Practical Direct Chosen Ciphertext Secure Key-Policy Attribute-Based Encryption with Public Ciphertext Test

Weiran Liu[1,2], Jianwei Liu[1,3], Qianhong Wu[1,3], Bo Qin[2], and Yunya Zhou[1]

[1] School of Electronic and Information Engineering, Beihang University, Beijing
liuweiran900217@gmail.com, {liujianwei,qianhong.wu}@buaa.edu.cn
[2] School of Information, Renmin University of China, Beijing
bo.qin@ruc.edu.cn
[3] The Academy of Satellite Application, Beijing

Abstract. We propose a direct Key-Policy Attribute-Based Encryption (KP-ABE) scheme with semantic security against adaptively chosen ciphertext attacks (CCA2) in the standard model. Compared with its counterpart with security against chosen-plaintext attacks (CPA), the cost of our scheme is only a Chameleon hash. In contrast to the Boyen-Mei-Waters shrink approach from CPA-secure $(l+1)$-Hierarchical Identity Based Encryption ($(l+1)$-HIBE) to CCA2-secure l-HIBE, our approach only adds one on-the-fly dummy attribute. Further, our approach only requires that the underlying ABE is selectively secure and allows public ciphertext test. A major obstacle for the security proof in this scenario is that the simulator cannot prepare the challenge ciphertext associated with the on-the-fly dummy attribute due to the selective security constraint. We circumvent this obstacle with a Chameleon hash. Technically, unlike existing use of Chameleon hash in (online/offline) signature applications, our work shows Chameleon hash can also have unique applications in encryption schemes.

Keywords: Attribute-Based Encryption, Chameleon Hash, Chosen Ciphertext Security.

1 Introduction

Attribute-Based Encryption (ABE) allows an encryptor to share data with users according to specified access policies. ABE can be classified into two categories, Key-Policy ABE (KP-ABE) [21] and Ciphertext-Policy ABE (CP-ABE) [3]. In KP-ABE, secret keys are associated with access policies and ciphertexts are associated with sets of attributes. One can decrypt if and only if the set of attributes specified in the ciphertext satisfies the access policy in his/her secret key. In contrast, the CP-ABE ciphertexts are associated with access policies and the secret keys specify sets of attributes. Due to its capability of providing fine-grained access control over encrypted data, ABE is extensively applied for many cloud storage applications [17, 29, 32].

M. Kutyłowski and J. Vaidya (Eds.): ESORICS 2014, Part II, LNCS 8713, pp. 91–108, 2014.

Semantic security against adaptively chosen ciphertext attacks (CCA2) is widely recognized as a standard security notion for cryptosystems against active attacks. ABE is usually suggested to enforce fine-grained access control on outsourced data in cloud storage and computing applications. In such applications, active attackers who can modify ciphertexts in storage or in transit may reveal useful information from sensitive data even if the employed ABE system is chosen plaintext (CPA) secure. Thus, it is desirable to deploy CCA2-secure ABE to defend against such strong attackers.

There are some ABE schemes that can be shown CCA2 security in the standard model. They are reasonably less efficient than their CPA-secure counterparts. Some of them involve the one-time signature cryptographic primitives [16, 21, 33]. However, one-time signatures either have high storage or high computational cost. Specifically, one-time signature schemes based on cryptographic hash functions involve long public keys and signatures; and the one-time signature schemes based on number-theoretic assumptions have the advantage of short public keys and signatures, but yield expensive computational cost. Some other CCA2-secure ABE schemes are with the restriction of only supporting single threshold access policies [15, 20], which is inconvenient when the system is required to support complicated access policies.

It is preferable to construct CCA2-secure ABE from CPA-secure ones by directly using the underlying ABE structures and requiring no extra inefficient cryptographic primitives. In 2006, Boyen *et al.* [6] introduced a shrink approach that can directly obtain CCA2-secure l-Hierarchical Identity-Based Encryption (HIBE) from CPA-secure $(l+1)$-HIBE. By setting $l = 0$, their approach can directly obtain CCA2-secure Public Key Encryption (PKE) from CPA-secure Identity-Based Encryption (IBE). The key point is to first hash the intermediate ciphertext components independent of the identity to obtain a dummy identity, and then generate the final ciphertext using the dummy "identity". A natural question is whether we can construct a direct CCA2-secure ABE by applying their approach.

Issues in Direct CCA2-Secure ABE. There are three main issues in constructing CCA2-secure ABE using the Boyen-Mei-Waters approach.

Arbitrary-Attributes Requirement. Similarly, the hash output may be treated as an attribute in a direct CCA2-secure construction. However, the hash output cannot be controlled. This implies that the underlying ABE needs to have the property of "large universe", i.e., supporting arbitrary strings as attributes.

Delegatability Obstacle. The Boyen-Mei-Waters approach leverages the delegatability of (H)IBE systems. Specifically, one encrypts to the "hash identity" at a lower level than all users who can delegate a key to this hash identity for ciphertext validity test. This is the reason why their shrink approach converts a $(l+1)$-HIBE to be a CCA2-secure l-HIBE, and converts a CPA-secure IBE to be a CCA2-secure PKE. A straightforward application of their approach in ABE settings requires the underlying CPA-secure ABE allows to delegate a key to arbitrary attributes. However, the only ABE scheme due to Deng *et al.* [19] allowing hierarchical attributes cannot support arbitrary attributes.

Full Identity/Attribute Security Obstacle. If the (H)IBE system is of only selective-identity security, instead of full-identity security, then by applying Boyen-Mei-Waters approach one obtains a CCA2-secure (H)IBE Key Encapsulation Mechanism (KEM), instead of a fully functional encryption system. The main obstacle for fully functional encryption is that in the security proof, due to the selective security constraint, the simulator cannot prepare the challenge ciphertext associated with the hashed "identity". The same problem occurs in direct construction of CCA2-secure ABE if the underlying CPA-secure ABE has only selective security. It is challenging to obtain direct CCA2-secure ABE schemes from ABE schemes with selective CPA-security.

Our Contributions. We propose a direct publicly verifiable CCA2-secure KP-ABE scheme in the standard model. We achieve this goal by addressing the above issues in direct CCA2-secure KP-ABE construction. We exploit a recent CPA-secure KP-ABE one [30] with the property of "large universe". This property addresses the *Arbitrary-Attributes Requirement.*

We add one on-the-fly dummy attribute in our construction, instead of extending one attribute hierarchy. The on-the-fly dummy attribute is computed by hashing the intermediate ciphertext components independent of the specified attribute set. The other ciphertext components are generated by the new attribute set containing the dummy attribute. In the decryption procedure, the receiver can validate the ciphertext with the dummy attribute. This approach circumvent the *Delegatability Obstacle* with only a marginal cost, i.e., by adding constant size ciphertext components related to the dummy attribute. Further more, the ciphertext validity test only involves public information. This public ciphertext test property is enjoyable and allows a third party, e.g., a gateway or firewall, to filter encrypted spams in some applications.

Our proposal is a fully functional CCA2-secure ABE scheme, instead of a CCA2-secure KEM scheme, although the underlying ABE is only selectively secure. We circumvent the obstacle in the security proof by replacing a regular hash with a Chameleon hash. The cost to achieve CCA2 security from CPA security is only a Chameleon hash. The Chameleon hash plays a critical role in the security proof. Specifically, the universal forgeability (w.r.t. the hash trapdoor holder) of the Chameleon hash allows the simulator to prepare the challenge ciphertext associated with the hashed "attribute" even if the underlying ABE has only selective security. Technically, our constructions illustrate novel and unique applications of Chameleon hash in encryption systems, in contrast to previous main use of Chameleon hash in (online/offline) signature applications [1, 11, 14].

Related Work. ABE was introduced by Sahai and Waters [31]. Goyal *et al.* extended the idea and distinguished KP-ABE and CP-ABE. The KP-ABE and CP-ABE systems were then respectively proposed by Goyal *et al.* [21] and Bethencourt *et al.* [3] that support monotonic access policies. Fully secure constructions in the standard model were first provided by Okamoto *et al.* [27] and Lewko *et al.* [23]. Many other ABE schemes have been further proposed to gain more preferable properties, such as hierarchical ABE (allowing users with attributes of higher hierarchy to delegate keys to lower levels) [17], "non-monotonic

access policies" (supporting general access structures with negation boolean formulas) [26], "large universe" [24, 25, 28, 30], and "multiple central authorities" (there exists multiple authenticated PKGs in ABE) [8, 9, 24]. The latest work [18] on ABE achieves black-box traitor tracing in which a tracing algorithm can be invoked to find the secret keys leaked for illegal access to encrypted data.

