The preliminary version of this paper appeared in *Proceedings of The 1st IEEE International Workshop on Big Data and IoT Security in Smart Computing - IEEE BITS 2017* under the same title. This is the full version.

Short CCA-Secure Ciphertext-Policy Attribute-Based Encryption ⋆

Hiroaki Anada[1] and Seiko Arita[2]

[1] Department of Information Security, University of Nagasaki
W408, 1-1-1, Manabino, Nagayo-cho, Nishisonogi-gun, Nagasaki, 851-2195 JAPAN
anada@sun.ac.jp
[2] Institute of Information Security
509, 2-14-1, Tsuruya-cho, Kanagawa-ku, Yokohama, 221-0835 JAPAN
arita@iisec.ac.jp

May 26, 2017

Abstract. We propose a technique of individually modifying an attribute-based encryption scheme (ABE) that is secure against chosen-plaintext attacks (CPA) into an ABE scheme that is secure against chosen-ciphertext attacks (CCA) in the standard model. We demonstrate the technique in the case of the Waters ciphertext-policy ABE (CP-ABE). Our technique is helpful when a Diffie-Hellman tuple to be verified is in the terminal group of a bilinear map. We utilize the Twin Diffie-Hellman Trapdoor Test of Cash, Kiltz and Shoup, and it results in expansion of secret key length and decryption cost of computation by a factor of four, whereas public key length, ciphertext length and encryption cost of computation remain almost the same. In the case that the size of attribute sets are small, those lengths and costs are smaller than those of the CP-ABE obtained via the generic transformation of Yamada et al. in PKC 2011.
Keywords: public-key cryptography, attribute-based encryption, direct chosen-ciphertext security, twin Diffie-Hellman.

1 Introduction

Attribute-based encryption (ABE) was first proposed by Sahai and Waters [SW05] to realize fine-grained access control by encryption, where attributes mean authorized credentials. In ciphertext-policy ABE (CP-ABE) introduced by the subsequent work of Goyal, Pandey, Sahai and Waters [GPSW06], ciphertexts are associated with access policies over attributes, while secret keys are associated with sets of attributes. A secret key works to decrypt a ciphertext if and only if the associated set of attributes satisfies the associated access policy. Since the proposal, it has been studied to attain certain properties such as indistinguishability against chosen-plaintext attacks (IND-CPA) in the standard model [Wat11] and adaptive security against adversary's choice of a target access structure [LOS$^+$10].

In this paper, we work through a problem of constructing a shorter ABE scheme that attains indistinguishability against chosen-ciphertext attacks (IND-CCA) in the standard model. Here CCA means that an adversary can collects decryption results of ciphertexts of its choise through attacking.

⋆ This work is partially supported by kakenhi Grant-in-Aid for Scientific Research (C) JP15K00029 from Japan Society for the Promotion of Science.

Let us recall the case of identity-based encryption (IBE). The CHK transformation of Canetti, Halevi and Katz [CHK04] is a generic tool for obtaining IND-CCA secure IBE scheme. It transforms any hierarchical IBE (HIBE) scheme that is selective-ID IND-CPA secure into an IBE scheme that is adaptive-ID IND-CCA secure. A point of the CHK transformation is that it introduces a dummy identity `vk` that is a verification key of a one-time signature. Then a ciphertext is attached with `vk` and a signature σ. which is generated each time one executes encryption. In contrast, direct chosen-ciphertext security technique for IBE of Boyen, Mei and Waters [BMW05] is an individual technique for obtaining an IND-CCA secure IBE scheme. It converts a HIBE scheme that is adaptive-ID IND-CPA secure into an IBE scheme that is adaptive-ID IND-CCA secure. Though the technique needs to treat each scheme individually, the obtained scheme attains better performance than that obtained by the generic tool (the CHK transformation). Let us transfer into the case of ABE. The transformation of Yamada et al. [YAHK11] is a generic tool for obtaining IND-CCA secure ABE scheme. It transforms any ABE scheme (with delegatability or verifiability) that is IND-CPA secure into an ABE scheme that is IND-CCA secure. A point of their transformation is, similar to the case of IBE, that it introduces a dummy attribute `vk` that is a verification key of a one-time signature. Then a ciphertext is attached with `vk` and a signature σ. Notice here that developing direct chosen-ciphertext security technique for ABE (in the standard model) is a missing piece. One of the reason seems that there is an obstacle that a Diffie Hellman tuple to be verified is in the terminal group of a bilinear map. In that situation, the bilinear map looks of no use.

1.1 Our Contribution

A first contribution is that we fill in the missing piece of direct chosen-ciphertext security for ABE. We develop a technique and apply it to the Waters CP-ABE scheme [Wat11] to obtain IND-CCA security. A second technical contribution is as follows. To overcome the above obstacle, we employ and apply the Twin Diffie-Hellman Trapdoor Test of Cash, Kiltz and Shoup [CKS08]. In addition to that, we also utilize the algebraic trick of Boneh and Boyen [BB04] and Kiltz [Kil06] to reply for adversary's decryption query. In total, we develop the technique to realize direct chosen-ciphertext security.

1.2 Related Works

Waters [Wat11] pointed out that IND-CCA security would be attained by the CHK transformation. Gorantla, Boyd and Nieto [GBN10] constructed a IND-CCA secure CP-ABKEM in the random oracle model. Yamada et al. [YAHK11] proposed a generic transformation of a IND-CPA secure ABE scheme into a IND-CCA secure ABE scheme. Their transformation is considered to be an ABE-version and versatile. Especially, it can be applied to non-pairing-based scheme.

The Waters CP-ABE [Wat11] can be captured as a CP-ABKEM: the blinding factor can be considered as a random one-time key. In addition, the CP-ABKEM is IND-CPA secure because the Waters CP-ABE is proved to be IND-CPA secure. For theoretical simplicity, we will provide *a scheme of KEM first, and then an encryption scheme*. We modify the Waters CP-ABKEM, which is IND-CPA secure, into a KEM which is IND-CCA secure.

