \Phi_1)$.
- Check if the equations (1) and (2) hold. If fails, return $\perp$ and abort. Otherwise, perform the following checks:

$$\mathcal{V}(svk', \sigma', (rk_4, rk_5, rk_{6,i}, rk_7)) \stackrel{?}{=} 1,$$
$$\hat{e}(rk_5, \hat{g}_1^{svk'} \hat{g}_2^{\phi}) \stackrel{?}{=} \hat{e}(g, rk_7).$$

  If fails, return $\perp$ and abort. If $W' \not\models (M', \rho')$, return $\perp$ and abort.
- Compute the reconstruction constants ω'_x such that $\sum_{\rho(x')\in W'} \omega'_x M'_x = 1$. Next, it computes δ as below:

$$\delta = rk_4 \Big/ \prod_{\rho(x')\in W'} \frac{\hat{e}(rk_5, K_{x',1})^{\omega'_x}}{\hat{e}(rk_{6,\rho(x')}, K_{x',2})^{\omega'_x}}.$$

- Compute $Q^{H_1(\delta)^{-1}} = \hat{e}(g,g)^{s\alpha}$ and output the plaintext $m = C_0/\hat{e}(g,g)^{s\alpha}$.

5.2 Attack on the scheme

In this section we present a CCA-attack on the scheme due to Ge *et al* [9]. Note that the following attack is launched in the second-level ciphertext CCA-security game. In Phase 1 of the security game, the challenger $\mathcal{C}$ provides the adversary $\mathcal{A}$ with all possible re-encryption keys in the system, as per the security definition in [9]. Let $D^* = (\Phi_1^*, \Phi_2^*)$ be the challenge ciphertext generated by $\mathcal{C}$, which is the re-encryption of message m_ψ (selected randomly by $\mathcal{C}$ during challenge phase) from a delegator's attribute set satisfying access structure (M', ρ') towards target access structure (M^*, ρ^*). The CCA attack is demonstrated below.

1. $\mathcal{A}$ parses the re-encryption key $RK_{(M',\rho')\to(M^*,\rho^*)} = (rk_1^*, rk_2^*, rk_3^*, W^*, svk'^*, rk_4^*, rk_5^*, rk_{6,i}^*, rk_7^*, \sigma'^*)$.
2. $\mathcal{A}$ creates a first-level decryption query for a ciphertext generated as below:
   - Set $C_0 = rk_4^*$, $C_1 = W^*$, $C_2 = rk_5^*$ and $C_3 = rk_7^*$.
   - For all attributes $i \in W^*$, set $C_i = rk_{6,i}^*$.
   - Set the signature $\sigma = \sigma'^*$.
   - The ciphertext $C_{A1} = (svk'^*, C_0, C_1, C_2, C_i, C_3, \sigma)$ is passed a parameter to the first level decryption oracle provided by the challenger.
3. The challenger decrypts the ciphertext C_{A1} using Decrypt algorithm to extract δ^* used in the generation of re-encryption key corresponding to the challenge ciphertext D^*.
4. $\mathcal{A}$ parses the challenge ciphertext component $\Phi_2^* = (W''^*, svk''^*, C_0''^*, C_2''^*, C_i''^*, C_3''^*, \sigma''^*)$.
5. Using this second-level challenge ciphertext, the adversary now creates another decryption query for a first level ciphertext generated as below:
   - Set $C_0 = C_0''^*$, $C_1 = W''^*$, $C_2 = C_2''^*$ and $C_3 = C_3''^*$.
   - For all attributes $i \in W''^*$, set $C_i = C_i''^*$.
   - Set the signature $\sigma = \sigma''^*$.
   - The ciphertext $C_{A2} = (svk''^*, C_0, C_1, C_2, C_i, C_3, \sigma)$ is passed a parameter to the first level decryption oracle provided by the challenger.
6. On decryption of C_{A2}, $\mathcal{A}$ receives key^* used in generation of the challenge ciphertext D^*. Therefore, $\mathcal{A}$ can now recover G^* using the symmetric decryption algorithm $G^* = SYM.Dec(H_2(key^*), \Phi_1^*)$.
7. $\mathcal{A}$ parses $G^* = (C||(W^*, svk^*, rk_4^*, rk_5^*, rk_{6,i}^*, rk_7^*, \sigma^*)||Q)$ and then parses $C = (svk, C_0, C_1, C_2, C_i, C_3, \sigma)$.
8. Finally $\mathcal{A}$ computes $m_\psi = \frac{C_0}{Q^{H_1(\delta)^{-1}}}$ as per ReDecrypt() algorithm.

This completes the description of the attack. $\mathcal{A}$ can recover the original message m_ψ re-encrypted by $\mathcal{C}$ towards a target access structure (M^*, ρ^*) in the challenge ciphertext D^*, which successfully breaks the CCA security of the scheme.

6 Analysis of an RCCA-secure KP-ABPRE scheme

6.1 Review of the scheme due to Li *et al.* [11]

The selectively RCCA-secure KP-ABPRE scheme due to Li *et al.* [11] consists of the following algorithms.

- **Setup**$(1^\kappa, U)$: The setup algorithm takes as input the universe description U, where $U = \{0,1\}^*$ and security parameter κ. It chooses groups $\mathbb{G}_0, \mathbb{G}_1$ of prime order p. Let g by a generator of $\mathbb{G}_0$. It randomly picks values $\alpha \in \mathbb{Z}_p$ and $g_1, h \in \mathbb{G}_0$, and sets $MSK = \alpha$. k is a parameter determined by κ and $\{0,1\}^k$ is the message space $\mathcal{M}$. Three cryptographic hash functions are chosen as follows: $F : \{0,1\}^* \to \mathbb{G}_0$,

$H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \rightarrow \{0,1\}^k$. The public parameters $params = \langle g, g_1, h, g^\alpha, F, H_1, H_2, \hat{e}(g,h)^\alpha \rangle$.

– **KeyGen**$(MSK, (M,\rho), params)$: On input of an access structure (M,ρ) where $l \times k$ is the size of access structure M, the trusted authority executes Share(M,ρ,α) to obtain l sets of shares $\lambda_{\rho(i)} = M_i \cdot v$, where it picks $y_2, \cdots y_n \in \mathbb{Z}_p$ and sets $v = (\alpha, y_2, \cdots y_n)$. It chooses $r_1, \cdots r_l \in \mathbb{Z}_p^*$ and computes the private key $SK_{(M,\rho)}$ as below:

$$K_{11} = h^{\lambda_{\rho(1)}} \cdot F(\rho(1))^{r_1}, K_{21} = g^{r_1}, \cdots, K_{1l} = h^{\lambda_{\rho(l)}} \cdot F(\rho(l))^{r_l}, K_{2l} = g^{r_l}$$

It outputs $SK_{(M,\rho)}$ along with the description of (M,ρ).

– **Encrypt**$(m, W, params)$: Given as input a message $m \in \mathcal{M}$ and an attribute set W, the first-level encryption algorithm randomly picks $R \in \mathbb{G}_1$ and computes $s_1 = H_1(R, m)$ and $r = H_2(R)$. It computes the first level encryption C as below:

$$C_0 = R \cdot \hat{e}(g,h)^{\alpha s_1},\ C_1 = g^{s_1},\ C_2 = g_1^{s_1},\ C_3 = r \oplus m, \{C_x = F(x)^{s_1}, \forall x \in W\}$$

It outputs ciphertext $C = \langle C_0, C_1, C_2, C_3, \{C_x, \forall x \in W\}, W \rangle$.

– **ReKeyGen**$(SK_{(M,\rho)}, (M,\rho), (M',\rho'), params)$: To generate a re-encryption key from an access structure (M,ρ) to (M',ρ'), the trusted authority calls the **KeyGen** algorithm and chooses $d \in \mathbb{Z}_p^*$ at random and computes $g^{\alpha d}$, $\{g_1^{\lambda_{\rho(i)} d}\}$ where $\lambda_\rho(i)$ for $i = 1$ to l is the set of l shares corresponding to the access structure (M,ρ). It picks an attribute $W' \models (M',\rho')$ and encrypts $g^{\alpha d}$ with the attributes of W' by computing $CT_1 =$ **Encrypt**$(g^{\alpha d}, W', params)$. It picks random $r'_1 \cdots r'_l \in \mathbb{Z}_p$ and recomputes the re-encryption key $RK_{(M,\rho)\rightarrow(M',\rho')}$ as follows:

$$rk_{11} = h^{\lambda_{\rho(1)}} \cdot F(\rho(1))^{r'_1} \cdot g_1^{\lambda_{\rho(1)} d}, rk_{21} = g^{r'_1}, \cdots, rk_{1l} = h^{\lambda_{\rho(l)}} \cdot F(\rho(l))^{r'_l} \cdot g_1^{\lambda_{\rho(l)} d}, rk_{2l} = g^{r'_l}$$

It returns the re-key as $RK_{(M,\rho)\rightarrow(M',\rho')} = (\{rk_{1i}, rk_{2i}$ for i=1 to $l\}, CT_1)$.

– **ReEncrypt**$(C, RK_{(M,\rho)\rightarrow(M',\rho')}, params)$: Given as input a first-level ciphertext C and a re-encryption key $RK_{(M,\rho)\rightarrow(M',\rho')}$, the re-encryption algorithm executes Reconstruct (M,ρ,W) to obtain a set $\{\omega_i : i \in I\}$ of secret reconstruction constants where $I = \{i \in [l] : att_{\rho(i)} \in W\}$. If $W \models (M,\rho)$, then the relation $\sum_{i \in I} \omega_i \lambda_{\rho(i)} = \alpha$ implicitly holds. It computes the re-encrypted ciphertext component D_4 as below:

$$\begin{aligned} D_4 &= \frac{\hat{e}(C_1, \prod_{i \in I} rk_{1i}^{\omega_i})}{\prod_{i \in I} \hat{e}(rk_{2i}, C_{\rho(i)}^{\omega_i})} \\ &= \hat{e}(g,h)^{s_1 \alpha} \hat{e}(g, g_1)^{(s_1 \alpha d)} \end{aligned}$$

It sets $D_0 = C_0, D_1 = C_1, D_2 = C_2, D_3 = C_3, D_5 = CT_1$ and outputs second level ciphertext $D = \langle D_0, D_1, D_2, D_3, D_4, D_5 \rangle$.