Several ABE systems have been proposed with CCA2 security in the standard model. The KP-ABE scheme proposed by Goyal *et al.* [21] can be converted to have CCA2 security by revising the Canetti-Halevi-Katz approach [7] from CPA-secure IBE [5] to CCA2-secure PKE schemes at the cost of one-time signatures. Cheung *et al.* [16] leveraged the Canetti-Halevi-Katz approach [7] in CP-ABE to construct a CCA2-secure CP-ABE. Yamada *et al.* [33] introduced a generic construction to transform CPA-secure ABE to CCA2-secure ones if the involved ABE schemes satisfy either delegatability or verifiability. The above CCA2-secure ABE schemes involve one-time signatures. Chen *et al.* [15] and Ge *et al.* [20] recently constructed direct CCA2-secure ABE without one-time signatures, with a restriction on only supporting threshold access policies.

Paper Organization. The rest of the paper is organized as follows. In Sec. 2, we review prime-order bilinear groups, the number-theoretic assumption we use, and the background information about access structures, linear secret-sharing schemes and Chameleon hash functions. Sec. 3 formalizes KP-ABE and their CCA2 security definitions. We propose our practical CCA2-secure KP-ABE with detailed analyses in Sec. 4, followed by concluding remarks in Sec. 5.

2 Preliminaries

2.1 Notations

We use $[a, b]$ to denote the set $\{a, a+1, \cdots, b\}$ containing consecutive integers. We write $[a]$ as shorthand for $[1, a]$ if no ambiguities are caused. For a set S, its cardinality is denoted by $|S|$. We denote $s_1, s_2, \cdots, s_n \xleftarrow{R} S$ for $n \in \mathbb{N}$ to show that $s_1, \cdots, s_n$ is chosen uniformly at random from S. We use $\mathbb{Z}_p^{m\times n}$ to denote the matrices of m rows and n columns with entries in $\mathbb{Z}_p$. Specifically, the row vectors and column vectors are denoted by $\mathbb{Z}_p^{1\times n}$ and $\mathbb{Z}_p^{m\times 1}$ respectively. For the given two vectors $\vec{v}, \vec{w}$ of any type, we denote by v_i the i-th entry in $\vec{v}$, and by $\left\langle \vec{v}, \vec{w} \right\rangle$ the inner product of the two vectors.

2.2 Bilinear Groups and Computational Assumption

Our scheme is built on prime-order bilinear groups which can be efficiently generated by a generator $\mathcal{G}$ with a security parameter λ. The bilinear group system can be represented as tuple $(p, \mathbb{G}, \mathbb{G}_T, e)$, where p is a large prime, $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups of order p, and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfying three properties. (1) *Bilinearity*: for all $g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, $e(g^a, h^b) = e(g, h)^{ab}$; (2)*Non-degeneracy*: there exists at least an element $g \in \mathbb{G}$ such that $e(g, g)$ has

order p in $\mathbb{G}_T$; (3) *Computability*: there exists an efficient algorithm (in polynomial time with respect to λ) to compute the bilinear pairing $e(u, v)$ for all $u, v \in \mathbb{G}$. Although our scheme is built from above symmetric pairing groups, the constructions do not rely on the symmetric property of the bilinear groups and can be easily extended to asymmetric pairing groups.

The security of our scheme relies on a weak version of the Decisional Bilinear Diffie-Hellman Assumption (wDBDH), introduced by Rouselakis and Waters [30], reviewed as follows.

Let $(p, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$ be the description of the bilinear groups that is outputted by the group generator $\mathcal{G}$. Let $g \xleftarrow{R} \mathbb{G}$ be a random generator of $\mathbb{G}$. Then, choose $q+3$ random exponents $x, y, z, b_1, b_2, \cdots, b_q \xleftarrow{R} \mathbb{Z}_p$. The q-wDBDH problem in $\mathbb{G}$ is to determine whether the given element $T \in \mathbb{G}_T$ equals $e(g, g)^{xyz}$, or a random element in $\mathbb{G}_T$ by taken the input as

$$D \leftarrow \begin{pmatrix} g, g^x, g^y, g^z, g^{(xz)^2} & \\ g^{b_i}, g^{xzb_i}, g^{xz/b_i}, g^{x^2zb_i}, g^{y/b_i^2}, g^{y^2/b_i^2} & i \in [q] \\ g^{xzb_i/b_j}, g^{yb_i/b_j^2}, g^{xyzb_i/b_j}, g^{(xz)^2b_i/b_j} & i \in [q], j \in [q], i \neq j \end{pmatrix}$$

The advantage of an algorithm $\mathcal{A}$ that outputs $b \in \{0, 1\}$ in solving q-wDBDH in $\mathbb{G}$ is defined as

$$Adv_{\mathcal{A}}(\lambda) = \left| \Pr\left[\mathcal{A}\left(D, T = e(g,g)^{xyz}\right) = 1\right] - \Pr\left[\mathcal{A}\left(D, T \xleftarrow{R} \mathbb{G}_T\right) = 1\right] \right| - \frac{1}{2}$$

Definition 1. *We say that the (ϵ, q)-weak Decisional Bilinear Diffie-Hellman Assumption in $\mathbb{G}$ holds if no polynomial time algorithm has at least a non-negligible advantage $Adv_{\mathcal{A}}(\lambda) \geq \epsilon$ in solving the q-wDBDH problem in $\mathbb{G}$.*

2.3 Access Structures and Linear Secret Sharing Schemes

Definition 2. *(**Access Structure** [2]) Let $\mathcal{U}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\mathcal{U}}$ is monotone if for all $B \in \mathbb{A}$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. An access structure (monotone access structure) on $\mathcal{U}$ is a collection (monotone collection) $\mathbb{A}$ of non-empty subsets of $\mathcal{U}$, i.e., $\mathbb{A} \subseteq 2^{\mathcal{U}} \backslash \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.*

In Attribute-Based Encryption systems, the roles of the parties are taken by the attributes in the attribute universe $\mathcal{U}$. Therefore, the access structure $\mathbb{A}$ will contain the authorized sets of attributes. We restrict the access structure to monotone access structure. However, it is possible to realize general access structure from a monotone one by having the not of an attribute as a separate attribute altogether [3, 21], at a cost of doubling the total number of attributes in the system.

Definition 3. *(**Linear Secret Sharing Schemes (LSSS)** [2, 30]) Let $\mathcal{U}$ be the attribute universe. A secret sharing scheme $\prod$ with domain of secrets $\mathbb{Z}_p$ for realizing access structure on $\mathcal{U}$ is linear if*

1. *The shares of a secret $s \in \mathbb{Z}_p$ for each attribute form a vector over $\mathbb{Z}_p$.*
2. *For each access structure $\mathbb{A}$ on $\mathcal{U}$, there exists a share-generating matrix $M \in \mathbb{Z}_p^{l \times n}$ for $\prod$. For all $i \in [l]$, we define the function $\rho(i)$ that labels the i-th row of M with attributes from $\mathcal{U}$, i.e., $\rho(i) : i \to \mathcal{U}$. When we consider the column vector $\vec{v} = (s, r_2, r_3, \cdots, r_n)^{\mathrm{T}}$, where $r_2, \cdots, r_n \xleftarrow{R} \mathbb{Z}_p$, then $M\vec{v} \in \mathbb{Z}_p^{l \times 1}$ is the vector of l shares of the secret s according to $\prod$. The share $(M\vec{v})_i$ belongs to the attribute $\rho(i)$ for $i \in [l]$.*

From now on, we refer to the tuple (M, ρ) as the access structure $\mathbb{A}$ encoded by the LSSS-policy. As pointed out by Beimel [2], all secret sharing schemes should satisfy the following requirements:

1. *Reconstruction Requirement.* The secret can be reconstructed efficiently for authorized sets.
2. *Security Requirement.* It is hard to reveal any partial information about the secret for any unauthorized sets.