It should be noted that, in *key-policy ABE (KP-ABE)* [GPSW06] where ciphertexts are associated with sets of attributes while secret keys are associated with access policies over attributes, there is a remarkable work of a KP-ABE scheme with constant-size ciphertexts [ALdP11]. On the other hand, in CP-ABE schemes there are several works that attain the property of constant-size ciphertexts [HLR10,CZF11,GZC$^+$12], but (to the best of the authors' knowledge) those schemes can treat only limited classes of access structures such as the threshold type.

1.3 Efficiency Comparison

We compare efficiency of our `CP-ABKEM` to the original Waters $\texttt{CP-ABKEM}_{\mathrm{cpa}}$. We also compare efficiency of the CP-ABKEM obtained by the generic transformation of Yamada et al. [YAHK11]. Here

Table 1. Efficiency Comparison of IND-sel-CCA secure CP-ABKEMs obtained from Waters $\texttt{CP-ABKEM}_{\mathrm{cpa}}$.

Scheme	L(PK)	L(SK_S)	L(CT)	C(Encap)	C(Decap)
Generic transform of Yamada et al. [YAHK11]	$+2\lambda^2(\mathbb{G})$	$+2\lambda^2(\mathbb{G})$	$+3\lambda^2$(bit)	$+2\lambda^2$exp.($\mathbb{G}$)	$+2\lambda^2$pair.(e)
Our individual modification (CP-ABKEM)	$+3(\mathbb{G}_T)$	$\times 4$	$+2(\mathbb{G}_T)$	$+4$exp.($\mathbb{G}_T$)	$\times 4$

1) L(data) denotes length of data, C(algorithm) denotes computational amount of algorithm.
2) $+$ and $\times$ mean increment and multiplier to the length or computational amount of the Waters $\texttt{CP-ABKEM}_{\mathrm{cpa}}$.
3) ($\mathbb{G}$), ($\mathbb{G}_T$) and (bit) mean elements in $\mathbb{G}$, elements in $\mathbb{G}_T$ and bits, respectively.
4) exp.($\mathbb{G}$) and pair.(e) mean a computational amount of one exponentiation in $\mathbb{G}$ and one pairing computation by the map e, respectively.

the generic transformation [YAHK11] is considered in the setting of small attribute universe [GPSW06], delegation case and the Lamport one-time signature case. Table 1 shows these comparison. Our individual technique results in expansion of secret key length and decryption cost of computation by a factor of four, while public key length, ciphertext length and encryption cost of computation are almost the same as those of the Waters. In the case that the size of attribute sets are up to the square of the security parameter λ, lengths and costs of our CP-ABKEM are smaller than those of the CP-ABE obtained via the generic transformation of Yamada et al. [YAHK11].

1.4 Organization of the Paper

In Section 2, we survey concepts, definitions and techniques needed. In Section 3, we construct a CP-ABKEM from the Waters CP-ABKEM [Wat11] and provide a proof that it attains the IND-sel-CCA security based on the IND-sel-CPA security of the Waters CP-ABKEM. In Section 1.3, we compare efficiency of our CP-ABKEM and CP-ABE with the ones obtained by the generic transformation of Yamada et al.[YAHK11] to the Waters CP-ABKEM. In Section 4, we conclude our works and list up future works to be studied.

2 Preliminaries

The security parameter is denoted λ. A prime of bit length λ is denoted p. A multiplicative cyclic group of order p is denoted $\mathbb{G}$. The ring of exponent domain of $\mathbb{G}$, which consists of integers from 0 to $p-1$ with modulo p operation, is denoted $\mathbb{Z}_p$.

2.1 Bilinear Map

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$ and e be a bilinear map, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The bilinear map e has the following properties:
1. Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq \mathrm{id}_{\mathbb{G}_T}$ (: the identity element of the group $\mathbb{G}_T$).

Parameters of a bilinear map are generated by a probabilistic polynomial time (PPT) algorithm Grp on input λ: $(p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \texttt{Grp}(\lambda)$.

Hereafter we assume that the group operation in $\mathbb{G}$ and $\mathbb{G}_T$ and the bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ are computable in PT in λ.

2.2 Access Structure

Let $\mathcal{U} = \{\chi_1, \ldots, \chi_u\}$ be a set of attributes, or simply set $\mathcal{U} = \{1, \ldots, u\}$ by numbering. An *access structure*, which corresponds to an access policy, is defined as a collection $\mathbb{A}$ of non-empty subsets of $\mathcal{U}$; that is, $\mathbb{A} \subset 2^{\mathcal{U}} \backslash \{\phi\}$. An access structure $\mathbb{A}$ is called *monotone* if for any $B \in \mathbb{A}$ and $B \subset C$, $C \in \mathbb{A}$ holds. The sets in $\mathbb{A}$ are called authorized sets, and the sets not in $\mathbb{A}$ are called unauthorized sets. We will consider in this paper only monotone access structures.

2.3 Linear Secret-Sharing Scheme

We only describe a linear secret-sharing scheme (LSSS) in our context of attribute-based schemes. A secret-sharing scheme Π over the attribute universe $\mathcal{U}$ is called linear over $\mathbb{Z}_p$ if:
1. The shares for each attribute form a vector over $\mathbb{Z}_p$,
2. There exists a matrix M of size $l \times n$ called the share-generating matrix for Π and a function ρ which maps each row index i of M to an attribute in $\mathcal{U} = \{1, \ldots, u\}$: $\rho : \{1, ..., l\} \to \mathcal{U}$.

To make shares, we first choose a random vector $\boldsymbol{v} = (s, y_2, \ldots, y_n) \in \mathbb{Z}_p^n$: s is a secret to be shared. For $i = 1$ to l, we calculate each share $\lambda_i = \boldsymbol{v} \cdot M_i$, where M_i denotes the i-th row vector of M and $\cdot$ denotes the formal inner product. LSSS $\Pi = (M, \rho)$ defines an access structure $\mathbb{A}$ through ρ.