– **Decrypt**$(C$ or $D, SK_{(M',\rho')}, params)$: If the input to this algorithm is a first level ciphertext C encrypted under (M,ρ), the decryption algorithm invokes **ReEncrypt**$(C, RK_{(M,\rho)\rightarrow(M',\rho')}, params)$ to obtain $D = \langle D_0, D_1, D_2, D_3, D_4, D_5 \rangle$. If the input is a second level ciphertext D and a private key $SK_{(M',\rho')}$, the algorithm first decrypts $D_5 = \langle C'_0, C'_1, C'_2, C'_3, \{C'_x\}, W'\rangle$ by checking if $W' \models (M', \rho')$ and computing the set of reconstruction constants $\{\omega'_i : i \in I\}$ where $I = \{i \in [l] : att_{\rho(i)'} \in W'\}$ using Reconstruct (M', ρ', W') such that $\sum_{\rho'(i)\in W'} \omega'_i M'_i = 1$ holds implicitly. It computes:

$$CT_2 = \frac{\hat{e}(C'_1, \prod_{i\in I} K_{1i}^{\omega'_i})}{\prod_{i\in I} \hat{e}(K_{2i}, C'^{\omega'_i}_{\rho(i)})} \\ = \hat{e}(g,h)^{s_1\alpha}$$

It extracts $g^{\alpha d} = D_0/CT_2$. It computes $CT_3 = \hat{e}(g^{\alpha d}, D_2) = \hat{e}(g, g_1)^{s_1\alpha d}$. Next, it computes $R = D_0 \cdot CT_3/D_4$, $m = D_3 \oplus H_2(R)$ and $s = H_1(R, m)$. If $D_0 \stackrel{?}{=} R \cdot \hat{e}(g,h)^{\alpha s}$ and $D_4 \stackrel{?}{=} \hat{e}(g,h)^{s_1\alpha} \cdot \hat{e}(g^{\alpha d}, g_1^{s_1)}$, it outputs m. otherwise return $\perp$.

6.2 Attack on the scheme

In this section, we present an RCCA-attack on the scheme due to Li *et al* [11]. The following attack is launched in the first-level ciphertext RCCA-security game. Suppose that $C^* = \langle C^*_0, C^*_1, C^*_2, C^*_3, \{C^*_x\}, W^*\rangle$ is the challenge ciphertext generated by $\mathcal{C}$, which is the encryption of message m_ψ (selected randomly by $\mathcal{C}$ during challenge phase from messages $\{m_0, m_1\}$) towards a delegator's attribute set $W^* \models (M^*, \rho^*)$ where (M^*, ρ^*) is the target access structure. The RCCA attack launched by the adversary $\mathcal{A}$ is demonstrated below:

1. The adversary $\mathcal{A}$ picks $\beta \in \mathbb{Z}_p^*$ at random.
2. It computes $C_0" = C^*_0 \cdot \hat{e}(g,h)^{\alpha\beta} = R \cdot \hat{e}(g,h)^{\alpha(s^*_1+\beta)}$.
3. It computes $C_1" = C^*_1 \cdot g^\beta = g^{s^*_1+\beta}$.
4. It computes $C_2" = C^*_2 \cdot g_1^\beta = g_1^{s^*_1+\beta}$.
5. It picks $C_3" \in \{0,1\}^k$ at random.
6. For all $x \in W^*$, it computes $\{C_x" = C^*_x \cdot F(x)^\beta = F(x)^{s^*_1+\beta}\}$.
7. It constructs a first level ciphertext $C" = \langle C_0", C_1", C_2", C_3", \{C_x"\}, W^*\rangle$.
8. It queries the re-encryption oracle $\mathcal{O}_{RE}(C", (M^*, \rho^*), (M', \rho'))$ such that $\mathcal{O}_{SK}(M', \rho')$ is already queried upon for the access structure (M', ρ').
9. The returned second level ciphertext is $D = \langle D_0, D_1, D_2, D_3, D_4, D_5\rangle$, such that the ciphertext component $D_4 = \hat{e}(g,h)^{\alpha(s^*_1+\beta)} \cdot \hat{e}(g, g_1)^{(s^*_1+\beta)\alpha d}$.
10. $\mathcal{A}$ parses $D_5 = \langle C'_0, C'_1, C'_2, C'_3, \{C'_x\}, W'\rangle$. Since $SK_{(M',\rho')}$ is known to $\mathcal{A}$, it computes CT_2 as shown next, by computing the set of reconstruction constants $\{\omega'_i : i \in I\}$ where $I = \{i \in [l] : att_{\rho(i)'} \in W'\}$ by invoking Reconstruct (M', ρ', W') such that $\sum_{\rho'(i)\in W'} \omega'_i M'_i = 1$ implicitly holds.

$$CT_2 = \frac{\hat{e}(C'_1, \prod_{i\in I} K_{1i}^{\omega'_i})}{\prod_{i\in I} \hat{e}(K_{2i}, C'^{\omega'_i}_{\rho(i)})} \\ = \hat{e}(g,h)^{s_1\alpha}$$

11. It extracts $g^{\alpha d} = D_0/CT_2$ and computes $CT_3 = \hat{e}(g^{\alpha d}, D_2) = \hat{e}(g, g_1)^{(s^*_1+\beta)\alpha d}$.
12. It computes $R^* = D_0 \cdot CT_3/D_4$ and $r^* = H_2(R^*)$.
13. Finally, $\mathcal{A}$ computes $m_\psi = r \oplus C^*_3$.

This completes the description of the attack. $\mathcal{A}$ can recover the original message m_ψ encrypted by the challenger towards the target access structure (M^*, ρ^*) as a first level challenge ciphertext C^*, which successfully breaks the RCCA security of the scheme. Note that, as per the security definition for RCCA-secure PRE by Libert *et al.* [16], the adversary cannot issue a decryption query for any second level ciphertext D if **Decrypt**$(D, SK_{(M',\rho')}) \in \{m_0, m_1\}$. Accordingly, the adversary does not query the decryption of any challenge ciphertext derivative in the attack. In fact, the ciphertext component C_3'' is picked at random. The RCCA attack could be launched due to the absence of ciphertext validation in the re-encryption algorithm of the given construction.

7 Our Unidirectional CCA-secure KP-ABPRE Scheme

7.1 Our Construction

- **Setup**$(1^\kappa, U)$: The setup algorithm chooses two bilinear groups $\mathbb{G}_0, \mathbb{G}_1$ of prime order p. Let g and g_1 be the generators of $\mathbb{G}_0$, and $\hat{e} : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ denote an efficiently computable bilinear map. It chooses a random exponent $\alpha \in \mathbb{Z}_p^*$ and computes $Y = \hat{e}(g, g)^\alpha$. It picks $h_0 \in \mathbb{G}_0$ and for every attribute $att_y \in U$, picks $h_y \in \mathbb{G}_0$. l_m and l_σ are parameters determined by κ and $\{0,1\}^{l_m}$ is the size of the message space $\mathcal{M}$. Let $|U| = n$ be the size of the attribute universe. Five cryptographic hash functions are chosen as follows.

$$\begin{aligned}
&\mathcal{H}_1 : \{0,1\}^{l_m + l_\sigma} \to \mathbb{Z}_p^*,\\
&\mathcal{H}_2 : \mathbb{G}_1 \to \{0,1\}^{l_m + l_\sigma},\\
&\mathcal{H}_3 : \{0,1\}^* \times \mathbb{G}_1^3 \to \mathbb{G}_0,\\
&\mathcal{H}_4 : \{0,1\}^{l_m} \to \mathbb{Z}_p^*,\\
&\mathcal{H}_5 : \{0,1\}^* \times \mathbb{G}_1^2 \to \mathbb{G}_0.
\end{aligned}$$

  The hash functions are modelled as random oracles in the security proof. The public parameters returned are $params = \langle \mathbb{G}_0, \mathbb{G}_1, \hat{e}, p, g, g_1, h_0, h_1, \cdots h_n, Y, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4, \mathcal{H}_5 \rangle$. The master secret key MSK is α.

- **KeyGen**$(MSK, (M, \rho), params)$: On input of an access structure (M, ρ) where $l \times k$ is the size of matrix M, the CA executes Share(M, ρ, α) to obtain a set of l shares $\lambda_{\rho_i} = M_i \cdot v$, where $v \in_R \mathbb{Z}_p^k$, such that $v \cdot \mathbf{1} = \alpha$. Note that $\mathbf{1} = (1, 0, \cdots, 0)$ is a vector of length k. For each row M_i of the matrix M, it picks $r_i \in \mathbb{Z}_p^*$ and computes:

$$K_i = g^{\lambda_{\rho(i)}} (h_0 h_{\rho(i)})^{r_i}, K_i' = g^{r_i},\ K_i'' = \{K_{iy}'' : K_{iy}'' = h_y^{r_i}, \forall y \in [n] \backslash \{\rho(i)\}\}.$$

  It returns $SK_{(M,\rho)} = \langle (M, \rho), \{\forall i \in [l] : K_i, K_i', K_i''\} \rangle$ to the user.

- **Encrypt**$(m, W, params)$: Given as input a message $m \in \mathcal{M}$ and an attribute set W, the first-level encryption algorithm encrypts m as below:
  - Select $\sigma \in \{0,1\}^{l_\sigma}$.
  - Compute $s = \mathcal{H}_1(m||\sigma)$.
  - Compute $C_0 = (m||\sigma) \oplus \mathcal{H}_2(Y^s)$.

- Compute $C_1 = g^s$.
- Compute $C_2 = g_1^s$.
- Compute $C_3 = \left(h_0 \prod_{att_y \in W} h_y\right)^s$.
- Compute $C_4 = \left(\mathcal{H}_3(W, C_0, C_1, C_2, C_3)\right)^s$.
- Return the first-level ciphertext $C = \langle C_0, C_1, C_2, C_3, C_4, W \rangle$.

– **Decrypt**$(C, SK_{(M,\rho)}, params)$: On input of a first level ciphertext C and a private key $SK_{(M,\rho)}$, the decryption algorithm works as below:

1. First check if the ciphertext is well-formed as below:

$$\hat{e}(g, C_2) \stackrel{?}{=} \hat{e}(g_1, C_1) \tag{3}$$

$$\hat{e}(g, C_3) \stackrel{?}{=} \hat{e}(C_1, h_0 \prod_{att_y \in W} h_y), \tag{4}$$

$$\hat{e}(g, C_4) \stackrel{?}{=} \hat{e}(C_1, \mathcal{H}_3(W, C_0, C_1, C_2, C_3)). \tag{5}$$

2. If the checks fail, output $\perp$ and abort.
3. Otherwise, execute Reconstruct (M, ρ, W) to obtain a set $\{\omega_i : i \in I\}$ of secret reconstruction constants where $I = \{i \in [l] : att_{\rho(i)} \in W\}$. If $W \models (M, \rho)$, then the relation $\sum_{i \in I} \omega_i \lambda_{\rho(i)} = \alpha$ implicitly holds.
4. Compute:

$$E_1 = \prod_{i \in I} \Big(K_i \cdot \prod_{att_y \in W, y \neq \rho(i)} K''_{i,y}\Big)^{\omega_i}, \; E_2 = \prod_{i \in I} (K'_i)^{\omega_i}$$

5. Compute the plaintext:

$$(m||\sigma) = C_0 \oplus \mathcal{H}_2\left(\frac{\hat{e}(C_1, E_1)}{\hat{e}(C_3, E_2)}\right). \tag{6}$$

6. If $C_1 \stackrel{?}{=} g^{\mathcal{H}_1(m||\sigma)}$, return the plaintext m, else return $\perp$.

– **ReKeyGen**$(SK_{(M,\rho)}, (M, \rho), (M', \rho'), params)$: To generate a re-encryption key from an access structure (M, ρ) to (M', ρ'), the re-encryption key generation algorithm computes the re-key $RK_{(M,\rho) \to (M',\rho')}$ as follows:

1. Choose $\theta \in \mathbb{Z}_p$. Pick $\delta \in \{0,1\}^{l_m}$ and $\gamma \in \{0,1\}^{l_\sigma}$. For each row M_i of the matrix M of size $l \times k$, compute:

$$rk_{1i} = K_i^{\mathcal{H}_4(\delta)} \cdot \left(h_0 h_{\rho(i)}\right)^\theta,$$
$$rk_{2i} = (K'_i)^{\mathcal{H}_4(\delta)} \cdot g^\theta,$$
$$rk_{3i} = \{rk_{3iy} : rk_{3iy} = \left(K''^{H_4(\delta)}_{iy} \cdot h_{iy}^\theta\right), \forall y \in [n] \backslash \{\rho(i)\}\}.$$

2. Compute $s' = \mathcal{H}_1(\delta||\gamma)$.
3. Compute $rk_4 = (\delta||\gamma) \oplus \mathcal{H}_2(Y^{s'})$.
4. Compute $rk_5 = g^{s'}$.
5. Pick an attribute set $W' \models (M', \rho')$.
6. Compute $rk_6 = \left(h_0 \prod_{att_y \in W'} h_y\right)^{s'}$.