These two requirements are used in our setting. Let $\mathcal{S}$ denote an authorized set for the access structure $\mathbb{A}$ encoded by the LSSS-policy (M, ρ), where $M \in \mathbb{Z}_p^{l \times n}$ and $\rho : [l] \to \mathcal{U}$. We define $I_{\mathcal{S}} \subseteq [l]$ as $I_{\mathcal{S}} = \{i : \rho(i) \in \mathcal{S}\}$. On one hand, the *Reconstruction Requirement* states that there exists constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that for any valid shares $\{\lambda_i = (M\vec{v})_i\}_{i \in I}$ of a secret s according to $\prod$, we have $\sum_{i \in I} \omega_i \lambda_i = s$. Additionally, the constants $\{\omega_i\}_{i \in I}$ can be generated in time polynomial in the size of the share-generating matrix M [2]. On the other hand, the *Security Requirement* states that for any unauthorized sets $\mathcal{S}'$ for the access structure $\mathbb{A}$, such constants $\{\omega_i\}$ do not exist. We define $I_{\mathcal{S}'} \subseteq [l]$ as $I_{\mathcal{S}'} = \{i : \rho(i) \in \mathcal{S}'\}$. In this case, there exists a vector $\vec{\omega} = (\omega_1, \cdots, \omega_n) \in \mathbb{Z}_p^{1 \times n}$, such that $\langle \vec{M}_i, \vec{\omega} \rangle = 0$ for all $i \in I_{\mathcal{S}'}$ and ω_1 can be any non zero element in $\mathbb{Z}_p$, where $\vec{M}_i$ is the i-th row of the secret-generating matrix M.

2.4 Chameleon Hash

Our scheme exploits the so-called Chameleon hash functions which were first introduced by Krawczyk and Rabin [22], further refined respectively by Ateniese *et al.* [1] and by Chen *et al.* [10–14]. A Chameleon hash has a key pair (pk_{ch}, sk_{ch}). Anyone who knows the public key pk_{ch} can efficiently compute the hash value for any given input. Meanwhile, there exists an efficient algorithm for the holder of the secret key sk_{ch} to find collisions for every given input, but anyone who does not have sk_{ch} cannot compute the collisions for any given input. Formally, a Chameleon hash function consists of three polynomial time algorithms:

– $(sk_{ch}, pk_{ch}) \leftarrow \mathbf{KeyGen_{ch}}(1^{\lambda})$. The algorithm $\mathbf{KeyGen_{ch}}$ takes the security parameter $\lambda \in \mathbb{N}$ as input, and outputs a pair containing a Chameleon hash secret key and a public key (sk_{ch}, pk_{ch}).

- $H_m \leftarrow \mathbf{Hash_{ch}}(pk_{ch}, m, r_{ch})$. The algorithm $\mathbf{Hash_{ch}}$ takes as inputs the Chameleon hash public key pk_{ch}, a message m, and an auxiliary random parameter r_{ch}. It outputs the hashed value H_m.
- $r'_{ch} \leftarrow \mathbf{UForge_{ch}}(sk_{ch}, m, r_{ch}, m')$. The algorithm $\mathbf{UForge_{ch}}$ takes as inputs the Chameleon hash secret key sk_{ch}, a message m with its auxiliary random parameter r_{ch} for perviously calculating its Chameleon hash value H_m, and another message $m' \neq m$. The algorithm outputs another auxiliary random parameter r'_{ch} such that $H_m = \mathbf{Hash_{ch}}(pk_{ch}, m, r_{ch}) = \mathbf{Hash_{ch}}(pk_{ch}, m', r'_{ch}) = H_{m'}$.

A Chameleon hash function should satisfy the following security requirements.

1. *Collision Resistance.* There is no efficient algorithm that takes as input the Chameleon hash public key pk_{ch} to find two pairs (m, r_{ch}), (m', r'_{ch}) where $m \neq m'$, such that $\mathbf{Hash_{ch}}(pk_{ch}, m, r_{ch}) = \mathbf{Hash_{ch}}(pk_{ch}, m', r'_{ch})$ except with negligible probability.
2. *Uniformity.* All messages m induce the same probability distribution on $H_m \leftarrow \mathbf{Hash_{ch}}(pk_{ch}, m, r_{ch})$ for r_{ch} chosen uniformly at random.

3 Key-Policy Attribute-Based Encryption

A KP-ABE system consists of four polynomial time algorithms defined as follows.

- $(pp, msk) \leftarrow \mathbf{Setup}(1^\lambda)$. The algorithm **Setup** only takes the security parameter $\lambda \in \mathbb{N}$ as input. It outputs a master secret key msk and a public parameter pp. We assume that the description of the attribute universe $\mathcal{U}$ is also included in the public parameter pp.
- $sk_{\mathbb{A}} \leftarrow \mathbf{KeyGen}(pp, msk, \mathbb{A})$. The algorithm **KeyGen** takes as inputs the public parameter pp, the master secret key msk, and an access structure $\mathbb{A}$. It outputs a secret key $sk_{\mathbb{A}}$ associated with the access structure $\mathbb{A}$.
- $ct_{\mathcal{S}} \leftarrow \mathbf{Encrypt}(pp, m, \mathcal{S})$. The algorithm **Encrypt** takes as inputs the public parameter pp, a message m in the plaintext space $\mathcal{M}$, and a set of attributes $\mathcal{S}$ on the attribute universe $\mathcal{U}$. The algorithm outputs the ciphertext $ct_{\mathcal{S}}$ for m associated with the attribute set $\mathcal{S}$.
- $m \leftarrow \mathbf{Decrypt}(pp, sk_{\mathbb{A}}, ct_{\mathcal{S}})$. The algorithm **Decrypt** takes as inputs the public parameter pp, a secret key $sk_{\mathbb{A}}$ associated with an access structure $\mathbb{A}$, and a ciphertext $ct_{\mathcal{S}}$ for a message $m \in \mathcal{M}$ associated with an attribute set $\mathcal{S}$. It returns m.

A KP-ABE system is correct if for all $(pp, msk) \leftarrow \mathbf{Setup}(1^\lambda)$, all $sk_{\mathbb{A}} \leftarrow \mathbf{KeyGen}(pp, msk, \mathbb{A})$, all $m \in \mathcal{M}$, and all $ct_{\mathcal{S}} \leftarrow \mathbf{Encrypt}(pp, m, \mathcal{S})$ with $\mathcal{S} \subseteq \mathcal{U}$, if $\mathcal{S}$ satisfies $\mathbb{A}$, then $\mathbf{Decrypt}(pp, sk_{\mathbb{A}}, ct_{\mathcal{S}}) = m$.

We next define the indistinguishability against chosen ciphertext attacks in KP-ABE systems. In this security model, the adversary is allowed to obtain the secret keys associated with any access structure $\mathbb{A}$ of its choice and to issue decryption queries for its chosen ciphertexts, provided that the adversary does not query for the secret keys with access structures that can be satisfied

by the challenge attribute set $\mathcal{S}^*$, or for the challenge ciphertext of one of its chosen message. We require that even such an adversary cannot distinguish the encrypted messages.

Formally, the selective chosen attribute set and chosen ciphertext security model is defined through a game played by an adversary and a challenger. Both of them are given the security parameter λ as input.

- **Init.** The adversary $\mathcal{A}$ commits to a challenge attribute set $\mathcal{S}^*$ and sends it to the challenger.
- **Setup.** The challenger gives the public parameter pp to the adversary $\mathcal{A}$.
- **Phase 1.** The adversary $\mathcal{A}$ adaptively issues two kinds of queries:
  - Secret key query for an access structure $\mathbb{A}$ that is not satisfied by the challenge attribute set $\mathcal{S}^*$. The challenger generates a secret key for $\mathbb{A}$ and gives it to the adversary.
  - Decryption query for the ciphertext $ct_{\mathcal{S}}$ with an attribute set $\mathcal{S}$. The challenger responds by constructing an access structure $\mathbb{A}$ satisfied by the attribute set $\mathcal{S}$, and running **KeyGen**(pp, msk, $\mathbb{A}$) to generate a secret key $sk_{\mathbb{A}}$. It then runs **Decrypt**(pp, $sk_{\mathbb{A}}$, $ct_{\mathcal{S}}$) to decrypt the ciphertext $ct_{\mathcal{S}}$ and returns the resulting message to the adversary.
- **Challenge.** When adversary $\mathcal{A}$ decides that **Phase 1** is over, it outputs two equal-length messages m_0 and m_1 on which it wishes to challenge. The challenger flips a random coin $b \in \{0, 1\}$ and encrypts m_b under the challenge attribute set $\mathcal{S}^*$. It returns the challenge ciphertext ct^* to $\mathcal{A}$.
- **Phase 2.** The adversary $\mathcal{A}$ further adaptively issues two kinds of queries:
  - Secret key query for access structures $\mathbb{A}$ that is not satisfied by the challenge attribute set $\mathcal{S}^*$.
  - Decryption query for the ciphertext $ct_{\mathcal{S}}$ with a constraint that $ct_{\mathcal{S}} \neq ct^*$.