Suppose that an attribute set S satisfies $\mathbb{A}$ ($S \in \mathbb{A}$) and let $I_S = \rho^{-1}(S) \subset \{1, \ldots, l\}$. Then, let $\{\omega_i \in \mathbb{Z}_p; i \in I_S\}$ be a set of constants (*linear reconstruction constants*) such that if $\{\lambda_i \in \mathbb{Z}_p; i \in I_S\}$ are valid shares of a secret s according to M, then $\sum_{i \in I_S} \omega_i \lambda_i = s$. It is khown that these constants $\{\omega_i\}_{i \in I_S}$ can be found in time polynomial in l: the row size of the share-generating matrix M. If S does not satisfy $\mathbb{A}$ ($S \notin \mathbb{A}$), then no such constants $\{\omega_i\}_{i \in I_S}$ exist.

2.4 Ciphertext-Policy Attribute-Based Key Encapsulation Mechanism

A ciphertext-policy attribute-based key encapsulation mechanism (CP-ABKEM) consists of four PPT algorithms (Setup, Encap, Keygen, Decap)[3].

Setup($\lambda, \mathcal{U}$). A setup algorithm Setup takes as input the security parameter λ and the attribute universe $\mathcal{U} = \{1, \ldots, u\}$. It returns a public key PK and a master secret key MSK.

Encap(PK, $\mathbb{A}$). An encapsulation algorithm Encap takes as input the public key PK and an access structure $\mathbb{A}$. It returns a random string κ and its encapsulation ψ.

KeyGen(PK, MSK, S). A key generation algorithm KeyGen takes as input the public key PK, the master secret key MSK and an attribute set S. It returns a secret key SK_S corresponding to S.

Decap(PK, SK_S, ψ). A decapsulation algorithm Decap takes as input the public key PK, an encapsulation (we also call it a ciphertext according to context) ψ and a secret key SK_S. It first checks whether $S \in \mathbb{A}$, where S and $\mathbb{A}$ are contained in SK_S and ψ, respectively. If the check result is False, it puts $\hat{\kappa} = \perp$. It returns a decapsulation result $\hat{\kappa}$.

Chosen-Ciphertext Attack on CP-ABKEM. According to previous works (for example, see [GBN10]), the chosen-ciphertext attack on a CP-ABKEM is formally defined as the indistinguishability game (IND-CCA game), that is described as the following experiment of an adversary $\mathcal{A}$.

$$
\begin{aligned}
&\mathbf{Experiment}^{\text{ind-cca}}_{\mathcal{A},\texttt{CP-ABKEM}}(\lambda, \mathcal{U}) \\
&\quad (\mathrm{PK}, \mathrm{MSK}) \leftarrow \mathbf{Setup}(\lambda, \mathcal{U}) \\
&\quad \mathbb{A}^* \leftarrow \mathcal{A}^{\mathbf{KeyGen}(\mathrm{PK},\mathrm{MSK},\cdot),\mathbf{Decap}(\mathrm{PK},\mathrm{SK}_\cdot,\cdot)}(\mathrm{PK}, \mathcal{U}) \\
&\quad (\kappa^*, \psi^*) \leftarrow \mathbf{Encap}(\mathrm{PK}, \mathbb{A}^*), \kappa \leftarrow \mathit{KeySp}(\lambda), b \leftarrow \{0, 1\} \\
&\quad \text{If } b = 1 \text{ then } \tilde{\kappa} = \kappa^* \text{ else } \tilde{\kappa} = \kappa \\
&\quad b' \leftarrow \mathcal{A}^{\mathbf{KeyGen}(\mathrm{PK},\mathrm{MSK},\cdot),\mathbf{Decap}(\mathrm{PK},\mathrm{SK}_\cdot,\cdot)}(\tilde{\kappa}, \psi^*) \\
&\quad \text{If } b' = b \text{ then return Win else return Lose.}
\end{aligned}
$$

[3] In Gorantla, Boyd and Nieto [GBN10], they say *encapsulation-policy* attribute-based-KEM (EP-AB-KEM) instead of saying ciphertext-policy attribute-based KEM here.

In the above experiment, two kinds of queries are issued by $\mathcal{A}$.

One is key-extraction queries. Indicating an attribute set S_i, $\mathcal{A}$ queries its key-extraction oracle $\mathrm{KeyGen}(\mathrm{PK}, \mathrm{MSK}, \cdot)$ for the secret key SK_{S_i}. Here we do not require any input attribute sets S_{i_1} and S_{i_2} to be distinct.

Another is decapsulation queries. Indicating a pair (S_j, ψ_j) of an attribute set and an encapsulation, $\mathcal{A}$ queries its decapsulation oracle $\mathrm{Decap}(\mathrm{PK}, \mathrm{SK}_{\cdot}, \cdot)$ for the decapsulation result $\hat{\kappa}_j$. Here an access structure $\mathbb{A}_j$, which is used to generate an encapsulation ψ_j, is implicitely included in ψ_j. In the case that $S \notin \mathbb{A}$, $\hat{\kappa}_j = \perp$ is replied to $\mathcal{A}$.

Both kinds of queries are at most q_{k} and q_{d} times in total, respectively, which are polynomial in λ.

The access structure $\mathbb{A}^*$ decrared by $\mathcal{A}$ is called a *target access structure*. Two restrictions are imposed on $\mathcal{A}$ concerning $\mathbb{A}^*$. In key-extraction queries, each attribute set S_i must satisfy $S_i \notin \mathbb{A}^*$. In decapsulation queries, each pair (S_j, ψ_j) must satisfy $S_j \notin \mathbb{A}^*$ in the phase before the declaration of $\mathbb{A}^*$ and each pair (S_j, ψ_j) must satisfy $S_j \notin \mathbb{A}^* \vee \psi_j \neq \psi^*$ in the phase after the declaration of $\mathbb{A}^*$.