7. Compute $rk_7 = \left(\mathcal{H}_5(W', rk_4, rk_5, rk_6)\right)^{s'}$.
8. Return the re-encryption key $RK_{(M,\rho)\to(M',\rho')} = (\{\forall i \in [l] : rk_{1i}, rk_{2i}, rk_{3i}\}, rk_4, rk_5, rk_6, rk_7, W')$.

– **ReEncrypt**$(C, RK_{(M,\rho)\to(M',\rho')}, params)$: Given as input a first-level ciphertext C and a re-encryption key $RK_{(M,\rho)\to(M',\rho')}$, the re-encryption algorithm re-encrypts the first-level ciphertext as below:
  1. It checks the validity of the ciphertext C using Equations 3, 4, 5.
  2. Check the validity of the re-encryption key by checking if the following equations hold:

$$\hat{e}(g, rk_6) \stackrel{?}{=} \hat{e}(h_0 \prod_{att_y \in W'} h_y, rk_5) \tag{7}$$

$$\hat{e}(rk_7, g) \stackrel{?}{=} \hat{e}(\mathcal{H}_5(W', rk_4, rk_5, rk_6), rk_5). \tag{8}$$

  If the above checks fail, return $\perp$.
  3. Otherwise, if $S \models M$, compute a set $\{\omega_i : i \in I\}$ of secret reconstruction constants where $I = \{i \in [l] : att_{\rho(i)} \in W\}$ using Reconstruct (M, ρ, W) such that $\sum_{\rho(i)\in S} \omega_i M_i = 1$ holds implicitly. Compute:

$$re_1 = \prod_{i\in I} \Big(rk_{1i} \cdot \prod_{att_y \in W, y\neq\rho(i)} rk_{3i}\Big)^{\omega_i},\ re_2 = \prod_{i\in I}(rk_{2i})^{\omega_i}$$

  4. Next, compute:

$$D_0 = \frac{\hat{e}(C_1, re_1)}{\hat{e}(C_3, re_2)}$$

$$= Y^{s\mathcal{H}_4(\delta)}. \tag{9}$$

  5. Set $D_1 = C_0,\ D_2 = C_1,\ D_3 = rk_4,\ D_4 = rk_5,\ D_5 = rk_6,\ D_6 = rk_7$.
  6. Return the second level ciphertext as $D = (D_0, D_1, D_2, D_3, D_4, D_5, D_6, W')$.

– **ReDecrypt**$(D, SK_{(M',\rho')}, params)$: Inorder to decrypt a second-level ciphertext D, the decryption algorithm proceeds as below:
  1. Check if the ciphertext is well-formed by checking the following equations:

$$\hat{e}(D_4, h_0 \prod_{att_y \in W'} h_y) \stackrel{?}{=} \hat{e}(g, D_5) \tag{10}$$

$$\hat{e}(D_6, g) \stackrel{?}{=} \hat{e}(\mathcal{H}_5(W', D_3, D_4, D_5), D_4). \tag{11}$$

  2. If the check fails, abort and return $\perp$.
  3. Otherwise, if $W' \models (M', \rho')$, compute the set of reconstruction constants $\{\omega'_i : i \in I\}$ where $I = \{i \in [l] : att_{\rho(i)'} \in W'\}$ using Reconstruct (M', ρ', W') such that $\sum_{\rho'(i)\in W'} \omega'_i M'_i = 1$ holds implicitly. Compute:

$$E'_1 = \prod_{i\in I} \Big(K_i \cdot \prod_{att_y \in W', y\neq\rho(i)} K''_{i,y}\Big)^{\omega'_i},\ E'_2 = \prod_{i\in I}(K'_i)^{\omega'_i}$$

4. Compute δ as below:

$$(\delta||\gamma) = D_3 \oplus \mathcal{H}_2\left(\frac{\hat{e}(D_4, E_1')}{\hat{e}(D_5, E_2')}\right). \tag{12}$$

5. If $D_4 \stackrel{?}{=} g^{\mathcal{H}_1(\delta||\gamma)}$ does not hold, return $\perp$. Otherwise, extract the plaintext as below:

$$(m||\sigma) = D_1 \oplus \mathcal{H}_2(D_0^{1/\mathcal{H}_4(\delta)}). \tag{13}$$

6. If $D_2 \stackrel{?}{=} g^{\mathcal{H}_1(m||\sigma)}$, output m, otherwise return $\perp$.

7.2 Correctness

– Correctness of first-level decryption from Equation (6):

$$\begin{aligned}
RHS &= C_0 \oplus \mathcal{H}_2\left(\frac{\hat{e}(C_1, E_1)}{\hat{e}(C_3, E_2)}\right) \\
&= C_0 \oplus \mathcal{H}_2\left(\frac{\hat{e}(g^s, \prod_{i \in I}(g^{\lambda_{\rho(i)}} h_0^{r_i} h_{\rho(i)}^{r_i} \prod_{att_y \in W, y \notin \rho(i)} h_y^{r_i})^{\omega_i})}{\hat{e}(h_0^s \prod_{att_y \in W} h_y^s, \prod_{i \in I}(g^{r_i})^{\omega_i})}\right) \\
&= (m||\sigma) \oplus \mathcal{H}_2(Y^s) \oplus \mathcal{H}_2(\hat{e}(g,g)^{\alpha s}). \\
&= (m||\sigma). \\
&= LHS.
\end{aligned}$$

– Correctness of computation of re-encrypted ciphertext component D_0 from Equation (9):

$$\begin{aligned}
RHS &= \frac{\hat{e}(C_1, re_1)}{\hat{e}(C_3, re_2)} \\
&= \frac{\hat{e}\big(g^s, \prod_{i \in I}\big(rk_{1i} \cdot \prod_{att_y \in W, y \notin \rho(i)} rk_{3i}\big)^{\omega_i}\big)}{\hat{e}((h_0 \prod_{att_y \in W} h_y)^s, \prod_{i \in I}(rk_{2i})^{\omega_i})} \\
&= \frac{\hat{e}(g^s, \prod_{i \in I}(g^{\lambda_i \cdot \omega_i})^{\mathcal{H}_4(\delta)}) \cdot \hat{e}(g^s, (h_0 \prod_{att_y \in W} h_y)^{(r_i \mathcal{H}_4(\delta) + \theta)\omega_i})}{\hat{e}((h_0 \prod_{att_y \in W} h_y)^s, \prod_{i \in I}(g^{r_i \mathcal{H}_4(\delta) + \theta})^{\omega_i})} \\
&= \hat{e}\big(g^s, g^{(\sum_{i \in I} \omega_i \lambda_{\rho_i})\mathcal{H}_4(\delta)}\big) \\
&= Y^{s\mathcal{H}_4(\delta)} \\
&= LHS.
\end{aligned}$$

– Correctness of re-decryption in computing δ from Equation (12):

$$\begin{aligned}
LHS &= D_3 \oplus \mathcal{H}_2\left(\frac{\hat{e}(D_4, E_1')}{\hat{e}(D_5, E_2')}\right) \\
&= D_3 \oplus \mathcal{H}_2\left(\frac{\hat{e}(g^{s'}, \prod_{i\in I}(g^{\lambda_{\rho(i)}} h_0^{r_i'} h_{\rho(i)}^{r_i'} \prod_{att_y \in W, y\notin \rho(i)} h_y^{r_i'})^{\omega_i'})}{\hat{e}(h_0^{s'} \prod_{att_y\in W'} h_y^{s'}, \prod_{i\in I}(K_i')^{\omega_i'})}\right) \\
&= (\delta||\gamma) \oplus \mathcal{H}_2(\hat{e}(g,g)^{\alpha s'}) \oplus \mathcal{H}_2(\hat{e}(g,g)^{s'\alpha}) \\
&= (\delta||\gamma). \\
&= RHS.
\end{aligned}$$

– Correctness of re-decryption from Equation (13):

$$\begin{aligned}
RHS &= D_1 \oplus \mathcal{H}_2(D_0^{1/\mathcal{H}_4(\delta)}) \\
&= (m||\sigma) \oplus \mathcal{H}_2(Y^s) \oplus \mathcal{H}_2((Y^{s\mathcal{H}_4(\delta)})^{1/\mathcal{H}_4(\delta)}) \\
&= (m||\sigma) \\
&= LHS.
\end{aligned}$$

7.3 Security Proof

Lemma 1. *[2] Let (M, ρ) be an LSSS realising access structure $\mathbb{A}$ over a set of parties $\mathcal{P}$. Let M be a share generating matrix of size $l \times k$. For an attribute set $S \subset U$ such that $S \notin \mathbb{A}$, there exists a polynomial time algorithm that outputs a vector $w = (-1, w_2, w_3, \cdots, w_k) \in \mathbb{Z}_p^k$ such that for all rows M_i where $\rho(i) \in S$, it holds that $M_i \cdot w = 0$.*

First Level Ciphertext Security

Theorem 1. *If a $(t, \epsilon)IND\text{-}PRE\text{-}CCA$ adversary $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the CCA security of the given KP-ABPRE scheme for first level ciphertext, with access to random oracles $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4, \mathcal{H}_5$, then there exists an algorithm $\mathcal{C}$ that solves the n-DBDHE problem with advantage ϵ' within time t' where,*

$$\epsilon' \geq \frac{1}{q_{\mathcal{H}_2}}\left(2\epsilon - \frac{q_{\mathcal{H}_1}}{2^{l_m+l_\sigma}} - \frac{q_{ReEnc}}{p} - q_{Dec}\left(\frac{q_{\mathcal{H}_1}}{(2^{l_m+l_\sigma})} + \frac{1}{p}\right)\right),$$

$$\begin{aligned}
t' \leq t &+ (q_{\mathcal{H}_3} + q_{\mathcal{H}_5} + (n+2)lq_{SK} + (n+6)lq_{RK} + 7q_{ReEnc} + 3q_{Dec} + 5q_{RD})t_e \\
&+ (8q_{ReEnc} + 8q_{Dec} + 6q_{RD})t_{bp}.
\end{aligned}$$

where l is the number of rows in access structure (M, ρ), n is the size of attribute universe and t_e, t_{bp} denote the time taken for exponentiation and bilinear pairing operations respectively.