  The challenger responds the same as in **Phase 1**.
- **Guess.** Finally, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins in the game if $b = b'$.

The advantage of such an adversary $\mathcal{A}$ in attacking the KP-ABE system with security parameter λ is defined as $Adv_{\mathcal{A}}^{\text{KP-ABE}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$.

Definition 4. *A KP-ABE system is selective chosen attribute set and chosen ciphertext secure if for any probabilistic polynomial time adversary $\mathcal{A}$, the advantage of breaking the security game defined above is at most a negligible function.*

4 Direct CCA2-Secure KP-ABE with Public Verifiability

4.1 Basic Ideas

We first provide an overview of the construction. We exploit specific ciphertext structure in the ABE scheme in [30] and address the three issues shown in Sec. 1 to obtain a practical CCA2-secure KP-ABE scheme in the standard model.

1. Our construction is based on a recent KP-ABE system [30]. The CCA2-secure IBE schemes [6] exploits a specific structure of the underlying IBE ciphertext [4] including Decision Diffie-Hellman (DDH) tuple, i.e., $(g, g^r, h \cdot u^{ID}, (h \cdot u^{ID})^r)$, where ID is the target identity. The KP-ABE system [30] contains a DDH ciphertext tuple and allows arbitrary attributes, which addresses the arbitrary attribute requirement.
2. Instead of extending an attribute hierarchy in ABE, our construction adds one on-the-fly dummy attribute used for ciphertext validation in the decryption procedure. We split the original attribute universe into two parts, one for regular attribute universe $\mathcal{U}$, the other for verification universe $\mathcal{V}$ for the on-the-fly attributes. This trick ensures that the dummy attributes will only be used for ciphertext validation, and allows us to circumvent the arbitrary attribute delegation obstacle.
3. We exploit Chameleon hash $\mathbf{Hash_{ch}}$ to solve the problem that the simulator cannot in advance know the challenge on-the-fly dummy attribute in the security proof. In the **Setup** phase, the simulator generates a temporary message and calls $\mathbf{Hash_{ch}}$ to obtain the on-the-fly dummy attribute. When learning the actual challenge message in the **Challenge** phase, the simulator replaces the temporary message to the actual message, while keeping the dummy attribute unchanged by using the "universe collision" property of Chameleon hash. In the adversary's view, the Chameleon hash function keeps collision resistant since it does not know the Chameleon hash secret key.

4.2 Our Construction

Let $\mathcal{U} = \left[0, \frac{p-1}{2}\right]$ be the regular attribute universe, and $\mathcal{V} = \left[\frac{p+1}{2}, p-1\right]$ the verification universe. Note that $\mathcal{U} \cap \mathcal{V} = \emptyset$, and $\mathcal{U} \cup \mathcal{V} = \mathbb{Z}_p$, where $\mathbb{Z}_p$ is the original attribute universe in Rouselakis-Waters KP-ABE. Our CCA2-secure KP-ABE scheme works as follows.

– $\mathbf{Setup}(1^\lambda)$. Run $(p, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$ to generate a prime p, two groups $\mathbb{G}, \mathbb{G}_T$ of order p, and a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. A secure Chameleon hash function $\mathbf{Hash_{ch}}$: $\{0,1\}^* \rightarrow \left[\frac{p+1}{2}, p-1\right]$ with an auxiliary parameter universe $\mathcal{R}$ is also employed in the scheme. Then, select a random generator $g \xleftarrow{R} \mathbb{G}$, random elements $h, u, w \xleftarrow{R} \mathbb{G}$, and a random exponent $\alpha \xleftarrow{R} \mathbb{Z}_p$. The algorithm calls $(sk_{ch}, pk_{ch}) \leftarrow \mathbf{KeyGen_{ch}}(1^\lambda)$ to obtain a (sk_{ch}, pk_{ch}) pair for the Chameleon hash. The public parameter is $pp \leftarrow (\mathbf{Hash_{ch}}, \mathcal{R}, pk_{ch}, g, h, u, w, e(g,g)^\alpha)$. The master secret key is $msk \leftarrow (\alpha)$.

– $\mathbf{KeyGen}(pp, msk, (M, \rho))$. For generating a secret key for an access structure encoded by the LSSS-policy (M, ρ), where $M \in \mathbb{Z}_p^{l \times n}$ and $\rho : [l] \rightarrow \left[0, \frac{p-1}{2}\right]$, the key generation algorithm first picks $n-1$ random exponents $y_2, \cdots, y_n \xleftarrow{R} \mathbb{Z}_p$, and constructs the vector $\vec{y} = (\alpha, y_2, \cdots, y_n)^{\mathrm{T}}$. Then, the vector of the shares $\vec{\lambda}$ can be computed by the key generation algorithm as $\vec{\lambda} = (\lambda_1, \lambda_2, \cdots, \lambda_l)^{\mathrm{T}} = M\vec{y}$. The algorithm next picks

l random exponents $t_1, t_2, \cdots, t_l \xleftarrow{R} \mathbb{Z}_p$. For every $i \in [l]$, it calculates $K_{i,0} = g^{\lambda_i} w^{t_i}, K_{i,1} = \left(h \cdot u^{\rho(i)}\right)^{-t_i}, K_{i,2} = g^{t_i}$. The secret key is

$$sk_{(M,\rho)} \leftarrow \left((M,\rho), \{K_{i,0}, K_{i,1}, K_{i,2}\}_{i\in[l]}\right)$$

– **Encrypt**(pp, m, $\mathcal{S}$). Given the attribute set $\mathcal{S} = \{A_1, A_2, \cdots, A_\kappa\}$, where $\kappa = |\mathcal{S}|$, the algorithm first picks $\kappa + 2$ random exponents $s, r_0, r_1, \cdots, r_\kappa \xleftarrow{R} \mathbb{Z}_p$. It then computes $C = m \cdot e(g,g)^{\alpha s}$, $C_0 = g^s$, $C_{0,1} = g^{r_0}$, and for every $i \in [\kappa]$, $C_{i,1} = g^{r_i}, C_{i,2} = \left(h \cdot u^{A_i}\right)^{r_i} w^{-s}$. It picks a parameter $r_{ch} \xleftarrow{R} \mathcal{R}$ and sets $V = \mathbf{Hash_{ch}}\left(pk_{ch}, pk_{ch} \| C \| C_0 \| C_{0,1} \| C_{1,1} \| C_{2,1} \| \cdots \| C_{\kappa,1}, r_{ch}\right) \in \mathcal{V}$. The $C_{0,2}$ component can be computed as $C_{0,2} = \left(h \cdot u^V\right)^{r_0} w^{-s}$. The algorithm finally outputs the ciphertext associated with the attribute set $\mathcal{S}$ as

$$ct_{\mathcal{S}} \leftarrow \left(\mathcal{S}, r_{ch}, C, C_0, C_{0,1}, C_{0,2}, \{C_{i,1}, C_{i,2}\}_{i\in[\kappa]}\right).$$

– **Decrypt**(pp, $sk_{(M,\rho)}$, $ct_{\mathcal{S}}$). Before decrypting $ct_{\mathcal{S}}$, the algorithm first calculates $V = \mathbf{Hash_{ch}}\left(pk_{ch}, pk_{ch} \| C \| C_0 \| C_{0,1} \| C_{1,1} \| C_{2,1} \| \cdots \| C_{\kappa,1}, r_{ch}\right)$, where $\kappa = |\mathcal{S}|$. Then, it verifies whether the ciphertext is legitimate by testing whether the following equation holds for each $i \in [\kappa]$

$$e\left(g, C_{i,2}\right) \stackrel{?}{=} e\left(C_{i,1}, h \cdot u^{A_i}\right) / e\left(C_0, w\right) \tag{1}$$

It additionally tests whether the following equation holds

$$e\left(g, C_{0,2}\right) \stackrel{?}{=} e\left(C_{0,1}, h \cdot u^{V}\right) / e\left(C_0, w\right) \tag{2}$$

Note that the above ciphertext validity test can be done publicly since it only involves public parameter pp and ciphertext $ct_{\mathcal{S}}$.
If any equality does not hold, the ciphertext is invalid and the decryption algorithm outputs $\perp$. Otherwise, the algorithm calculates the row set of M that provides the share to attributes in the given attribute set $\mathcal{S}$, i.e., $I = \{i : \rho(i) \in \mathcal{S}\}$. Then, it computes the constants $\{\omega_i \in \mathbb{Z}_p\}_{i\in I}$ such that $\sum_{i\in I} \omega_i \vec{M}_i = (1, 0, \cdots, 0)$, where $\vec{M}_i$ is the i-th row of the share-generating matrix M. We note that the constants $\{\omega_i \in \mathbb{Z}_p\}_{i\in I}$ can be found in polynomial time for all $\mathcal{S}$ that satisfies the access structure [2].
Finally, the message m can be recovered by computing

$$B = \prod_{i\in I} \left(e\left(K_{i,0}, C_0\right) e\left(K_{i,1}, C_{j,1}\right) e\left(K_{i,2}, C_{j,2}\right)\right)^{\omega_i}$$

where j is the index of the attribute $\rho(i)$ in $\mathcal{S}$ that is depended on i, and computing $m = C/B$.