The *advantage* of the adversary $\mathcal{A}$ over `CP-ABKEM` in the IND-CCA game is defined as the following probability:

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{A},\texttt{CP-ABKEM}}(\lambda, \mathcal{U}) = \Pr[\mathbf{Experiment}^{\text{ind-cca}}_{\mathcal{A},\texttt{CP-ABKEM}}(\lambda, \mathcal{U}) \text{ returns } \textsc{Win}].$$

`CP-ABKEM` is called *secure against chosen-ciphertext attacks* if, for any PPT adversary $\mathcal{A}$ and for any attribute universe $\mathcal{U}$ [4], $\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{A},\texttt{CP-ABKEM}}(\lambda, \mathcal{U})$ is negligible in λ.

In the *selective game on a target access structure* (IND-sel-CCA game), the adversary $\mathcal{A}$ declares a target access structure $\mathbb{A}^*$ *before* $\mathcal{A}$ receives a public key PK, which is defined as the following experiment.

$\mathbf{Experiment}^{\text{ind-sel-cca}}_{\mathcal{A},\texttt{CP-ABKEM}}(\lambda, \mathcal{U})$
$\mathbb{A}^* \leftarrow \mathcal{A}(\lambda, \mathcal{U}), (\mathrm{PK}, \mathrm{MSK}) \leftarrow \mathbf{Setup}(\lambda, \mathcal{U})$
$\epsilon \leftarrow \mathcal{A}^{\mathbf{KeyGen}(\mathrm{PK},\mathrm{MSK},\cdot),\mathbf{Decap}(\mathrm{PK},\mathrm{SK}_{\cdot},\cdot)}(\mathrm{PK})$
$(\kappa^*, \psi^*) \leftarrow \mathbf{Encap}(\mathrm{PK}, \mathbb{A}^*), \kappa \leftarrow \mathit{KeySp}(\lambda), b \leftarrow \{0, 1\}$
If $b = 1$ then $\tilde{\kappa} = \kappa^*$ else $\tilde{\kappa} = \kappa$
$b' \leftarrow \mathcal{A}^{\mathbf{KeyGen}(\mathrm{PK},\mathrm{MSK},\cdot),\mathbf{Decap}(\mathrm{PK},\mathrm{SK}_{\cdot},\cdot)}(\tilde{\kappa}, \psi^*)$
If $b' = b$ then return $\textsc{Win}$ else return $\textsc{Lose}$.

In the indistinguishability game against *chosen-plaintext attack* (IND-CPA game), the adversary $\mathcal{A}$ issues no decapsulation query (that is, $q_{\mathrm{d}} = 0$).

The *advantage* $\mathbf{Adv}^{\text{game}}_{\mathcal{A},\text{scheme}}(\lambda, \mathcal{U})$ of the adversary $\mathcal{A}$ over a scheme in a game is defined in the same way as above.

Ciphertext-Policy Attribute-Based Encryption Scheme. In the case of a ciphertext-policy attribute-based encryption scheme (CP-ABE), $\mathrm{Encap}(\mathrm{PK}, \mathbb{A})$ and $\mathrm{Decap}(\mathrm{PK}, \mathrm{SK}_S, \psi)$ are replaced by PPT algorithms $\mathrm{Encrypt}(\mathrm{PK}, \mathbb{A}, m)$ and $\mathrm{Decrypt}(\mathrm{PK}, \mathrm{SK}_S, \mathrm{CT})$, respectively, where m and CT mean a message and a ciphertext, respectively.

The IND-CCA game for CP-ABE is defined in the same way as for CP-ABKEM above, except the following difference. In Challenge phase, the adversary $\mathcal{A}$ submits two equal length messages (plaintexts) m_0 and m_1. Then the challenger flips a coin $b \in \{0, 1\}$ and gives an encryption result CT of m_b to $\mathcal{A}$. In Guess phase, the adversary $\mathcal{A}$ returns $b' \in \{0, 1\}$. If $b' = b$, then $\mathcal{A}$ wins in the IND-CCA game. Otherwise, $\mathcal{A}$ loses.

[4] We must distinguish the two cases; the case that $\mathcal{U}$ is small (i.e. $|\mathcal{U}| = u$ is bounded by some polynomial of λ) and the case that $\mathcal{U}$ is large (i.e. u is not necessarily bounded by a polynomial of λ). We assume the *small case* unless we state the large case explicitely.

2.5 The Twin Diffie-Hellman Technique

A 6-tuple $(g, X_1, X_2, Y, Z_1, Z_2) \in \mathbb{G}^6$ is called a *twin Diffie-Hellman tuple* if the tuple is written as $(g, g^{x_1}, g^{x_2}, g^y, g^{x_1 y}, g^{x_2 y})$ for some elements x_1, x_2, y in $\mathbb{Z}_p$. In other words, a 6-tuple $(g, X_1, X_2, Y, Z_1, Z_2)$ is a twin Diffie-Hellman tuple (twin DH tuple, for short) if $Y = g^y$ and $Z_1 = X_1^y$ and $Z_2 = X_2^y$.

The following lemma of Cash, Kiltz and Shoup will be used in the security proof to decide whether a tuple is a twin DH tuple or not.

Lemma 1 (Cash, Kiltz and Shoup [CKS08] "Trapdoor Test")
Let X_1, r, s be mutually independent random variables, where X_1 takes values in $\mathbb{G}$, and each of r, s is uniformly distributed over $\mathbb{Z}_p$. Define the random variable $X_2 = X_1^{-r} g^s$. Suppose that $\hat{Y}, \hat{Z}_1, \hat{Z}_2$ are random variables taking values in $\mathbb{G}$, each of which is defined independently of r. Then the probability that the truth value of $\hat{Z}_1{}^r \hat{Z}_2 = \hat{Y}^s$ does not agree with the truth value of $(g, X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$ being a twin DH tuple is at most $1/p$. Moreover, if $(g, X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$ is a twin DH tuple, then $\hat{Z}_1{}^r \hat{Z}_2 = \hat{Y}^s$ certainly holds.