Proof. If an adversary $\mathcal{A}$ that asks atmost $q_{\mathcal{H}_i}$ random oracle queries to $\mathcal{H}_i$ where $i \in \{1, 2, \cdots, 5\}$ breaks the IND-PRE-CCA security for the first level ciphertexts of the KP-ABPRE scheme, we show that we can construct a PPT algorithm $\mathcal{C}$ that can break the n-DBDHE assumption with non-negligible advantage. The algorithm $\mathcal{C}$ accepts as input the n-DBDHE challenge $\langle (g, g^b, g^a, g^{a^2}, \cdots, g^{a^n}, g^{a^{n+2}}, \cdots, g^{a^{2n}}) \in \mathbb{G}, T \in \mathbb{G}_1 \rangle$ and plays the role of a challenger in the following CCA-game with the adversary $\mathcal{A}$.

– **Initialization:** The adversary $\mathcal{A}$ shares the target access structure (M^*, ρ^*) with the challenger $\mathcal{C}$ on which it wishes to be challenged and the challenger $\mathcal{C}$ picks any $W^* \models (M^*, \rho^*)$ as the target attribute set. $\mathcal{C}$ generates the public parameters as follows. It picks $\alpha' \in_R \mathbb{Z}_p^*$, and implicitly sets msk $\alpha = \alpha' + a^{n+1}$. It computes $Y = e(g, g)^{\alpha'} \cdot e(g^a, g^{a^n})$.

$$\text{In fact, } Y = e(g, g)^{\alpha'} \cdot e(g^a, g^{a^n}) \\ = e(g, g)^{\alpha' + (a^{n+1})} \\ = e(g, g)^{\alpha}.$$

Let $g_1 = g^{\beta}$, where $\beta \in_R \mathbb{Z}_p^*$ is known to the challenger $\mathcal{C}$. For all attributes $att_y \in U$, $\mathcal{C}$ computes h_y as below:
- Pick $t_y \in \mathbb{Z}_p^*$.
- Compute $h_y = g^{a^{n+1-y}} \cdot g^{t_y}$.

The challenger $\mathcal{C}$ picks $t_0 \in_R \mathbb{Z}_p^*$ and computes $h_0 = (\prod_{att_y \in W^*} h_y)^{-1} \cdot g^{t_0}$. It maintains two lists L_{SK} and L_{RK} to store the list of private keys and the re-encryption keys generated by $\mathcal{C}$ and contain tuples of the form :
- $L_{SK} : \langle (M, \rho), SK_{(M,\rho)} \rangle$.
- $L_{RK} : \langle (M, \rho)(M', \rho'), RK_{(M,\rho) \to (M',\rho')} \rangle$.

Both the lists are initially empty. $\mathcal{C}$ returns $params : \langle p, g, g_1, \hat{e}, Y, h_0, h_1, \cdots, h_n, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4, \mathcal{H}_5 \rangle$.

– **Phase 1: Oracle Queries:** $\mathcal{C}$ responds as shown below:
-$\mathcal{H}_1(m||\sigma)$: $\mathcal{C}$ maintains a list $L_{\mathcal{H}_1}$ that contains tuples of the form $\langle m \in \{0,1\}^{l_m}, \sigma \in \{0,1\}^{l_\sigma}, s \in \mathbb{Z}_p^* \rangle$. If the given query already appears in $L_{\mathcal{H}_1}$ in a tuple $\langle m, \sigma, s \rangle$, return the predefined value s. Otherwise, pick $s \in_R \mathbb{Z}_p^*$, add tuple $\langle m, \sigma, s \rangle$ to $L_{\mathcal{H}_1}$ and respond with $\mathcal{H}_1(m||\sigma) = s$.

-$\mathcal{H}_2(\chi)$: $\mathcal{C}$ maintains a list $L_{\mathcal{H}_2}$ that contains tuples of the form $\langle \chi \in \mathbb{G}_1, \nu \in \{0,1\}^{l_m + l_\sigma} \rangle$. If the given query already appears in $L_{\mathcal{H}_2}$ in a tuple $\langle \chi, \nu \rangle$, return the predefined value ν. Otherwise, pick $\nu \in_R \{0,1\}^{l_m + l_\sigma}$, add tuple $\langle \chi, \nu \rangle$ to $L_{\mathcal{H}_2}$ and respond with $\mathcal{H}_2(\chi) = \nu$.

-$\mathcal{H}_3(W, C_0, C_1, C_2, C_3)$: $\mathcal{C}$ maintains a list $L_{\mathcal{H}_3}$ that contains tuples of the form $\langle W, C_0 \in_R \{0,1\}^{l_m + l_\sigma}, C_1 \in \mathbb{G}_0, C_2 \in \mathbb{G}_0, C_3 \in \mathbb{G}_0, \upsilon \in \mathbb{Z}_p^*, \mu \in_R \mathbb{G}_0 \rangle$. If the given query already appears in $L_{\mathcal{H}_3}$ in a tuple $\langle W, C_0, C_1, C_2, C_3, \upsilon, \mu \rangle$, return the predefined value μ. Otherwise, pick $\upsilon \in_R \mathbb{Z}_p^*$ and compute $\mu = (g^{a^n})^{\upsilon}$. Next, add tuple $\langle W, C_0, C_1, C_2, C_3, \upsilon, \mu \rangle$ to $L_{\mathcal{H}_3}$ and respond with $\mathcal{H}_3(W, C_0, C_1, C_2, C_3) = \mu$.

-$\mathcal{H}_4(\delta)$: $\mathcal{C}$ maintains a list $L_{\mathcal{H}_4}$ that contains tuples of the form $\langle \delta \in \{0,1\}^{l_m}, \tau \in \mathbb{Z}_p^* \rangle$. If the query already appears in $L_{\mathcal{H}_4}$ in a tuple $\langle \delta, \tau \rangle$, return the predefined value τ. Else if $\delta = \delta^*$, output "*failure*" and abort. Otherwise, pick $\tau \in_R \mathbb{Z}_p^*$, add tuple $\langle \delta, \tau \rangle$ to $L_{\mathcal{H}_4}$ and respond with $\mathcal{H}_4(\delta) = \tau$.

-$\mathcal{H}_5(W', rk_4, rk_5, rk_6)$: $\mathcal{C}$ maintains a list $L_{\mathcal{H}_5}$ that contains tuples of the form $\langle W', rk_4 \in \{0,1\}^{l_m+l_\sigma}, rk_5 \in \mathbb{G}_0, rk_6 \in \mathbb{G}_0, \vartheta \in \mathbb{Z}_p^*, \eta \in \mathbb{G}_0\rangle$. If the query already appears in $L_{\mathcal{H}_5}$ in a tuple $\langle W', rk_4, rk_5, rk_6, \vartheta, \eta\rangle$, return the predefined value η. Otherwise, pick $\vartheta \in_R \mathbb{Z}_p^*$, compute $\eta = (g^{a^n})^\vartheta$, add tuple $\langle W', rk_4, rk_5, rk_6, \vartheta, \eta\rangle$ to $L_{\mathcal{H}_5}$ and return η.

-**Private Key Extraction** $\mathcal{O}_{SK}(M, \rho)$: When $\mathcal{A}$ queries for the private keys corresponding to an access structure (M, ρ) such that $W^* \not\models (M, \rho)$, $\mathcal{C}$ searches list L_{RK} for a tuple $\langle (M', \rho'), (M, \rho), RK_{(M',\rho')\to(M,\rho)}\rangle$ such that $W^* \models (M', \rho')$. If such a tuple exists, $\mathcal{C}$ outputs *"failure"* and aborts. Otherwise, $\mathcal{C}$ checks if the given query already exists in list L_{SK} in a tuple $\langle (M, \rho), SK_{(M,\rho)}\rangle$. If not present, for each row M_i of the matrix M where $att_{\rho(i)} \in W^*$, there exists a vector $w = (-1, w_2, \cdots, w_k) \in \mathbb{Z}_p^k$ such that $M_i \cdot w = 0$, as per Lemma 1. The challenger $\mathcal{C}$ picks $z_2', z_3', \cdots, z_k' \in_R \mathbb{Z}_p^*$ and sets $v' = (0, z_2', z_3', \cdots, z_k') \in \mathbb{Z}_p^k$. It implicitly sets $v = -(\alpha' + a^{n+1})w + v'$. Next, it generates the private keys corresponding to each row M_i as per the following two cases.

- If $att_{\rho(i)} \in W^*$: From Lemma 1, note that $M_i \cdot w = 0$ holds good. The share $\lambda_{\rho(i)}$ is computed as below:

$$\begin{aligned}\lambda_{\rho(i)} &= M_i \cdot v\\ &= -(\alpha' + a^{n+1})(M_i \cdot w) + M_i \cdot v'\\ &= M_i \cdot v'.\end{aligned}$$

  The challenger $\mathcal{C}$ picks $r_i \in_R \mathbb{Z}_p^*$ and computes the private keys as below:

$$\begin{aligned}&K_i = g^{\lambda_{\rho(i)}}(h_0 h_{\rho(i)})^{r_i}, K_i' = g^{r_i},\\ &K_i'' = \{K_{iy}'' : K_{iy}'' = h_y^{r_i}, \forall y \in [n]\backslash\{\rho(i)\}\}.\end{aligned}$$

- If $att_{\rho(i)} \notin W^*$: The challenger $\mathcal{C}$ picks $r_i' \in_R \mathbb{Z}_p^*$ and implicitly sets $r_i = a^{\rho(i)}(M_i \cdot w) + r_i'$ and computes the private keys as below:
  - $K_i = g^{(M_i\cdot v')-\alpha'(M_i\cdot w)} \cdot (h_0 h_{\rho(i)})^{r_i'} \cdot (g^{a^{\rho(i)}})^{t_0\cdot(M_i\cdot w)}$ $\cdot (\prod_{att_y\in W^*} g^{a^{\rho(i)}\cdot t_y} \cdot g^{a^{n+1-y-\rho(i)}})^{(M_i\cdot w)} \cdot (g^{a^{\rho(i)}})^{t_{\rho(i)}\cdot(M_i\cdot w)}$
  - $K_i' = \left(g^{a^{\rho(i)}}\right)^{(M_i\cdot w)} \cdot g^{r_i'}$
  - $K_i'' = \{K_{iy}'' : K_{iy}'' = \left(\left(g^{a^{\rho(i)}}\right)^{t_y} \cdot g^{a^{n+\rho(i)+1-y}}\right)^{(M_i\cdot w)} \cdot h_y^{r_i'}, \forall y \in [n]\backslash\{\rho(i)\}\}$