Consistency. If the ciphertext is associated with the attribute set $\mathcal{S}$, then for every $i \in [\kappa]$ where $\kappa = |\mathcal{S}|$, we have that

$$\begin{aligned} e\left(g, C_{i,2}\right) &= e\left(g, \left(h \cdot u^{A_i}\right)^{r_i} w^{-s}\right) = e\left(g, \left(h \cdot u^{A_i}\right)^{r_i}\right) \cdot e\left(g, w^{-s}\right) \\ &= e\left(g^{r_i}, h \cdot u^{A_i}\right) \cdot e\left(g^{-s}, w\right) = \frac{e\left(g^{r_i}, h \cdot u^{A_i}\right)}{e\left(g^s, w\right)} = \frac{e\left(C_{i,1}, h \cdot u^{A_i}\right)}{e\left(C_0, w\right)} \end{aligned}$$

Accordingly, Equation (1) holds. With the similar procedure shown above, Equation (2) holds for the valid on-the-fly dummy attribute

$$V = \mathbf{Hash_{ch}}\left(pk_{ch}, pk_{ch} \| C_0 \| C_{1,1} \| C_{2,1} \| \cdots \| C_{\kappa,1}, r_{ch}\right)$$

Also, if the attribute set $\mathcal{S}$ of the ciphertext $ct_{\mathcal{S}}$ satisfies the access structure encoded by the LSSS-policy (M, ρ), then $\sum_{i \in I} \omega_i \lambda_i = \alpha$. Since j is the index of the attribute $\rho(i)$ in $\mathcal{S}$, we have that $\rho(i) = A_j$. Therefore

$$\begin{aligned}
B &= \prod_{i \in I} \left(e\left(K_{i,0}, C_0\right) e\left(K_{i,1}, C_{j,1}\right) e\left(K_{i,2}, C_{j,2}\right)\right)^{\omega_i} \\
&= \prod_{i \in I} \left(e\left(g^{\lambda_i} w^{t_i}, g^s\right) e\left(\left(h \cdot u^{\rho(i)}\right)^{-t_i}, g^{r_j}\right) e\left(g^{t_i}, \left(h \cdot u^{A_j}\right)^{r_j} w^{-s}\right)\right)^{\omega_i} \\
&= \prod_{i \in I} \left(e\left(g^{\lambda_i}, g^s\right) e\left(w^{t_i}, g^s\right) e\left(g^{t_i}, w^{-s}\right)\right)^{\omega_i} \\
&= \prod_{i \in I} \left(e\left(g^{\lambda_i}, g^s\right)\right)^{\omega_i} = e(g, g^s)^{\sum_{i \in I} \omega_i \lambda_i} = e(g, g)^{\alpha s}
\end{aligned}$$

Thus, we have that $C/B = m \cdot e(g, g)^{\alpha s} / e(g, g)^{\alpha s} = m$.

Public Verifiability. Our scheme is a publicly verifiable CCA2-secure KP-ABE, since the above ciphertext validity test only involves public parameter pp and ciphertext $ct_{\mathcal{S}}$. This property is useful to build advanced Attribute-Based cryptography protocols, e.g., ciphertext filtering KP-ABE, in which anyone (e.g., a gateway of a firewall) can verify whether the ciphertext is encrypted by the specified access structure to filter spams (i.e., invalid ciphertexts) without requiring the secret keys of the receivers.

4.3 Performance Analysis

Table 1 compares our CCA2-secure KP-ABE with the underlying Rouselakis-Waters CPA-secure KP-ABE. In the table, the secret key $sk_{(M,\rho)}$ is associated with the LSSS-policy (M, ρ) with $M \in \mathbb{Z}_p^{l \times n}$, and the ciphertext $ct_{\mathcal{S}}$ is associated with the attribute set $\mathcal{S}$ with $\kappa = |\mathcal{S}|$. We denote τ_e as one exponent operation in $\mathbb{G}$ and $\mathbb{G}_T$, τ_m as one multiplication operation in $\mathbb{G}$ and $\mathbb{G}_T$, τ_p as one pairing operation time, and τ_h as one Chameleon hash operation time.

From Table 1, it can be seen that the additional overheads for CCA2-security are considerably low. Specifically, the secret key size in our CCA2-secure KP-ABE scheme remain the same as that of the underlying Rouselakis-Waters KP-ABE scheme [30]. For a ciphertext associated with arbitrary number of attributes, only two more group elements in $\mathbb{G}$ are added. The encryption algorithm needs only one more hash operation, three more exponent operations in $\mathbb{G}$ and two more multiplication operation in $\mathbb{G}$.

Table 1. Comparisons among Rouselakis-Waters KP-ABE and our KP-ABE

	Rouselakis-Waters ABE	Our KP-ABE
Security	CPA-secure	CCA2-secure
pp Size	5	$5 + \|pk_{ch}\|$
$sk_{(M,\rho)}$ Size	$3l + \|(M,\rho)\|$	$3l + \|(M,\rho)\|$
$ct_{\mathcal{S}}$ Size	$3\kappa + 2$	$3\kappa + 4 + \|r_{ch}\|$
KeyGen Time	$5l \cdot \tau_e + 2l \cdot \tau_m$	$5l \cdot \tau_e + 2l \cdot \tau_m$
Encrypt Time	$(3\kappa + 2) \cdot \tau_e + (2\kappa + 1) \cdot \tau_m$	$(3\kappa + 5) \cdot \tau_e + (2\kappa + 3) \cdot \tau_m + \tau_h$

We note that the work in [16] also exploits dummy attributes to achieve CCA2-secure ABE. In [16], signatures are added to CPA-secure ABE for validating ciphertext in decryption. Each bit of the verification keys K_v is treated as an attribute, which introduces $|K_v|$ dummy attributes. In contrast, we only introduce one dummy attribute. Our approach is compact and efficient.

4.4 Security Analysis

The Chameleon hash is critical in our proof. In the **Init** phase, the simulator chooses a random challenge message m^* and an auxiliary parameter r^* to construct the challenge on-the-fly dummy attribute V^*. In the **Challenge** phase, given the challenge messages m_0, m_1, the simulator obtains a collision pair (m_b^*, r_b^*) such that $\mathbf{Hash_{ch}}(pk_{ch}, m^*, r^*) = \mathbf{Hash_{ch}}(pk_{ch}, m_b^*, r_b^*) = V^*$ to make the challenge on-the-fly dummy attribute V^* unchanged. This approach allows the simulator in advance obtain the challenge dummy attribute. Since all challenge attributes can be obtained in the **Setup** phase, the simulator can correctly play the selective security game with the adversary.

Formally, the selective chosen attribute set and chosen ciphertext security result is guaranteed by Theorem 1.

Theorem 1. *Let $\mathbb{G}$ be a group of prime order p equipped with an efficient bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Our KP-ABE scheme is selective chosen attribute set and chosen ciphertext secure if the $(\epsilon, q+1)$-wDBDH assumption holds in $\mathbb{G}$, the employed Chameleon hash function is secure, and the challenge attribute set $\mathcal{S}^*$ that the adversary commits to satisfies that $|\mathcal{S}^*| \leq q$.*

Proof. Suppose that there exists an algorithm $\mathcal{A}$ that has advantage ϵ to break our KP-ABE system in the security game defined in Sec. 3. We construct an algorithm $\mathcal{B}$ that can solve the $(q+1)$-wDBDH problem in $\mathbb{G}$. The input of the algorithm $\mathcal{B}$ is the challenge tuple (D, T) of the $(q+1)$-wDBDH problem. Algorithm $\mathcal{B}$ interacts with $\mathcal{A}$ as follows.

Init. Algorithm $\mathcal{A}$ sends the challenge set $\mathcal{S}^* = \{A_1^*, A_2^*, \cdots, A_\kappa^*\}$ to $\mathcal{B}$.