Note that Lemma 1 is a statistical property. Especially, Lemma 1 holds without any number theoretic assumption. To be precise, we consider the following experiment of an algorithm *Cheat* with *unbounded computational power (not limited to PPT)*, where *Cheat*, given a triple (g, X_1, X_2), tries to complete a 6-tuple $(g, X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$ which passes the "Trapdoor Test" but which is *not* a twin DH tuple.

$$
\begin{array}{l}
\mathbf{Experiment}^{\text{twinDH-test}}_{Cheat,\mathbb{G}}(\lambda) \\
\quad (g, X_1) \leftarrow \mathbb{G}^2, (r, s) \leftarrow \mathbb{Z}_p^2, X_2 = X_1^{-r} g^s \\
\quad \mathbb{G}^3 \ni (\hat{Y}, \hat{Z}_1, \hat{Z}_2) \leftarrow \mathit{Cheat}(g, X_1, X_2) \\
\quad \text{If } \hat{Z}_1{}^r \hat{Z}_2 = \hat{Y}^s \wedge (g, X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2) \text{ is NOT a twin DH tuple,} \\
\qquad \text{then return } \textsc{Win} \text{ else return } \textsc{Lose}
\end{array}
$$

Let us define the advantage of *Cheat* over $\mathbb{G}$ as follows.

$$\mathbf{Adv}^{\text{twinDH-test}}_{Cheat,\mathbb{G}}(\lambda) = \Pr[\mathbf{Experiment}^{\text{twinDH-test}}_{Cheat,\mathbb{G}}(\lambda) \text{ returns } \textsc{Win}].$$

Now we are ready to complement Lemma 1.

Lemma 2 (a Complement for Cash, Kiltz and Shoup [CKS08] "Trapdoor Test")
For any algorithm Cheat with unbounded computational power, $\mathbf{Adv}^{twinDH\text{-}test}_{Cheat,\mathbb{G}}(\lambda)$ is at most $1/p$.

For a proof of Lemma 2, see Appendix A.

3 Securing the Waters CP-ABKEM against Chosen-Ciphertext Attacks

In this section, we describe our direct chosen-ciphertext security technique by applying it to the Waters CP-ABE [Wat11].

Overview of Our Technique The Waters CP-ABE is proved to be secure in the IND-sel-CPA game [Wat11]. We convert it into a scheme that is secure in the IND-sel-CCA game by employing the Twin Diffie-Hellman technique of Cash, Kiltz and Shoup [CKS08] and the algebraic trick of Boneh and Boyen [BB04] and Kiltz [Kil06].

In encryption, a ciphertext becomes to contain additional two elements (d_1, d_2), which function in decryption as a "check sum" to verify that a tuple is certainly a twin DH tuple.

In security proof, the Twin Diffie-Hellman Trapdoor Test does the function instead. It is noteworthy that we can not use the bilinear map instead because the tuple to be verified is in the terminal group. In addition, the algebraic trick enables to answer for adversary's decryption queries. Note also that the both technique become compatible by introducing random variables like in Anada and Arita [AA11].

Key Encapsulation and Encryption. The Waters CP-ABE can be captured as a CP-ABKEM: the blinding factor of the form $e(g,g)^{\alpha s}$ in the Waters CP-ABE can be considered as a random one-time key. So we call it the Waters CP-ABKEM hereafter and denote it as $\texttt{CP-ABKEM}_{\rm cpa}$. Likewise, we distinguish parameters and algorithms of $\texttt{CP-ABKEM}_{\rm cpa}$ by the index $_{\rm cpa}$. For theoretical simplicity, we first develop a KEM CP-ABKEM.

3.1 Our Construction

Our CP-ABKEM consists of the following four PPT algorithms (Setup, Encap, KeyGen, Decap). Roughly speaking, the Waters original scheme $\texttt{CP-ABKEM}_{\rm cpa}$ (the first scheme in [Wat11]) corresponds to the case $k=1$ below excluding the "check sum" (d_1, d_2).

Setup($\lambda, \mathcal{U}$). Setup takes as input the security parameter λ and the attribute universe $\mathcal{U}=\{1,\ldots,u\}$. It runs $\texttt{Grp}(\lambda)$ to get $(p,\mathbb{G},\mathbb{G}_T,g,e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order p, $e:\mathbb{G}\to\mathbb{G}_T$ is a bilinear map and g is a generator of $\mathbb{G}$. These become public parameters. Then Setup chooses u random group elements $h_1,\ldots,h_u\in\mathbb{G}$ that are associated with the u attributes. In addition, it chooses random exponents $\alpha_k\in\mathbb{Z}_p, k=1,\ldots,4, a\in\mathbb{Z}_p$ and a hash key $\eta\in HKey(\lambda)$. The public key is published as $\mathrm{PK}=(g,g^a,h_1,\ldots,h_u,e(g,g)^{\alpha_1},\ldots,e(g,g)^{\alpha_4},\eta)$. The authority sets $\mathrm{MSK}=(g^{\alpha_1},\ldots,g^{\alpha_4})$ as the master secret key.