  Observe that the private keys computed are identically distributed as the keys generated by the KeyGen algorithm in the construction. In fact, we have:

$$\begin{aligned}K_i = g^{(M_i\cdot v')-\alpha'(M_i\cdot w)} \cdot (h_0 h_{\rho(i)})^{r_i'} \cdot (g^{a^{\rho(i)}})^{t_0\cdot(M_i\cdot w)}\\ \cdot \Big(\prod_{att_y\in W^*} g^{a^{\rho(i)}\cdot t_y} \cdot g^{a^{n+1-y-\rho(i)}}\Big)^{(M_i\cdot w)} \cdot \left(g^{a^{\rho(i)}}\right)^{t_{\rho(i)}\cdot(M_i\cdot w)}\end{aligned}$$

$$\begin{aligned}= g^{(M_i\cdot v')-\alpha'(M_i\cdot w)} \cdot (h_0 h_{\rho(i)})^{r_i'} \cdot g^{-a^{n+1}(M_i\cdot w)}\\ \cdot (h_0)^{a^{\rho(i)}(M_i\cdot w)} \cdot (g^{a^{\rho(i)}})^{t_{\rho(i)}(M_i\cdot w)} \cdot g^{a^{n+1}(M_i\cdot w)}\end{aligned}$$

$$= g^{M_i \cdot v' - \alpha'(M_i \cdot w)} \cdot g^{-a^{n+1}(M_i \cdot w)} \cdot (h_0 h_{\rho(i)})^{r'_i}$$
$$\cdot (h_0)^{a^{\rho(i)}(M_i \cdot w)} \cdot (g^{t_{\rho(i)}} \cdot g^{a^{n+1-\rho(i)}})^{a^{\rho(i)}(M_i \cdot w)}$$

$$= g^{M_i \cdot v' - \alpha'(M_i \cdot w)} \cdot g^{-a^{n+1}(M_i \cdot w)} \cdot (h_0 h_{\rho(i)})^{a^{\rho(i)}(M_i \cdot w) + r'_i}$$

$$= g^{\lambda_{\rho(i)}} \cdot (h_0 h_{\rho(i)})^{r_i}.$$

$$\begin{aligned} K'_i &= \left(g^{a^{\rho(i)}}\right)^{(M_i \cdot w)} \cdot g^{r'_i} \\ &= g^{a^{\rho(i)}(M_i \cdot w) + r'_i} \\ &= g^{r_i}. \end{aligned}$$

$$\begin{aligned} K''_i &= \left(\left(g^{a^{\rho(i)}}\right)^{t_y} \cdot g^{a^{n+\rho(i)+1-y}}\right)^{(M_i \cdot w)} \cdot h_y^{r'_i} \\ &= (g^{t_y} \cdot g^{a^{n+1-y}})^{a^{\rho(i)} \cdot (M_i \cdot w)} \cdot h_y^{r'_i} \\ &= h_y^{a^{\rho(i)}(M_i w) + r'_i}. \\ &= h_y^{r_i}. \end{aligned}$$

$\mathcal{C}$ returns the private keys $\langle \forall i \in [l] : K_i, K'_i, K''_i \rangle$ to $\mathcal{A}$. It is straightforward to note that the challenger $\mathcal{C}$ performs atmost $(n+2)l$ exponentiation operations for every invocation to the private key extraction oracle.

Re-encryption Key Generation$(\mathcal{O}_{RK}((M, \rho), (M', \rho')))$: On input of two access structures (M, ρ) and (M', ρ'), the challenger $\mathcal{C}$ checks if the given query already appears in list L_{RK} in a tuple $\langle (M, \rho), (M', \rho'),$
$RK_{(M,\rho) \to (M',\rho')} \rangle$. If not present, $\mathcal{C}$ generates the re-encryption key as below:

- If $W^* \models (M, \rho)$ and $\langle M', \rho', SK_{(M',\rho')} \rangle \in L_{SK}$ such that the private keys of access structure (M', ρ') has already been queried upon, output *"failure"* and abort.
- Otherwise, check if the private key $SK_{(M,\rho)}$ already appears in L_{SK}. If not present, invoke $\mathcal{O}_{SK}(M, \rho)$ to generate the private keys corresponding to (M, ρ).
- Generate re-encryption keys as per ReKeyGen protocol.
- Update list L_{RK} with the tuple $\langle (M, \rho), (M', \rho'), RK_{(M,\rho) \to (M',\rho')} \rangle$ and return $RK_{(M,\rho) \to (M',\rho')}$.

It is straightforward to note that the challenger $\mathcal{C}$ performs atmost $(n+6)l$ exponentiation operations for every invocation to the re-encryption key generation oracle.

-Re-Encryption$(\mathcal{O}_{RE}(C, (M, \rho), (M', \rho')))$: On input of a first level ciphertext C encrypted under an attribute set W satisfying (M, ρ), re-encrypt using the following steps:

- Check ciphertext validity using Equations (3), (4), (5).
- If they hold, check list $L_{\mathcal{H}_3}$ for tuple $\langle W, C_0, C_1, C_2, C_3, \upsilon, \mu \rangle$ such that $\mu = (g^{a^n})^{\upsilon}$ for some known $\upsilon \in \mathbb{Z}_p^*$.

- Compute:

$$X = \hat{e}((C_4)^{1/\upsilon}, g^a) \cdot \hat{e}(C_1, g^{\alpha'})$$
$$= \hat{e}((g^{a^n})^s, g^a) \cdot \hat{e}(g^s, g^{\alpha'})$$
$$= e(g,g)^{s \cdot \alpha}$$
$$= (Y)^s.$$

- Pick $\delta \in \{0,1\}^{l_m}, \gamma \in \{0,1\}^{l_\sigma}, \theta \in \mathbb{Z}_p^*$.
- Compute $D_0 = X^{\mathcal{H}_4(\delta)}$.
- Set $D_1 = C_0$, $D_2 = C_1$.
- Compute $s' = \mathcal{H}_1(\delta||\gamma)$.
- Compute $D_3 = (\delta||\gamma) \oplus \mathcal{H}_2(Y^{s'})$, $D_4 = g^{s'}$.
- Pick any attribute set $W' \models (M', \rho')$.
- Compute $D_5 = \left(h_0 \prod_{att_y \in W'} h_y\right)^{s'}$.
- Compute $D_6 = \left(\mathcal{H}_5(W', D_3, D_4, D_5)\right)^{s'}$.
- Return ciphertext $D = (D_0, D_1, D_2, D_3, D_4, D_5, D_6, W')$.

It is straightforward to note that the challenger $\mathcal{C}$ performs atmost 7 exponentiation and 8 bilinear pairing operations for every invocation to the re-encryption oracle.

-Decryption$(\mathcal{O}_{Dec}(C, (M, \rho)))$: On input of a first level ciphertext C encrypted under an attribute set $W \models (M, \rho)$, decrypt as shown below:
- Check ciphertext validity using Equations (3), (4) and (5).
- If they hold, check list $L_{\mathcal{H}_3}$ for tuple $\langle W, C_0, C_1, C_2, C_3, \upsilon, \mu \rangle$ such that $\mu = (g^{a^n})^\upsilon$ for some known $\upsilon \in \mathbb{Z}_p^*$.
- Compute:

$$X = \hat{e}((C_4)^{1/\upsilon}, g^a) \cdot \hat{e}(C_1, g^{\alpha'})$$
$$= (Y)^s.$$

- Search list $L_{\mathcal{H}_2}$ for a tuple $\langle X, \nu \rangle$ and compute $(m||\sigma) = C_0 \oplus \nu$.
- Search list $L_{\mathcal{H}_1}$ for a tuple $\langle m, \sigma, s \rangle$. If $C_1 \stackrel{?}{=} g^{\mathcal{H}_1(m||\sigma)}$, return the plaintext m, otherwise return $\perp$.

It is straightforward to note that the challenger $\mathcal{C}$ performs atmost 3 exponentiation and 8 bilinear pairing operations for every invocation to the decryption oracle.

-Re-Decryption$(\mathcal{O}_{RD}(D, (M', \rho')))$: On input of a second level ciphertext D encrypted under attribute set $W' \models (M', \rho')$, decrypt as shown below:
- Check ciphertext validity using Equations (10) and (11).
- If they hold, check list $L_{\mathcal{H}_5}$ for tuple $\langle W', D_3, D_4, D_5, \vartheta, \eta \rangle$ such that $\eta = (g^{a^n})^\vartheta$ for some known $\vartheta \in \mathbb{Z}_p^*$.
- Compute:

$$X' = \hat{e}((D_6)^{1/\vartheta}, g^a) \cdot \hat{e}(D_2, g^{\alpha'})$$
$$= \hat{e}((g^{a^n})^{s'}, g^a) \cdot \hat{e}(g^{s'}, g^{\alpha'})$$
$$= \hat{e}(g,g)^{s'(a^{n+1} + \alpha')}$$
$$= \hat{e}(g,g)^{s'\alpha}$$
$$= (Y)^{s'}.$$

  - Search list $L_{\mathcal{H}_2}$ for a tuple $\langle X', \nu' \rangle$ and compute $(\delta || \gamma) = D_3 \oplus \nu'$.
  - If $D_4 \stackrel{?}{=} g^{\mathcal{H}_1(\delta||\gamma)}$, compute the plaintext $(m||\sigma) = D_1 \oplus \mathcal{H}_2(D_0^{1/\mathcal{H}_4(\delta)})$.
  - If $D_2 \stackrel{?}{=} g^{\mathcal{H}_1(m||\sigma)}$, return m, otherwise return $\perp$.

  It is straightforward to note that the challenger $\mathcal{C}$ performs atmost 5 exponentiation and 6 bilinear pairing operations for every invocation to the re-decryption oracle.

– **Challenge**: $\mathcal{A}$ outputs two messages (m_0, m_1) to $\mathcal{C}$. The challenger $\mathcal{C}$ picks $\psi \in \{0,1\}$ at random and encrypts m_ψ under attribute set W^* as below:
  - Pick $\sigma^* \in \{0,1\}^{l_\sigma}$.
  - Implicitly define $\mathcal{H}_1(m_\psi || \sigma^*) = b$.
  - Compute $C_0^* = (m_\psi || \sigma^*) \oplus \mathcal{H}_2(Z \cdot \hat{e}(g^b, g^{\alpha'}))$.
  - Compute $C_1^* = g^b$, $C_2^* = (g^b)^\beta$, $C_3^* = (g^b)^{t_0}$.
  - Set $\mathcal{H}_3(W^*, C_0^*, C_1^*, C_2^*, C_3^*) = g^k$, where $k \in_R \mathbb{Z}_p^*$.
  - Compute $C_4^* = (g^b)^k$.
  - Return the ciphertext $C^* = (C_0^*, C_1^*, C_2^*, C_3^*, C_4^*, W^*)$.

  Observe that the first-level ciphertext computed is identically distributed as the ciphertext generated by the Encrypt algorithm in the construction. In fact, we have:

$$\begin{aligned} C_0^* &= (m_\psi || \sigma^*) \oplus \mathcal{H}_2(Z \cdot \hat{e}(g^b, g^{\alpha'})) \\ &= (m_\psi || \sigma^*) \oplus \mathcal{H}_2(\hat{e}(g^b, g^{a^{n+1}}) \cdot \hat{e}(g^b, g^{\alpha'})) \\ &= (m_\psi || \sigma^*) \oplus \mathcal{H}_2(\hat{e}(g^b, g^{\alpha})) \\ &= (m_\psi || \sigma^*) \oplus \mathcal{H}_2(Y^s). \end{aligned}$$

$$\begin{aligned} C_3^* &= (g^b)^{t_o} \\ &= (g^{t_0})^b \\ &= (g^{t_0} (\prod_{att_y \in W^*} h_y)^{-1} \cdot (\prod_{att_y \in W^*} h_y))^b \\ &= (h_0 \prod_{att_y \in W^*} h_y)^s. \end{aligned}$$

Phase-2: $\mathcal{A}$ continues to query the oracles maintained by $\mathcal{C}$ subject to the constraints stated in the security model.