Setup. Algorithm $\mathcal{B}$ sets the public parameter pp as follows. It randomly chooses $\widetilde{u} \overset{R}{\leftarrow} \mathbb{Z}_p$ and sets

$$(g, u, w, e(g,g)^\alpha) = (g, g^{\widetilde{u}} \cdot \prod_{i \in [\kappa+1]} g^{y/b_i^2}, g^x, e(g^x, g^y))$$

It then sets $C_0^* = g^s = g^z$, $C_{0,1}^* = g^{r_0} = g^{b_{\kappa+1}}$, and $C_{i,1}^* = g^{r_i} = g^{b_i}$ for all $i \in [\kappa]$. It next chooses a secure Chameleon hash function $\mathbf{Hash_{ch}} : \{0,1\}^* \to \left[\frac{p+1}{2}, p-1\right]$ with an auxiliary parameter universe $\mathcal{R}$ and runs $(sk_{ch}, pk_{ch}) \leftarrow \mathbf{KeyGen_{ch}}(1^\lambda)$. Algorithm $\mathcal{B}$ picks a random auxiliary parameter $r_{ch}^* \overset{R}{\leftarrow} \mathcal{R}$, a random challenge message $m^* \overset{R}{\leftarrow} \mathbb{G}_T$, sets $C^* = m^* \cdot T$, and calls $\mathbf{Hash_{ch}}$ to calculate $V^* = \mathbf{Hash_{ch}}(pk_{ch}, C^*, r_{ch}^*)$. Finally, algorithm $\mathcal{B}$ picks a random exponent $\widetilde{h} \overset{R}{\leftarrow} \mathbb{Z}_p$ and sets

$$h = g^{\widetilde{h}} \cdot \prod_{i \in [\kappa+1]} g^{xz/b_i} \cdot \prod_{i \in [\kappa]} \left(g^{y/b_i^2}\right)^{-A_i^*} \left(g^{y/b_{\kappa+1}^2}\right)^{-V^*}$$

The public parameter is $pp \leftarrow (\mathbf{Hash_{ch}}, \mathcal{R}, pk_{ch}, g, h, u, w, e(g,g)^\alpha)$ and $\mathcal{B}$ is implicitly set $msk \leftarrow (xy)$ and $s \leftarrow z$, which $\mathcal{B}$ cannot know their values from D. Note also that sk_{ch} and r_{ch}^* is kept secret by $\mathcal{B}$.

Phase 1. Algorithm $\mathcal{A}$ adaptively issues two kinds of queries.

Secret Key Queries: Secret key query for a LSSS-policy (M, ρ) for which the challenge set $\mathcal{S}^*$ is not authorized. Suppose that $M \in \mathbb{Z}_p^{l \times n}$ and $\rho : [l] \to \left[0, \frac{p-1}{2}\right]$. According to *security requirement* of LSSS that has previously shown in Sec. 2.3, algorithm $\mathcal{B}$ can use linear algebra to generate a vector $\vec{\omega} = (\omega_1, \omega_2, \cdots, \omega_n)^{\mathrm{T}} \in \mathbb{Z}_p^n$ satisfying that $\omega_1 = 1$ and $\langle \vec{M_i}, \vec{\omega} \rangle = 0$ for all $i \in [l]$ such that $\rho(i) \in \mathcal{S}^*$. Then, algorithm $\mathcal{B}$ randomly chooses $\widetilde{y}_2, \widetilde{y}_3, \cdots, \widetilde{y}_n \overset{R}{\leftarrow} \mathbb{Z}_p$ and implicitly sets $\vec{y} = xy\vec{\omega} + (0, \widetilde{y}_2, \widetilde{y}_3, \cdots, \widetilde{y}_n)^{\mathrm{T}} = (xy, \widetilde{y}_2 + \omega_2, \widetilde{y}_3 + \omega_3, \cdots, \widetilde{y}_n + \omega_n)^{\mathrm{T}}$. For all $i \in [l]$, we have the following two cases:

1. $i \in [l]$ such that $\rho(i) \in \mathcal{S}^*$. In this case, we have that $\omega_1 = 1$ and $\langle \vec{M_i}, \vec{\omega} \rangle = 0$. Therefore, the share λ_i is $\lambda_i = \langle \vec{M_i}, \vec{y} \rangle = \langle \vec{M_i}, xy\vec{\omega} + (0, \widetilde{y}_2, \widetilde{y}_3, \cdots, \widetilde{y}_n)^{\mathrm{T}} \rangle = xy \cdot 0 + \langle \vec{M_i}, (0, \widetilde{y}_2, \cdots, \widetilde{y}_n)^{\mathrm{T}} \rangle$, where we set $\widetilde{\lambda}_i = \langle \vec{M_i}, (0, \widetilde{y}_2, \cdots, \widetilde{y}_n)^{\mathrm{T}} \rangle$ that can be calculated by $\mathcal{B}$. Algorithm $\mathcal{B}$ can finally pick a random exponent $\widetilde{t}_i \overset{R}{\leftarrow} \mathbb{Z}_p$, and outputs $K_{i,0} = g^{\widetilde{\lambda}_i} w^{\widetilde{t}_i}, K_{i,1} = \left(h \cdot u^{\rho(i)}\right)^{-\widetilde{t}_i}, K_{i,2} = g^{\widetilde{t}_i}$.
2. $i \in [l]$ such that $\rho(i) \notin \mathcal{S}^*$. In this case, we have that $\langle \vec{M_i}, \vec{\omega} \rangle \neq 0$. The share λ_i is also formed as $\lambda_i = \langle \vec{M_i}, \vec{y} \rangle = xy \cdot \langle \vec{M_i}, \vec{\omega} \rangle + \langle \vec{M_i}, (0, \widetilde{y}_2, \cdots, \widetilde{y}_n)^{\mathrm{T}} \rangle = xy \cdot \langle \vec{M_i}, \vec{\omega} \rangle + \widetilde{\lambda}_i$, where $xy \cdot \langle \vec{M_i}, \vec{\omega} \rangle \neq 0$. Algorithm $\mathcal{B}$ next picks a random $\widetilde{t}_i \overset{R}{\leftarrow} \mathbb{Z}_p$ to implicitly set t_i as

$$t_i = -y\langle \vec{M_i}, \vec{\omega} \rangle + \sum_{j \in [\kappa]} \frac{xzb_j \langle \vec{M_i}, \vec{\omega} \rangle}{\rho(i) - A_j^*} + \frac{xzb_{\kappa+1} \langle \vec{M_i}, \vec{\omega} \rangle}{\rho(i) - V^*} + \widetilde{t}_i$$

Since $\rho(i) \notin \mathcal{S}^*$, we have that $\rho(i) - A_j^* \neq 0$ for all $j \in [k]$. Also, $\rho(i) \in \left[0, \frac{p-1}{2}\right]$ while $V^* \in \left[\frac{p+1}{2}, p-1\right]$ so that $\rho(i) - V^* \neq 0$. Therefore, t_i can be

well-defined and it is properly distributed due to the randomness of $\widetilde{t}_i$. The first component $K_{i,0}$ can be calculated as

$$K_{i,0} = g^{\lambda_i} w^{t_i} = g^{xy \cdot W_i + \widetilde{\lambda}_i} \cdot w^{-y \cdot W_i + \left(\sum\limits_{j \in [\kappa]} \frac{xzb_j \cdot W_i}{\rho(i) - A_j^*} \right) + \frac{xzb_{\kappa+1} \cdot W_i}{\rho(i) - V^*} + \widetilde{t}_i}$$
$$= g^{\widetilde{\lambda}_i} \cdot w^{\widetilde{t}_i} \cdot \prod_{j \in [\kappa]} \left(g^{x^2 z b_j} \right)^{\frac{W_i}{\rho(i) - A_j^*}} \cdot \left(g^{x^2 z b_{\kappa+1}} \right)^{\frac{W_i}{\rho(i) - V^*}}$$

where we denote $W_i = \langle \overrightarrow{M}_i, \overrightarrow{\omega} \rangle$. Accordingly, $K_{i,1}$ can be calculated as