Encap(PK, $\mathbb{A}$). The encapsulation algorithm Encap takes as input the public key PK and an LSSS access structure $\mathbb{A}=(M,\rho)$, where M is an $l\times n$ matrix and ρ is the function which maps each row i of M to an attribute in $\mathcal{U}=\{1,\ldots,u\}$. Encap first chooses a random value $s\in\mathbb{Z}_p$ that is the encryption exponent s and random values $y_2,\ldots,y_n\in\mathbb{Z}_p$. Then Encap forms a vector $\boldsymbol{v}=(s,y_2,\ldots,y_n)$. For $i=1$ to l, it calculates $\lambda_i=\boldsymbol{v}\cdot M_i$, where M_i denotes the i-th row vector of M. In addition, Encap chooses random values $r_1,\ldots,r_l\in\mathbb{Z}_p$. Then, a pair of a random one-time key and its encapsulation (κ,ψ) is computed as follows.

$$
\begin{aligned}
&\text{Put } C'=g^s; \text{For } i=1 \text{ to } l: C_i=g^{a\lambda_i}h_{\rho(i)}^{-r_i}, D_i=g^{r_i};\\
&\psi_{\rm cpa}=(\mathbb{A},C',((C_i,D_i);i=1,\ldots,l)),\tau\leftarrow H_\eta(\psi_{\rm cpa});\\
&\text{For } k=1 \text{ to } 4: \kappa_k=e(g,g)^{\alpha_k s}; d_1=\kappa_1^{\tau}\kappa_3, d_2=\kappa_2^{\tau}\kappa_4;\\
&(\kappa,\psi)=(\kappa_1,(\psi_{\rm cpa},d_1,d_2)).
\end{aligned}
$$

KeyGen(MSK, PK, S). The key generation algorithm KeyGen takes as input the master secret key MSK, the public key PK and a set S of attributes. KeyGen first chooses a random $t_k\in\mathbb{Z}_p, k=1,\ldots,4$. It creates the secret key SK_S as follows.

$$
\begin{aligned}
&\text{For } k=1 \text{ to } 4: K_k=g^{\alpha_k}g^{at_k}, L_k=g^{t_k}, \text{For } x\in S: K_{k,x}=h_x^{t_k};\\
&\mathrm{SK}_S=((K_k,L_k,(K_{k,x};x\in S));k=1,\ldots,4).
\end{aligned}
$$

Decap(PK, ψ, SK_S). The decapsulation algorithm Decap takes as input the public key PK, an encapsulation ψ for the access structure $\mathbb{A}=(M,\rho)$ and a private key SK_S for an attribute set S. It first checks whether $S\in\mathbb{A}$. If the result is FALSE, put $\hat{\kappa}=\perp$. else, let $I_S=\rho^{-1}(S)\subset\{1,\ldots,l\}$ and let $\{\omega_i\in\mathbb{Z}_p; i\in I_S\}$ be a set of linear reconstruction constants. Then, the decapsulation $\hat{\kappa}$ is computed as follows.

$$
\begin{aligned}
&\text{Parse } \psi \text{ into } (\psi_{\rm cpa}=(\mathbb{A},C',((C_i,D_i);i=1,\ldots,l)),d_1,d_2);\tau\leftarrow H_\eta(\psi_{\rm cpa});\\
&\text{For } k=1 \text{ to } 4: \hat{\kappa_k}=e(C',K_k)/\prod_{i\in I_S}(e(C_i,L_k)e(D_i,K_{k,\rho(i)}))^{\omega_i}=e(g,g)^{\alpha_k s},\\
&\text{If } \hat{\kappa_1}^{\tau}\hat{\kappa_3}\neq d_1\vee\hat{\kappa_2}^{\tau}\hat{\kappa_4}\neq d_2, \text{then put } \hat{\kappa}=\perp, \text{else put } \hat{\kappa}=\hat{\kappa_1}.
\end{aligned}
$$

3.2 Security and its Proof

Theorem 1 *If the Waters* $\texttt{CP-ABKEM}_{cpa}$ *[Wat11] is selectively secure against chosen-plaintext attacks and an employed hash function family* Hfam *has target collision resistance, then our* CP-ABKEM *is selectively secure against chosen-ciphertext attacks. More precisely, for any given PPT adversary* $\mathcal{A}$ *that attacks* CP-ABKEM *in the IND-sel-CCA game where decapsulation queries are at most* q_d *times, and for any attribute universe* $\mathcal{U}$*, there exist a PPT adversary* $\mathcal{B}$ *that attacks* $\texttt{CP-ABKEM}_{cpa}$ *in the IND-sel-CPA game and a PPT target collision finder* $\mathcal{CF}$ *on* Hfam *that satisfy the following tight reduction.*

$$\mathbf{Adv}^{ind\text{-}sel\text{-}cca}_{\mathcal{A},\texttt{CP-ABKEM}}(\lambda,\mathcal{U}) \leq \mathbf{Adv}^{ind\text{-}sel\text{-}cpa}_{\mathcal{B},\texttt{CP-ABKEM}_{cpa}}(\lambda,\mathcal{U}) + \mathbf{Adv}^{tcr}_{\mathcal{CF},Hfam}(\lambda) + \frac{q_d}{p}.$$

A definition of the target collision resistance game and the advantage of $\mathcal{CF}$ are given in Appendix H.

Proof. Given any adversary $\mathcal{A}$ that attacks our scheme CP-ABKEM in the IND-sel-CCA game, we construct an adversary $\mathcal{B}$ that attacks the Waters scheme $\texttt{CP-ABKEM}_{\mathrm{cpa}}$ in the IND-sel-CPA game as follows.

Commit to a Target Access Structure. $\mathcal{B}$ is given $(\lambda,\mathcal{U})$ as inputs, where λ is the security parameter and $\mathcal{U}=\{1,\ldots,u\}$ is the attribute universe. $\mathcal{B}$ invokes $\mathcal{A}$ on input $(\lambda,\mathcal{U})$ and gets a target access structure $\mathbb{A}^*=(M^*,\rho^*)$ from $\mathcal{A}$, where M^* is of size $l^*\times n^*$. $\mathcal{B}$ uses $\mathbb{A}^*$ as the target access structure of itself and outputs $\mathbb{A}^*$.