Guess: $\mathcal{A}$ eventually produces its guess $\psi' \in \{0,1\}$. If $\psi' = \psi$, $\mathcal{A}$ wins the game and $\mathcal{C}$ decides $\hat{e}(g,g)^{abc} = Z$, else Z is random.

Probability Analysis: We first analyse the simulation of the random oracles. The simulation of the random oracles takes place perfectly unless the following events take place:

– $E_{\mathcal{H}_1}$: Event that (m_ψ, σ^*) was queried to $\mathcal{H}_1$ function.
– $E_{\mathcal{H}_2}$: Event that $(Z \cdot \hat{e}(g^b, g^{\alpha'}))$ was queried to $\mathcal{H}_2$ function.

Next we analyse the simulation of the re-encryption oracle. The responses to $\mathcal{A}$'s re-encryption queries are perfect, unless $\mathcal{A}$ submits a valid second-level ciphertext without having queried the hash function $\mathcal{H}_3$ (we denote this event by $RErr$). Since $\mathcal{H}_3$ acts as a random oracle and $\mathcal{A}$ issues at most q_{ReEnc} re-encryption queries, we have $\Pr[RErr] = \frac{q_{ReEnc}}{p}$.

The simulation of the decryption oracle is perfect unless valid ciphertexts are rejected, which occurs when $\mathcal{A}$ queries the decryption oracle without having queried $\mathcal{H}_1$ and $\mathcal{H}_2$. Let E_{valid} denote the event that the ciphertext is a valid ciphertext. We have:

$$\begin{aligned} Pr[E_{valid}|\neg E_{\mathcal{H}_2}] &= Pr[E_{valid} \wedge E_{\mathcal{H}_1}|\neg E_{\mathcal{H}_2}] + Pr[E_{valid} \wedge \neg E_{\mathcal{H}_1}|\neg E_{\mathcal{H}_2}] \\ &\leq Pr[E_{valid} \wedge E_{\mathcal{H}_1}|\neg E_{\mathcal{H}_2}] + Pr[E_{valid}|\neg E_{\mathcal{H}_1} \wedge \neg E_{\mathcal{H}_2}] \\ &\leq \frac{Pr[E_{\mathcal{H}_1}]}{Pr[\neg E_{\mathcal{H}_2}]} + \frac{1}{p} \\ &\leq \frac{q_{H_1}}{(2^{l_m + l_\sigma})} + \frac{1}{p}. \end{aligned}$$

Let us denote E_{DEr} denote that the event $(E_{valid}|\neg E_{H_2})$ occurs during the entire simulation, and we obtain:

$$Pr[E_{DEr}] \leq q_{Dec}\Big(\frac{q_{H_1}}{(2^{l_m + l_\sigma})} + \frac{1}{p}\Big).$$

Let E_{er} denote the event $(E_{\mathcal{H}_2} \vee (E_{\mathcal{H}_1}|\neg E_{\mathcal{H}_2}) \vee E_{RErr} \vee E_{DEr})$. If event E_{er} does not happen, the adversary $\mathcal{A}$ does not gain any advantage in guessing ψ due to the randomness in the output of the random oracle $\mathcal{H}_2$. Therefore, $Pr[\psi' = \psi|\neg E_{er}] = \frac{1}{2}$. Note that:

$$\begin{aligned} Pr[\psi' = \psi] &= Pr[\psi' = \psi|\neg E_{er}]Pr[\neg E_{er}] + Pr[\psi' = \psi|E_{er}]Pr[E_{er}] \\ &\leq \frac{1}{2}Pr[\neg E_{er}] + Pr[E_{er}] = \frac{1}{2} + \frac{1}{2}Pr[E_{er}]. \end{aligned}$$

$$\text{Also, } Pr[\psi' = \psi] \geq Pr[\psi' = \psi|\neg E_{er}]Pr[\neg E_{er}] \geq \frac{1}{2} - \frac{1}{2}Pr[E_{er}].$$

From the definition of the advantage of CCA adversary, we have:

$$\begin{aligned} \epsilon &= |Pr[\psi' = \psi] - \frac{1}{2}| \\ &\leq \frac{1}{2}Pr[E_{er}] = \frac{1}{2}Pr[(E_{\mathcal{H}_2} \vee (E_{\mathcal{H}_1}|\neg E_{\mathcal{H}_2}) \vee E_{RErr} \vee E_{DEr})]. \end{aligned}$$

Therefore, we obtain the following bound on $Pr[E_{\mathcal{H}_2}]$ as:

$$\begin{aligned} Pr[E_{\mathcal{H}_2}] &\geq 2\epsilon - \Pr[E_{\mathcal{H}_1}|\neg E_{\mathcal{H}_2}] + \Pr[E_{RErr}] + \Pr[E_{DEr}] \\ &\geq 2\epsilon - \frac{q_{\mathcal{H}_1}}{2^{l_m + l_\sigma}} - \frac{q_{ReEnc}}{p} - q_{Dec}\Big(\frac{q_{H_1}}{(2^{l_m + l_\sigma})} + \frac{1}{p}\Big) \end{aligned}$$

Note that, if event $E_{\mathcal{H}_2}$ occurs, then the challenger $\mathcal{C}$ solves the n-$DBDHE$ instance with advantage:

$$\begin{aligned} \epsilon' &\geq \frac{1}{q_{\mathcal{H}_2}}Pr[E_{\mathcal{H}_2}] \\ &\geq \frac{1}{q_{\mathcal{H}_2}}\Big(2\epsilon - \frac{q_{\mathcal{H}_1}}{2^{l_m + l_\sigma}} - \frac{q_{ReEnc}}{p} - q_{Dec}\Big(\frac{q_{\mathcal{H}_1}}{(2^{l_m + l_\sigma})} + \frac{1}{p}\Big)\Big) \end{aligned}$$

The reduction involves asking $q_{\mathcal{H}_3}, q_{\mathcal{H}_5}$, q_{SK}, q_{RK}, q_{ReEnc}, q_{Dec} and q_{ReDec} queries to the $\mathcal{H}_3$ and $\mathcal{H}_5$ hash functions, private-key extraction, re-encryption key generation, re-encryption, decryption and re-decryption oracles, and the number of exponentiations done under these queries are $1, 1, (n+2)l, (n+6)l, 7, 3$ and 5 respectively. Additionally, the re-encryption, decryption and re-decryption oracles incur $8, 8$ and 6 bilinear pairing operations respectively. Hence, the total number of operations performed are $(q_{\mathcal{H}_3} + q_{\mathcal{H}_5} + (n+2)lq_{SK} + (n+6)lq_{RK} + 7q_{ReEnc} + 3q_{Dec} + 5q_{RD})t_e + (8q_{ReEnc} + 8q_{Dec} + 6q_{RD})t_{bp}$ where t_e is the time taken for one exponentiation operation and t_{bp} is the time taken for one bilinear pairing operation. Therefore, we can bound the running time of the challenger by:

$$t' \le t + (q_{\mathcal{H}_3} + q_{\mathcal{H}_5} + (n+2)lq_{SK} + (n+6)lq_{RK} + 7q_{ReEnc} + 3q_{Dec} + 5q_{RD})t_e \\ + (8q_{ReEnc} + 8q_{Dec} + 6q_{RD})t_{bp}.$$

This completes the proof of the theorem.

Second Level Ciphertext Security :

Theorem 2. *If a (t, ϵ)IND-PRE-CCA adversary $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the IND-PRE-CCA security of the given KP-ABPRE scheme for second level ciphertext, with access to the random oracles $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4, \mathcal{H}_5$, then there exists an algorithm $\mathcal{C}$ that solves the n-DBDHE problem with advantage ϵ' within time t' where,*

$$\epsilon' \ge \frac{1}{q_{\mathcal{H}_2}} \Bigg(2\epsilon \cdot \Big(1 - \frac{q_{\mathcal{H}_4}}{2^{l_m}} \Big) - \frac{q_{\mathcal{H}_1}}{2^{l_m + l_\sigma}} - q_{Dec} \Big(\frac{q_{\mathcal{H}_1}}{(2^{l_m + l_\sigma})} + \frac{1}{p} \Big) \Bigg),$$

$$t' \le t + (q_{\mathcal{H}_3} + q_{\mathcal{H}_5} + (n+2)lq_{SK} + (n+6)lq_{RK} + 3q_{Dec} + 5q_{RD})t_e + (8q_{Dec} + 6q_{RD})t_{bp}.$$

Proof. If an adversary $\mathcal{A}$ that asks atmost $q_{\mathcal{H}_i}$ random oracle queries to $\mathcal{H}_i$ where $i \in \{1, 2, \cdots, 5\}$ breaks the IND-PRE-CCA security for the second level ciphertexts of the KP-ABPRE scheme, we show that we can construct a PPT algorithm $\mathcal{C}$ that can break the n-DBDHE assumption with non-negligible advantage. The algorithm $\mathcal{C}$ accepts as input the n-DBDHE challenge $\langle (g, g^b, g^a, g^{a^2}, \cdots, g^{a^n}, g^{a^{n+2}}, \cdots, g^{a^{2n}}) \in \mathbb{G}, T \in \mathbb{G}_1 \rangle$ and plays the role of a challenger in the following CCA-game with the adversary $\mathcal{A}$.

- **Initialization:** The adversary $\mathcal{A}$ shares the target access structure (M^*, ρ^*) with the challenger $\mathcal{C}$ on which it wishes to be challenged and the challenger $\mathcal{C}$ picks any $W^* \models (M^*, \rho^*)$ as the target attribute set. $\mathcal{C}$ picks $\alpha' \in_R \mathbb{Z}_p^*$, and implicitly sets *msk* $\alpha = \alpha' + a^{n+1}$. It computes $Y = e(g,g)^{\alpha'} \cdot e(g^a, g^{a^n})$ as shown in the first-level ciphertext security game. Let $g_1 = g^\beta$, where $\beta \in_R \mathbb{Z}_p^*$ is known to the challenger $\mathcal{C}$. For all attributes $att_y \in U$, $\mathcal{C}$ picks $t_y \in \mathbb{Z}_p^*$. It compute $h_y = g^{a^{n+1-y}} \cdot g^{t_y}$. It picks $t_0 \in_R \mathbb{Z}_p^*$ and computes $h_0 = (\prod_{att_y \in W^*} h_y)^{-1} \cdot g^{t_0}$. It picks $\delta^* \in \{0,1\}^{l_m}$, used in re-encryption key generation oracle. It maintains two lists L_{SK} and L_{RK} to store the list of private keys and the re-encryption keys generated by $\mathcal{C}$ and contain tuples of the form :
  - $L_{SK} : \langle (M, \rho), SK_{(M,\rho)} \rangle$.
  - $L_{RK} : \langle (M, \rho)(M', \rho'), RK_{(M,\rho) \to (M',\rho')} \rangle$.