$$K_{i,1} = \left(u^{\rho(i)} h \right)^{-t_i} = \left(g^{\rho(i)\widetilde{u} + \widetilde{h}} \prod_{k \in [\kappa+1]} g^{\frac{\rho(i) y}{b_k^2} + \frac{xz}{b_k}} \prod_{k \in [\kappa]} \left(g^{\frac{y}{b_k^2}} \right)^{-A_k^*} \left(g^{\frac{y}{b_{\kappa+1}^2}} \right)^{-V^*} \right)^{yW_i}$$
$$\cdot \left(g^{\rho(i)\widetilde{u} + \widetilde{h}} \prod_{k \in [\kappa+1]} g^{\frac{\rho(i) y}{b_k^2} + \frac{xz}{b_k}} \prod_{k \in [\kappa]} \left(g^{\frac{y}{b_k^2}} \right)^{-A_k^*} \left(g^{\frac{y}{b_{\kappa+1}^2}} \right)^{-V^*} \right)^{-\sum\limits_{j \in [\kappa]} \frac{xzb_j \cdot W_i}{\rho(i) - A_j^*}}$$
$$\cdot \left(g^{\rho(i)\widetilde{u} + \widetilde{h}} \prod_{k \in [\kappa+1]} g^{\frac{\rho(i) y}{b_k^2} + \frac{xz}{b_k}} \prod_{j \in [\kappa]} \left(g^{\frac{y}{b_k^2}} \right)^{-A_k^*} \left(g^{\frac{y}{b_{\kappa+1}^2}} \right)^{-V^*} \right)^{-\frac{xzb_{\kappa+1} \cdot W_i}{\rho(i) - V^*} - \widetilde{t}_i}$$
$$= (g^y)^{W_i \left(\rho(i)\widetilde{u} + \widetilde{h} \right)} \prod_{k \in [\kappa]} \left(g^{\frac{y^2}{b_k^2}} \right)^{W_i \left(\rho(i) - A_k^* \right)} \left(g^{\frac{y^2}{b_{\kappa+1}^2}} \right)^{W_i \left(\rho(i) - V^* \right)}$$
$$\cdot \prod_{j \in [\kappa+1]} \left(g^{xzb_j} \right)^{\frac{-W_i \left(\rho(i)\widetilde{u} + \widetilde{h} \right)}{\rho(i) - A_j^*}} \left(g^{xzb_{\kappa+1}} \right)^{\frac{-W_i \cdot \left(\rho(i)\widetilde{u} + \widetilde{h} \right)}{\rho(i) - V^*}}$$
$$\cdot \prod_{\substack{j \in [\kappa] \\ k \in [\kappa+1]}} \left(g^{\frac{(xz)^2 b_j}{b_k}} \right)^{-\frac{W_i}{\rho(i) - A_j^*}} \prod_{k \in [\kappa+1]} \left(g^{(xz)^2 b_{\kappa+1}/b_k} \right)^{-\frac{W_i}{\rho(i) - V^*}} \left(u^{\rho(i)} h \right)^{-\widetilde{t}_i}$$
$$\cdot \left(\prod_{\substack{j,k \in [\kappa] \\ j \neq k}} \left(g^{\frac{xyzb_j}{b_k^2}} \right)^{\frac{\rho(i) - A_k^*}{\rho(i) - A_j^*}} \prod_{j \in [\kappa]} \left(g^{\frac{xyzb_j}{b_{\kappa+1}^2}} \right)^{\frac{\rho(i) - V^*}{\rho(i) - A_j^*}} \left(\prod_{k \in [\kappa]} g^{\frac{xyzb_{\kappa+1}}{b_k^2}} \right)^{\frac{\rho(i) - A_k^*}{\rho(i) - V^*}} \right)^{-W_i}$$

Similarly, the third component $K_{i,2}$ is formed as

$$K_{i,2} = g^{t_i} = g^{-y \cdot W_i + \left(\sum\limits_{j \in [\kappa]} \frac{xzb_j \cdot W_i}{\rho(i) - A_j^*} \right) + \frac{xzb_{\kappa+1} \cdot W_i}{\rho(i) - V^*} + \widetilde{t}_i}$$
$$= (g^y)^{-W_i} \cdot \prod_{j \in [\kappa]} \left(g^{xzb_j} \right)^{\frac{W_i}{\rho(i) - A_j^*}} \cdot \left(g^{xzb_{\kappa+1}} \right)^{\frac{W_i}{\rho(i) - V^*}} \cdot g^{\widetilde{t}_i}$$

Therefore, all components in $sk_{(M,\rho)}$ can be computed by $\mathcal{B}$. Hence, algorithm $\mathcal{B}$ can generate the secret key for the issued access structure (M, ρ) and correctly response to $\mathcal{A}$'s request.

Decryption Queries: Decryption query for the ciphertext $ct_{\mathcal{S}}$ associated with an attribute set $\mathcal{S} = \{A_1, \cdots, A_{|\mathcal{S}|}\}$. Algorithm $\mathcal{B}$ first computes

$$V = \mathbf{Hash_{ch}}\left(pk_{ch}, pk_{ch} \| C \| C_0 \| C_{1,1} \| \cdots \| C_{|\mathcal{S}|,1}, r_{ch}\right)$$

and determines whether the ciphertext is valid by checking Equation (1) for all $i \in [|\mathcal{S}|]$ and Equation (2). If one of the equalities does not hold, the ciphertext is invalid and $\mathcal{B}$ returns with $\perp$. Otherwise:

1. $\mathcal{S} \nsubseteq \mathcal{S}^*$. In this case, algorithm $\mathcal{B}$ can construct an access structure (M, ρ) such that $\mathcal{S}$ satisfies (M, ρ) but $\mathcal{S}^*$ does not (For example, an access structure $(A_1 \wedge A_2 \wedge \cdots \wedge A_{|\mathcal{S}|})$). Since $\mathcal{S}^*$ cannot be authorized by (M, ρ), algorithm $\mathcal{B}$ can run the same algorithm described in secret key query phase to generate a well-formed secret key $sk_{(M,\rho)}$ for that access structure and decrypt by running **Decrypt**$(pp, sk_{(M,\rho)}, ct_{\mathcal{S}})$.
2. $\mathcal{S} \subseteq \mathcal{S}^*$ and $V \neq V^*$. Algorithm $\mathcal{B}$ is unable to construct any secret keys for an access structure (M, p) for which $\mathcal{S}^*$ is not authorized. However, note that the ciphertext is additional encrypted by a on-the-fly dummy attribute V. When $V \neq V^*$, algorithm can generate a secret key for the access structure $(M = (1), \rho(1) = V)$. This secret key is indeed an invalid secret key in the actual system, since $V \notin \left[0, \frac{p-1}{2}\right]$ so that V is not in the attribute universe $\mathcal{U}$. However, algorithm $\mathcal{B}$ can use such a key to decrypt the issued ciphertext. Note that $\mathcal{A}$ cannot distinguish which key $\mathcal{B}$ uses to decrypt. Hence, the decryption result is a well-formed message.
3. $\mathcal{S} \subseteq \mathcal{S}^*$ and $V = V^*$. In this case, algorithm $\mathcal{B}$ is unable to respond. It terminates the simulation with $\mathcal{A}$, outputs a random bit $b \in \{0, 1\}$ and halts. Since the Chameleon hash function $\mathcal{B}$ employs satisfies the properties of *collision resistance* and *uniformity* for anyone without sk_{ch}, this case happens with negligible probability $1/|\mathcal{V}| = 2/(p-1)$.

Challenge. Algorithm $\mathcal{A}$ submits two equal-length messages $m_0, m_1 \in \mathbb{G}_T$ to $\mathcal{B}$. Algorithm $\mathcal{B}$ flips a random coin $b \in \{0, 1\}$. The first component C_0^* of the challenge ciphertext is previously computed as $C_0^* = g^s = g^z$ in **Setup** phase, so is $C_{i,1}^* = g^{r_i} = g^{b_i}$ for each $i \in [\kappa]$. Algorithm $\mathcal{B}$ sets

$$C_{i,2}^* = \left(h \cdot u^{A_i^*}\right)^{r_i} w^{-s} = g^{b_i\left(\widetilde{u}A_i^* + \widetilde{h}\right)} \prod_{j \in [\kappa+1]} g^{\frac{xzb_i}{b_j}} \prod_{j \in [\kappa]} g^{\frac{yb_i\left(A_i^* - A_j^*\right)}{b_j^2}} \cdot g^{\frac{yb_i\left(A_i^* - V^*\right)}{b_{\kappa+1}^2}}$$

$$= \left(g^{b_i}\right)^{\widetilde{u}A_i^* + \widetilde{h}} \cdot \prod_{\substack{j \in [\kappa+1] \\ j \neq i}} g^{\frac{xzb_i}{b_j}} \prod_{\substack{j \in [\kappa] \\ j \neq i}} g^{\frac{yb_i\left(A_i^* - A_j^*\right)}{b_j^2}} \cdot g^{\frac{yb_i\left(A_i^* - V^*\right)}{b_{\kappa+1}^2}}$$

for each $i \in [\kappa]$. For the on-the-fly dummy attribute V^*, algorithm $\mathcal{B}$ sets

$$C_{0,1}^* = g^{b_{\kappa+1}}, C_{0,2}^* = \left(g^{b_{\kappa+1}}\right)^{\widetilde{u}V^* + \widetilde{h}} \cdot \prod_{j \in [\kappa]} g^{\frac{xzb_{\kappa+1}}{b_j}} \prod_{j \in [\kappa]} g^{\frac{yb_{\kappa+1}\left(V^* - A_j^*\right)}{b_j^2}}$$