Set up. In return to outputting $\mathbb{A}^*$, $\mathcal{B}$ receives the public key $\mathrm{PK}_{\mathrm{cpa}}$ for $\texttt{CP-ABKEM}_{\mathrm{cpa}}$, which consists of the following components.

$$\mathrm{PK}_{\mathrm{cpa}} = (g, g^a, h_1, \ldots, h_u, e(g,g)^{\alpha}).$$

To set up a public key PK for CP-ABKEM, $\mathcal{B}$ *herein* needs a challenge instance: $\mathcal{B}$ queries its challenger and gets a challenge instance $(\tilde{\kappa}, \psi^*_{\mathrm{cpa}})$. It consists of the following components.

$$\begin{aligned}
&\tilde{\kappa} = e(g,g)^{\alpha s^*} \text{ OR a random one-time key } \kappa \in \mathit{KeySp}(\lambda),\\
&\psi^*_{\mathrm{cpa}} = (\mathbb{A}^*, C'^* = g^{s^*}, ((C^*_i, D^*_i); i=1,\ldots,l^*)).
\end{aligned}$$

Then $\mathcal{B}$ makes the rest of parameters of PK as follows.

$$\begin{aligned}
&\text{Pick up } \eta \leftarrow \mathit{HKey}(\lambda) \text{ and take } \tau^* \leftarrow H_\eta(\psi^*_{\mathrm{cpa}});\\
&\text{Put } e(g,g)^{\alpha_1} = e(g,g)^{\alpha};\\
&\text{Pick up } \gamma_1,\gamma_2 \leftarrow \mathbb{Z}_p \text{ and put } e(g,g)^{\alpha_2} = e(g,g)^{\gamma_2}/e(g,g)^{\alpha_1\gamma_1};\\
&\text{Pick up } \mu_1,\mu_2 \leftarrow \mathbb{Z}_p \text{ and put } e(g,g)^{\alpha_3} = e(g,g)^{\mu_1}/e(g,g)^{\alpha_1\tau^*},\\
&\qquad\qquad e(g,g)^{\alpha_4} = e(g,g)^{\mu_2}/e(g,g)^{\alpha_2\tau^*}.
\end{aligned}$$

Note we have implicitly set relations in the exponent domain:

$$\alpha_2 = \gamma_2 - \alpha_1\gamma_1, \quad \alpha_3 = \mu_1 - \alpha_1\tau^*, \quad \alpha_4 = \mu_2 - \alpha_2\tau^* = \mu_2 - (\gamma_2 - \alpha_1\gamma_1)\tau^*. \tag{1}$$

A public key PK for CP-ABKEM become:

$$\mathrm{PK} = (\mathrm{PK}_{\mathrm{cpa}}, e(g,g)^{\alpha_2}, e(g,g)^{\alpha_3}, e(g,g)^{\alpha_4}, \eta).$$

Then $\mathcal{B}$ inputs PK into $\mathcal{A}$. Note that PK determines the corresponding MSK uniquely.

Phase 1. $\mathcal{B}$ answers for two types of $\mathcal{A}$'s queries as follows.
(1) Key-Extraction Queries. In the case that $\mathcal{A}$ issues a key-extraction query for an attribute set $S \subset \mathcal{U}$, $\mathcal{B}$ has to simulate $\mathcal{A}$'s challenger. To do so, $\mathcal{B}$ issues key-extraction queries to $\mathcal{B}$'s challenger for S *repeatedly up to four times*. As replies, $\mathcal{B}$ gets four secret keys of the Waters $\texttt{CP-ABKEM}_{\mathrm{cpa}}$ for a single attribute set S:

$$\mathrm{SK}_{\mathrm{cpa},S,k} = (K_{\mathrm{cpa},k}, L_{\mathrm{cpa},k}, (K_{\mathrm{cpa},k,x}; x \in \mathcal{S})), k = 1, \ldots, 4.$$

We remark that, according to the randomness in the key-generation algorithm of the Waters $\texttt{CP-ABKEM}_{\mathrm{cpa}}$, all four secret keys $\mathrm{SK}_{\mathrm{cpa},S,1}, \ldots, \mathrm{SK}_{\mathrm{cpa},S,4}$ are random and mutually independent. To reply a secret key SK_S of our $\texttt{CP-ABKEM}$ to $\mathcal{A}$, $\mathcal{B}$ converts the four secret keys as follows.

$$\begin{aligned}
&\text{Put } K_1 = K_{\mathrm{cpa},1}, && L_1 = L_{\mathrm{cpa},1},\ K_{1,x} = K_{\mathrm{cpa},1,x}, x \in S;\\
&\text{Put } K_2 = g^{\gamma_2} K_{\mathrm{cpa},2}^{-\gamma_1}, && L_2 = L_{\mathrm{cpa},2}^{-\gamma_1},\ K_{2,x} = K_{\mathrm{cpa},2,x}^{-\gamma_1},\ x \in S;\\
&\text{Put } K_3 = g^{\mu_1} K_{\mathrm{cpa},3}^{-\tau^*}, && L_3 = L_{\mathrm{cpa},3}^{-\tau^*},\ K_{3,x} = K_{\mathrm{cpa},3,x}^{\tau^*},\quad x \in S;\\
&\text{Put } K_4 = g^{\mu_2 - \gamma_2\tau^*} K_{\mathrm{cpa},4}^{\gamma_1\tau^*}, && L_4 = L_{\mathrm{cpa},4}^{\gamma_1\tau^*},\ K_{4,x} = K_{\mathrm{cpa},4,x}^{\gamma_1\tau^*},\quad x \in S.
\end{aligned}$$

Then $\mathcal{B}$ replies $\mathrm{SK}_S = ((K_k, L_k, (K_{k,x}; x \in S)); k = 1, \ldots, 4)$ to $\mathcal{A}$.

(2) Decapsulation Queries. In the case that $\mathcal{A}$ issues a decapsulation query for (S, ψ), where $S \subset \mathcal{U}$ is an attribute set and $\psi = (\psi_{\mathrm{cpa}}, d_1, d_2)$ is an encapsulation concerning $\mathbb{A}$, $\mathcal{B}$ has to simulate $\mathcal{A}$'s challenger. To do so, $\mathcal{B}$ computes the decapsulation result $\hat{\kappa}$ as follows.