Both the lists are initially empty. $\mathcal{C}$ returns the public parameters $params : \langle p, g, g_1, \hat{e}, Y, h_0, h_1, \cdots, h_n, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4, \mathcal{H}_5 \rangle$.

– **Phase 1 (Oracle Queries):** $\mathcal{C}$ responds to the oracle queries in the similar manner as shown in the first-level ciphertext security game.

  -Private Key Extraction $\mathcal{O}_{SK}(M, \rho)$: When the adversary $\mathcal{A}$ queries for the private keys corresponding to an access structure (M, ρ) such that $W^* \not\models (M, \rho)$, $\mathcal{C}$ checks if the given query already exists in list L_{SK} in a tuple $\langle (M, \rho), SK_{(M,\rho)} \rangle$. If not present, for each row M_i of the matrix M where $att_{\rho(i)} \in W^*$, there exists a vector $w = (-1, w_2, \cdots, w_k) \in \mathbb{Z}_p^k$ such that $M_i \cdot w = 0$, as per Lemma 1. The challenger $\mathcal{C}$ picks $z'_2, z'_3, \cdots, z'_k \in_R \mathbb{Z}_p^*$ and sets $v' = (0, z'_2, z'_3, \cdots, z'_k) \in \mathbb{Z}_p^k$. It implicitly sets $v = -(\alpha' + a^{n+1})w + v'$. Next, it generates the private keys corresponding to each row M_i as per the following two cases.

  - If $att_{\rho(i)} \in W^*$: From Lemma 1, note that $M_i \cdot w = 0$ holds good. The share $\lambda_{\rho(i)}$ is computed as $\lambda_{\rho(i)} = M_i \cdot v = M_i \cdot v'$, as shown in the first-level ciphertext security game. The challenger $\mathcal{C}$ picks $r_i \in_R \mathbb{Z}_p^*$ and computes the private keys as below:

$$K_i = g^{\lambda_{\rho(i)}}(h_0 h_{\rho(i)})^{r_i}, K'_i = g^{r_i},\ K''_i = \{K''_{iy} : K''_{iy} = h_y^{r_i}, \forall y \in [n] \backslash \{\rho(i)\}\}.$$

  - If $att_{\rho(i)} \notin W^*$: The challenger $\mathcal{C}$ picks $r'_i \in_R \mathbb{Z}_p^*$ and implicitly sets $r_i = a^{\rho(i)}(M_i \cdot w) + r'_i$ and computes the private keys as below:
    - $K_i = g^{(M_i \cdot v') - \alpha'(M_i \cdot w)} \cdot (h_0 h_{\rho(i)})^{r'_i}$
      $\cdot (g^{a^{\rho(i)}})^{t_0 \cdot (M_i \cdot w)} \cdot (\prod_{att_y \in W^*} g^{a^{\rho(i)} \cdot t_y} \cdot g^{a^{n+1-y-\rho(i)}})^{(M_i \cdot w)} \cdot (g^{a^{\rho(i)}})^{t_{\rho(i)} \cdot (M_i \cdot w)}$.
    - $K'_i = \left(g^{a^{\rho(i)}}\right)^{(M_i \cdot w)} \cdot g^{r'_i}$.
    - $K''_i = \{K''_{iy} : K''_{iy} = \left(\left(g^{a^{\rho(i)}}\right)^{t_y} \cdot g^{a^{n+\rho(i)+1-y}}\right)^{(M_i \cdot w)} \cdot h_y^{r'_i}, \forall y \in [n] \backslash \{\rho(i)\}\}$.

  Note that the private keys computed are identically distributed as the keys generated by the KeyGen algorithm in the construction, as shown in the proof of DSK security. $\mathcal{C}$ returns the private keys $\langle \forall i \in [l] : K_i, K'_i, K''_i \rangle$ to the adversary $\mathcal{A}$.

  Re-encryption Key Generation$(\mathcal{O}_{RK}((M, \rho), (M', \rho')))$: On input of two access structures (M, ρ) and (M', ρ'), the challenger checks if the given query already appears in list L_{RK} in a tuple $\langle (M, \rho), (M', \rho'), RK_{(M,\rho) \to (M',\rho')} \rangle$. If not present, $\mathcal{C}$ generates the re-encryption key as per the following two cases:

  - If $W^* \not\models (M, \rho)$:
    * Check if the private key $SK_{(M,\rho)}$ already appears in L_{SK}. If not present, invoke $\mathcal{O}_{SK}(M, \rho)$ to generate the private keys corresponding to (M, ρ).
    * Generate the re-encryption keys as per the ReKeyGen protocol.
    * Return $RK_{(M,\rho) \to (M',\rho')}$.
  - If $W^* \models (M, \rho)$:
    * Let the matrix M be of size $l \times k$. Pick $z'_2, z'_3, \cdots, z'_k \in \mathbb{Z}_p^*$ and set $v' = (1, z'_2, z'_3 \cdots, z'_k)$.

* Implicitly set $v = (\alpha' + a^{n+1})v'$, such that $v = ((\alpha' + a^{n+1}, (\alpha' + a^{n+1})z'_2, \cdots, (\alpha' + a^{n+1})z'_k)$ is a vector of length k. Compute the share $\lambda_{\rho(i)}$ as below:

$$\begin{aligned}\lambda_{\rho(i)} &= M_i \cdot v \\ &= (\alpha' + a^{n+1})(M_i \cdot v').\end{aligned}$$

* Pick $\gamma \in \{0,1\}^{l_\sigma}$ and $\theta' \in \mathbb{Z}_p^*$.
* For each row M_i of matrix M, pick $r'_i \in \mathbb{Z}_p^*$. Implicitly, define $r_i = a^{n+1} \cdot r'_i$ and $\mathcal{H}_4(\delta^*) = a$. Set $\theta = -(a^{n+1} \cdot \mathcal{H}_4(\delta^*)) \cdot r'_i + \theta'$. Compute the re-encryption key components as below:

$$\begin{aligned} rk_{1i} &= (g^{a^{n+2}})^{(M_i \cdot v')} \cdot (g^a)^{\alpha'(M_i \cdot v')} \cdot (h_0 h_i)^{\theta'} \\ rk_{2i} &= g^{\theta'} \\ rk_{3i} &= \{rk_{3iy} : rk_{3iy} = (g^{n+1-y})^{\theta'}, \forall y \in [n] \backslash \{\rho(i)\}\} \end{aligned}$$

Observe that the re-encryption key components computed are identically distributed as the keys generated by the RekeyGen algorithm in the construction. In fact, we have:

$$\begin{aligned} rk_{1i} &= (g^{a^{n+2}})^{(M_i \cdot v')} \cdot (g^a)^{\alpha'(M_i \cdot v')} \cdot (h_0 h_i)^{\theta'} \\ &= g^{\lambda_{\rho(i)} \mathcal{H}_4(\delta^*)} \cdot (h_0 h_i)^{\mathcal{H}_4(\delta^*) r_i + \theta} \\ &= K_i^{\mathcal{H}_4(\delta^*)} (h_0 h_i)^{\theta}. \end{aligned}$$

$$\begin{aligned} rk_{2i} &= g^{\theta'} \\ &= g^{(a^{n+1} \cdot r'_i \cdot \mathcal{H}_4(\delta^*) + (-a^{n+1} \cdot r'_i \cdot \mathcal{H}_4(\delta^*)) + \theta')} \\ &= g^{r_i \mathcal{H}_4(\delta^*) + \theta} \\ &= (K'_i)^{\mathcal{H}_4(\delta^*)}. \end{aligned}$$

$$\begin{aligned} rk_{3i} &= \{rk_{3iy} : rk_{3iy} = (g^{n+1-y})^{\theta'}, \forall y \in [n] \backslash \{\rho(i)\}\} \\ &= h_{iy}^{(a^{n+1} \cdot r'_i \cdot \mathcal{H}_4(\delta^*) + (-a^{n+1} \cdot r'_i \cdot \mathcal{H}_4(\delta^*)) + \theta')} \\ &= h_{iy}^{(r_i \mathcal{H}_4(\delta^*) + \theta)} \\ &= (K''_{iy})^{\mathcal{H}_4(\delta^*)}. \end{aligned}$$

* Compute $s' = \mathcal{H}_1(\delta^* || \gamma)$.
* Compute $rk_4 = (\delta^* || \gamma) \oplus \mathcal{H}_2(Y^{s'})$.
* Compute $rk_5 = g^{s'}$.
* Pick an attribute set $W' \models (M', \rho')$.
* Compute $rk_6 = \left(h_0 \prod_{att_y \in W'} h_y\right)^{s'}$.
* Compute $rk_7 = \left(\mathcal{H}_5(W', rk_4, rk_5, rk_6)\right)^{s'}$.

* Return the re-encryption key $RK_{(M,\rho)\to(M',\rho')} = (\{\forall i \in [l] : rk_{1i}, rk_{2i}, rk_{3i}\}, rk_4, rk_5, rk_6, rk_7, W')$ and update list L_{RK}.

-Decryption$(\mathcal{O}_{Dec}(C, (M, \rho)))$: $\mathcal{C}$ responds to the decryption query for first-level ciphertexts in the same manner as shown in the first-level ciphertext security game.

-Re-Decryption$(\mathcal{O}_{RD}(D, (M', \rho')))$: $\mathcal{C}$ responds to the decryption query for second-level ciphertexts in the same manner as shown in the first-level ciphertext security game.

– **Challenge**: When $\mathcal{A}$ decides that Phase 1 is over, it outputs two messages (m_0, m_1) to $\mathcal{C}$. The challenger $\mathcal{C}$ picks $\psi \in \{0, 1\}$ at random and re-encrypts m_ψ under an access structure satisfying attribute set W^* as below:
  - Pick $\theta^* \in \mathbb{Z}_p^*$, $\delta^* \in \{0,1\}^{l_m}$ and $\sigma^*, \gamma^* \in \{0,1\}^{l_\sigma}$.
  - Compute $s = \mathcal{H}_1(m_\psi||\sigma^*)$.
  - Compute $D_0^* = (\hat{e}(g,g)^{\alpha'} \cdot e(g^a, g^{a^n}))^{s\mathcal{H}_4(\delta^*)} = Y^{s \cdot \mathcal{H}_4(\delta^*)}$.
  - Compute $D_1^* = (m_\psi||\sigma^*) \oplus \mathcal{H}_2(Y^s)$.
  - Compute $D_2^* = g^s$.
  - Implicitly define $\mathcal{H}_1(\delta^*||\gamma^*) = b$.
  - Compute $D_3^* = (\delta^*||\gamma^*) \oplus \mathcal{H}_2(Z \cdot \hat{e}(g^b, g^{\alpha'}))$.
  - Compute $D_4^* = g^b$.
  - Compute $D_5^* = (g^b)^{t_0}$.
  - Set $\mathcal{H}_5(W^*, D_3^*, D_4^*, D_5^*) = g^k$, where $k \in_R \mathbb{Z}_p^*$.
  - Compute $D_6^* = (g^b)^k$.
  - Return the second level ciphertext $D^* = (D_0^*, D_1^*, D_2^*, D_3^*, D_4^*, D_5^*, D_6^*, W^*)$.