To keep the on-the-fly dummy attribute V^* unchanged, algorithm $\mathcal{B}$ sets $C_b^* = m_b \cdot T$ and runs $r_b^* \leftarrow \mathbf{UForge_{ch}}(sk_{ch}, C^*, r_{ch}^*, pk_{ch} \| C_b^* \| C_0^* \| C_{0,1}^* \| C_{1,1}^* \| \cdots \| C_{\kappa,1}^*)$. Note that the functionality of $\mathbf{UForge_{ch}}$ ensures

$$V^* = \mathbf{Hash_{ch}}(pk_{ch}, pk_{ch} \| C_b^* \| C_0^* \| C_{0,1}^* \| C_{1,1}^* \| C_{2,1}^* \| \cdots \| C_{\kappa,1}^*, r_b^*)$$

Algorithm $\mathcal{B}$ responses $ct^* = \left(\mathcal{S}^*, r_b^*, C_b^*, C_0^*, C_{0,1}^*, C_{0,2}^*, \left\{C_{i,1}^*, C_{i,2}^*\right\}_{i\in[\kappa]}\right)$ to $\mathcal{A}$.

Phase 2. Algorithm $\mathcal{B}$ processes as in **Phase 1** to response two kinds of queries issued from $\mathcal{A}$.

Guess. Finally, adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. Algorithm $\mathcal{B}$ also outputs b' to answer the $(q+1)$-wDBDH problem. If $T = e(g,g)^{xyz} = e(g,g)^{\alpha s}$, then $\mathcal{B}$ plays the proper security game with $\mathcal{A}$ since the ct^* is a valid ciphertext for the message m_b. In this case, the advantage of algorithm $\mathcal{A}$ is ϵ. On the other hand, if $T \overset{R}{\leftarrow} \mathbb{G}_T$, then ct^* is a ciphertext for the random message chosen in $\mathbb{G}_T$ so that the advantage of $\mathcal{A}$ is exactly 0. Therefore, if $\mathcal{A}$ has advantage $Adv_{\mathcal{A}}^{\text{KP-ABE}}(\lambda) = \epsilon$ in breaking our KP-ABE scheme, then $\mathcal{B}$ can determine the distribution of T with advantage $Adv_{\mathcal{B}}(\lambda) \geq \epsilon - \frac{2}{p-1}$. □

5 Conclusion

We proposed a direct construction of fully functional CCA2-secure KP-ABE scheme. Compared with the underlying CPA-secure one, our scheme only introduces the cost of a Chameleon hash. Furthermore, our scheme allows public ciphertext validity test. Technically, in contrast existing Chameleon hash applications to signatures, our construction illustrates novel applications of Chameleon hashes in construction and security proofs of encryption schemes.

Acknowledgments and Disclaimer. We appreciate the anonymous reviewers for their valuable suggestions. We especially thank Willy Susilo for his many helps on preparing the final version of the paper. Dr. Bo Qin is the corresponding author. This paper is partially supported by the National Key Basic Research Program (973 program) through project 2012CB315905, by the Natural Science Foundation through projects 61272501, 61173154, 61370190 and 61003214, by the Fundamental Research Funds for the Central Universities, and the Research Funds(No. 14XNLF02) of Renmin University of China, by the Open Research Fund of Beijing Key Laboratory of Trusted Computing and by the Open Research Fund of The Academy of Satellite Application.

References

1. Ateniese, G., de Medeiros, B.: On the key exposure problem in Chameleon hashes. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 165–179. Springer, Heidelberg (2005)

2. Beimel, A.: Secure schemes for secret sharing and key distribution. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)
3. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE S&P 2007, pp. 321–334. IEEE Press, USA (2007)
4. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
5. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
6. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: ACM CCS 2005, pp. 320–329. ACM Press, New York (2005)
7. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
8. Chase, M., Chow, S.S.M.: Improving privacy and security in multi-authority attribute-based encryption. In: ACM CCS 2009, pp. 121–130. ACM Press, New York (2009)
9. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007)
10. Chen, X., Zhang, F., Kim, K.: Chameleon hashing without key exposure. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 87–98. Springer, Heidelberg (2004)
11. Chen, X., Zhang, F., Susilo, W., Mu, Y.: Efficient generic on-line/off-line signatures without key exposure. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 18–30. Springer, Heidelberg (2007)
12. Chen, X., Zhang, F., Susilo, W., Tian, H., Li, J., Kim, K.: Identity-based chameleon hash scheme without key exposure. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 200–215. Springer, Heidelberg (2010)
13. Chen, X., Zhang, F., Tian, H., Wei, B., Kim, K.: Discrete logarithm based Chameleon hashing and signatures without key exposure. Computers and Electrical Engineering 37(4), 614–623 (2011)
14. Chen, X., Zhang, F., Tian, H., Wei, B., Susilo, W., Mu, Y., Lee, H., Kim, K.: Efficient generic on-line/off-line (threshold) signatures without key exposure. Information Sciences 178(21), 4192–4203 (2008)
15. Chen, C., Zhang, Z., Feng, D.: Efficient ciphertext policy attribute-based encryption with constant-size ciphertext and constant computation-cost. In: Boyen, X., Chen, X. (eds.) ProvSec 2011. LNCS, vol. 6980, pp. 84–101. Springer, Heidelberg (2011)
16. Cheung, L., Newport, C.: Provably secure ciphertext policy ABE. In: ACM CCS 2007, pp. 456–465. ACM Press, New York (2007)
17. Deng, H., Wu, Q., Qin, B., Chow, S.S., Domingo-Ferrer, J., Shi, W.: Tracing and revoking leaked credentials: Accountability in leaking sensitive outsourced data. In: ACM ASIACCS 2014, pp. 425–443. ACM Press, New York (2014)
18. Deng, H., Wu, Q., Qin, B., Mao, J., Liu, X., Zhang, L., Shi, W.: Who is touching my cloud? ESORICS 2014, To Appear (2014)
19. Deng, H., Wu, Q., Qin, B., Domingo-Ferrer, J., Zhang, L., Liu, J., Shi, W.: Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts. Information Sciences 275, 370–384 (2014)

20. Ge, A.J., Zhang, R., Chen, C., Ma, C.G., Zhang, Z.F.: Threshold ciphertext policy attribute-based encryption with constant size ciphertexts. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 2012. LNCS, vol. 7372, pp. 336–349. Springer, Heidelberg (2012)
21. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM CCS 2006, pp. 89–98. ACM Press, New York (2006)
22. Krawczyk, H., Rabin, T.: Chameleon hashing and signatures. In: NDSS 2000, pp. 143–154. The Internet Society, San Diego (2000)
23. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
24. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (2011)
25. Lewko, A., Waters, B.: Unbounded HIBE and attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 547–567. Springer, Heidelberg (2011)
26. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: ACM CCS 2007, pp. 195–203. ACM Press, New York (2007)
27. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
28. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 349–366. Springer, Heidelberg (2012)
29. Qin, B., Wang, H., Wu, Q., Liu, J., Domingo-Ferrer, D.: Simultaneous authentication and secrecy in identity-based data upload to cloud. Cluster Computing 16(4), 845–859 (2013)
30. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: ACM CCS 2013, pp. 463–474. ACM Press, New York (2013)
31. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
32. Wang, Y., Wu, Q., Wong, D.S., Qin, Q., Chow, S.S.M., Liu, Z., Tan, X.: Offloading provable data Possession by securely outsourcing exponentiations in single untrusted program model. ESORICS 2014, To Appear (2014)
33. Yamada, S., Attrapadung, N., Hanaoka, G., Kunihiro, N.: Generic constructions for chosen-ciphertext secure attribute based encryption. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 71–89. Springer, Heidelberg (2011)