  Observe that the second-level ciphertext computed is identically distributed as the ciphertext generated by the ReEncrypt algorithm in the construction. In fact, we have:

$$\begin{aligned}
D_3^* &= (\delta^*||\gamma^*) \oplus \mathcal{H}_2(Z \cdot \hat{e}(g^b, g^{\alpha'})) \\
&= (\delta^*||\gamma^*) \oplus \mathcal{H}_2(\hat{e}(g^b, g^{a^{n+1}}) \cdot \hat{e}(g^b, g^{\alpha'})) \\
&= (\delta^*||\gamma^*) \oplus \mathcal{H}_2(\hat{e}(g^b, g^{a^{n+1}+\alpha'})) \\
&= (\delta^*||\gamma^*) \oplus \mathcal{H}_2(\hat{e}(g^b, g^{\alpha})) \\
&= (\delta^*||\gamma^*) \oplus \mathcal{H}_2(Y^{s'}).
\end{aligned}$$

$$\begin{aligned}
D_5^* &= (g^b)^{t_o} \\
&= (g^{t_0})^b \\
&= (g^{t_0}(\prod_{att_y \in W^*} h_y)^{-1} \cdot (\prod_{att_y \in W^*} h_y))^b \\
&= (h_0 \prod_{att_y \in W^*} h_y)^{s'}.
\end{aligned}$$

Phase-2: The adversary $\mathcal{A}$ continues to query the oracles maintained by $\mathcal{C}$ subject to the constraints stated in the security model.

Guess: The adversary $\mathcal{A}$ eventually produces its guess $\psi' \in \{0,1\}$. If $\psi' = \psi$, $\mathcal{A}$ wins the game and $\mathcal{C}$ decides $\hat{e}(g,g)^{abc} = Z$, else Z is random.

Probability Analysis: We first analyse the simulation of the random oracles. The simulation of the random oracles takes place perfectly unless the following events take place:

- $E_{\mathcal{H}_1}$: Event that (m_ψ, σ^*) was queried to the $\mathcal{H}_1$ hash function.
- $E_{\mathcal{H}_2}$: Event that $(Z \cdot \hat{e}(g^b, g^{\alpha'}))$ was queried to $\mathcal{H}_2$ hash function.
- $E_{\mathcal{H}_4}$: Event that δ^* was queried to the $\mathcal{H}_4$ oracle.

We have $\Pr[E_{\mathcal{H}_4}] = \left(\frac{q_{\mathcal{H}_4}}{2^{l_m}}\right)$. Let *abort* denote the event that $\mathcal{C}$ aborts the simulation. Hence, $\Pr[\neg abort] = \Pr[\neg E_{\mathcal{H}_4}] = \left(1 - \frac{q_{\mathcal{H}_4}}{2^{l_m}}\right)$.

The simulation of the decryption oracle is perfect unless valid ciphertexts are rejected, which occurs when $\mathcal{A}$ queries the decryption oracle without querying $\mathcal{H}_1$ and $\mathcal{H}_2$. Let E_{valid} denote the event that the ciphertext is a valid ciphertext. Let $E_{\mathcal{H}_1}$ and $E_{\mathcal{H}_2}$ denote the events that $\mathcal{H}_1$ and $E_{\mathcal{H}_2}$ has been queried by $\mathcal{A}$. Let us denote E_{DEr} denote that the event $(E_{valid}|\neg E_{H_2})$ occurs during the entire simulation. As shown in the probability analysis of the first level ciphertext, we obtain:

$$Pr[E_{DEr}] \leq q_{Dec}\Big(\frac{q_{H_1}}{(2^{l_m+l_\sigma})} + \frac{1}{p}\Big).$$

Let E_{er} denote the event $(E_{\mathcal{H}_2} \vee (E_{\mathcal{H}_1}|\neg E_{\mathcal{H}_2}) \vee E_{RErr} \vee E_{DEr})|\neg abort$. If E_{er} does not occur, the adversary $\mathcal{A}$ gain no advantage in guessing ψ due to the randomness in the output of $\mathcal{H}_2$ oracle, i.e., $Pr[\psi' = \psi|\neg E_{er}] = \frac{1}{2}$. As shown in the probability analysis of the first level ciphertext, by the definition of the advantage of CCA adversary, we have the advantage:

$$\begin{aligned}
\epsilon &= |Pr[\psi' = \psi] - \frac{1}{2}| \\
&\leq \frac{1}{2}Pr[E_{er}] = \frac{1}{2}Pr[(E_{\mathcal{H}_2} \vee (E_{\mathcal{H}_1}|\neg E_{\mathcal{H}_2}) \vee E_{DEr})|\neg abort] \\
&\leq \frac{1}{2}((\Pr[E_{\mathcal{H}_2}] + \Pr[E_{\mathcal{H}_1}|\neg E_{\mathcal{H}_2}] + \Pr[E_{DEr}])/\Pr[\neg abort]).
\end{aligned}$$

Therefore, we obtain the following bound on $Pr[E_{\mathcal{H}_2}]$ as:

$$\begin{aligned}
Pr[E_{\mathcal{H}_2}] &\geq 2\epsilon \cdot \Pr[\neg abort] - \Pr[E_{\mathcal{H}_1}|\neg E_{\mathcal{H}_2}] + \Pr[E_{DEr}] \\
&\geq 2\epsilon \cdot \left(1 - \frac{q_{\mathcal{H}_4}}{2^{l_m}}\right) - \frac{q_{\mathcal{H}_1}}{2^{l_m+l_\sigma}} - q_{Dec}\Big(\frac{q_{H_1}}{(2^{l_m+l_\sigma})} + \frac{1}{p}\Big)
\end{aligned}$$

Note that, if event $E_{\mathcal{H}_2}$ takes place, the challenger $\mathcal{C}$ solves the n-$DBDHE$ instance with an advantage:

$$\begin{aligned}
\epsilon' &\geq \frac{1}{q_{\mathcal{H}_2}}Pr[E_{\mathcal{H}_2}] \\
&\geq \frac{1}{q_{\mathcal{H}_2}}\left(2\epsilon \cdot \left(1 - \frac{q_{\mathcal{H}_4}}{2^{l_m}}\right) - \frac{q_{\mathcal{H}_1}}{2^{l_m+l_\sigma}} - q_{Dec}\Big(\frac{q_{\mathcal{H}_1}}{(2^{l_m+l_\sigma})} + \frac{1}{p}\Big)\right)
\end{aligned}$$

We bound the running time of the challenger by:

$$t' \leq t + (q_{\mathcal{H}_3} + q_{\mathcal{H}_5} + (n+2)lq_{SK} + (n+6)lq_{RK} + 3q_{Dec} + 5q_{RD})t_e$$
$$+ (8q_{Dec} + 6q_{RD})t_{bp}.$$

This completes the proof of the theorem. □

Collusion Resistance

Theorem 3. *[15] If a unidirectional single-hop KP-ABPRE scheme is IND-PRE-CCA secure for first level ciphertexts, then it is collusion resistant as well.*

8 Conclusion

Although several KP-ABPRE schemes have been proposed in the literature, to the best of our knowledge, only one CCA-secure scheme due to Ge *et al*. has reported the collusion resistant property. In this paper, we demonstrate a CCA-attack on their construction. We also give a construction of the first unidirectional KP-ABPRE scheme with constant size ciphertexts requiring a constant number of exponentiations in encryption, and constant number of bilinear pairing operations during decryption and re-decryption while satisfying CCA-security for first-level and second level ciphertexts. Also, the definition of collusion resistance is met wherein a colluding proxy and delegatee cannot obtain the private key of the delegator. Our work affirmatively resolves the problem of collusion resistance by proposing a computationally efficient collusion resistant KP-ABPRE scheme that supports monotonic access structures for fine-grained delegation of decryption rights.

References

1. Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *NDSS*, 2005.
2. Nuttapong Attrapadung and Hideki Imai. Dual-policy attribute based encryption. In *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings*, pages 168–185, 2009.
3. Amos Beimel. *Secure schemes for secret sharing and key distribution.* PhD thesis, Israel Institute of technology, Technion, Haifa, Israel, 1996.
4. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007,California, USA*, pages 321–334. IEEE Computer Society, 2007.
5. Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 127–144. Springer, 1998.
6. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, 2004.
7. Pei-Shan Chung, Chi-Wei Liu, and Min-Shiang Hwang. A study of attribute-based proxy re-encryption scheme in cloud environments. *I. J. Network Security*, 16(1):1–13, 2014.
8. Aloni Cohen. What about bob? the inadequacy of CPA security for proxy reencryption. In *Public-Key Cryptography - PKC 2019 - 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part II*, pages 287–316, 2019.

9. Chunpeng Ge, Willy Susilo, Liming Fang, Jiandong Wang, and Yunqing Shi. A cca-secure key-policy attribute-based proxy re-encryption in the adaptive corruption model for dropbox data sharing system. *Des. Codes Cryptography*, 86(11):2587–2603, 2018.
10. Chunpeng Ge, Willy Susilo, Jiandong Wang, Zhiqiu Huang, Liming Fang, and Yongjun Ren. A key-policy attribute-based proxy re-encryption without random oracles. *Comput. J.*, 59(7):970–982, 2016.
11. Keying Li, Jianfeng Wang, Yinghui Zhang, and Hua Ma. Key policy attribute-based proxy re-encryption and RCCA secure scheme. *J. Internet Serv. Inf. Secur.*, 4(2):70–82, 2014.
12. Keying Li, Yinghui Zhang, and Hua Ma. Key policy attribute-based proxy re-encryption with matrix access structure. In *2013 5th International Conference on Intelligent Networking and Collaborative Systems, Xi'an city, Shaanxi province, China, September 9-11, 2013*, pages 46–50, 2013.
13. Kaitai Liang, Liming Fang, Duncan S Wong, and Willy Susilo. A ciphertext-policy attribute-based proxy re-encryption scheme for data sharing in public clouds. *Concurrency and Computation: Practice and Experience*, 27(8):2004–2027, 2015.
14. Xiaohui Liang, Zhenfu Cao, Huang Lin, and Jun Shao. Attribute based proxy re-encryption with delegating capabilities. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 276–286. ACM, 2009.
15. Benoît Libert and Damien Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In *International Workshop on Public Key Cryptography*, pages 360–379. Springer, 2008.
16. Benoît Libert and Damien Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. *IEEE Trans. Information Theory*, 57(3):1786–1802, 2011.
17. Song Luo, Jian-bin Hu, and Zhong Chen. Ciphertext policy attribute-based proxy re-encryption. In Miguel Soriano, Sihan Qing, and Javier López, editors, *Information and Communications Security - 12th International Conference, ICICS 2010, Barcelona, Spain, December 15-17, 2010. Proceedings*, volume 6476 of *Lecture Notes in Computer Science*, pages 401–415. Springer, 2010.
18. Y. Sreenivasa Rao and Ratna Dutta. Computational friendly attribute-based encryptions with short ciphertext. *Theor. Comput. Sci.*, 668:1–26, 2017.
19. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 457–473. Springer, 2005.
20. Jian Weng, Robert H. Deng, Xuhua Ding, Cheng-Kang Chu, and Junzuo Lai. Conditional proxy re-encryption secure against chosen-ciphertext attack. In *Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009, Sydney, Australia, March 10-12, 2009*, pages 322–332, 2009.