If $S \notin \mathbb{A}$ then put $\hat{\kappa} = \perp$,
else
  Take $\tau \leftarrow H_\eta(\psi_{\mathrm{cpa}})$;
  Put $\hat{Y} = e(C', g)^{\tau - \tau^*}, \hat{Z}_1 = d_1 / e(C', g)^{\mu_1}, \hat{Z}_2 = d_2 / e(C', g)^{\mu_2}$;
  If $\hat{Z}_1^{\gamma_1} \hat{Z}_2 \neq \hat{Y}^{\gamma_2}$ (: call this checking TwinDH-Test)
    then put $\hat{\kappa} = \hat{\kappa_1} = \perp$
  else
    If $\tau = \tau^*$ then abort (: call this case Abort)
    else $\hat{\kappa} = \hat{\kappa_1} = \hat{Z}_1^{1/(\tau - \tau^*)}$.

Challenge. In the case that $\mathcal{A}$ queries its challenger for a challenge instance, $\mathcal{B}$ makes a challenge instance as follows.

$$\begin{aligned}
&\text{Put } d_1^* = e(C'^*, g)^{\mu_1}, d_2^* = e(C'^*, g)^{\mu_2};\\
&\text{Put } \psi^* = (\psi_{\mathrm{cpa}}^*, d_1^*, d_2^*).
\end{aligned}$$

Then $\mathcal{B}$ feeds $(\tilde{\kappa}, \psi^*)$ to $\mathcal{A}$ as a challenge instance.

Phase 2. The same as in Phase 1.

Guess. In the case that $\mathcal{A}$ returns $\mathcal{A}$'s guess $\tilde{b}$, $\mathcal{B}$ returns $\tilde{b}$ itself as $\mathcal{B}$'s guess.

In the above construction of $\mathcal{B}$, $\mathcal{B}$ can perfectly simulate the real view of $\mathcal{A}$ until the case Abort happens, except for a negligible case, and hence the algorithm $\mathcal{A}$ works as designed. To see the perfect simulation with a negligible exceptional case, we are enough to prove the following seven claims.

Claim 1 *The reply* $SK_S = ((K_k, L_k, (K_{k,x}; x \in S)); k = 1, \ldots, 4)$ *for a key-extraction query of* $\mathcal{A}$ *is a perfect simulation.*

Proof. We must consider the implicit relations (1). For the index 2, we have implicitly set the randomness $t_2 = t_{\mathrm{cpa},2}(-\gamma_1)$ and we get:

$$K_2 = g^{\gamma_2} K_{\mathrm{cpa},2}^{-\gamma_1} = g^{\gamma_2}(g^{\alpha_1} g^{at_{\mathrm{cpa},2}})^{-\gamma_1} = g^{\gamma_2}(g^{\alpha_1} g^{at_2/(-\gamma_1)})^{-\gamma_1} = g^{\gamma_2 - \alpha_1\gamma_1} g^{at_2} = g^{\alpha_2} g^{at_2},$$
$$L_2 = L_{\mathrm{cpa},2}^{-\gamma_1} = (g^{t_{\mathrm{cpa},2}})^{-\gamma_1} = g^{t_2},$$
$$K_{2,x} = K_{\mathrm{cpa},2,x}^{-\gamma_1} = (h_x^{t_{\mathrm{cpa},2}})^{-\gamma_1} = h_x^{t_2}, x \in S.$$

For the index 3 and 4, see Appendix B.

Claim 2 $(e(g,g),\ e(g,g)^{\alpha_1},\ e(g,g)^{\alpha_2},\ \hat{Y},\ \hat{Z}_1,\ \hat{Z}_2)$ *is a twin Diffie-Hellman tuple if and only if* $(e(g,g),\ e(g,g)^{\alpha_1\tau} e(g,g)^{\alpha_3},\ e(g,g)^{\alpha_2\tau} e(g,g)^{\alpha_4},\ e(C',g),\ d_1,\ d_2)$ *is a twin Diffie-Hellman tuple.*

Proof. This claim can be proved by a short calculation. See Appendix C.

Claim 3 *If* $(e(g,g), e(g,g)^{\alpha_1}, e(g,g)^{\alpha_2}, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$ *is a twin Diffie-Hellman tuple, then* $(\hat{Y}, \hat{Z}_1, \hat{Z}_2)$ *certainly passes the* TwinDH-Test*:* ${\hat{Z}_1}^{\gamma_1} \hat{Z}_2 = \hat{Y}^{\gamma_2}$.

Proof. This claim is a direct consequence of Lemma 1. □

Claim 4 *Consider the following event which we name as* $\textsc{Overlook}_i$*:*

In the i*-th* TwinDH-Test*, the following condition holds:*

$$\begin{cases} {\hat{Z}_1}^{\gamma_1} \hat{Z}_2 = \hat{Y}^{\gamma_2} \text{ holds and} \\ (e(g,g), e(g,g)^{\alpha_1}, e(g,g)^{\alpha_2}, \hat{Y}, \hat{Z}_1, \hat{Z}_2) \text{ is NOT a twin DH tuple.} \end{cases}$$

Then, for at most q_d *times decapsulation queries of* $\mathcal{A}$*, the probability that at least one* $\textsc{Overlook}_i$ *occurs is negligible in* λ*. More precisely, the following inequality holds:*

$$\Pr[\bigvee_{i=1}^{q_d} \textsc{Overlook}_i] \leq q_d/p. \tag{2}$$

Proof. To apply Lemma 2, we construct an algorithm $\mathit{Cheat}_{\lambda,\mathcal{U}}$ with unbounded computational power, which takes as input $(e(g,g), e(g,g)^{\alpha_1}, e(g,g)^{\alpha_2})$ and returns $(\hat{Y}, \hat{Z}_1, \hat{Z}_2)$ employing the adversary $\mathcal{A}$ as a subroutine. Fig. 1 shows the construction.

First, note that the view of $\mathcal{A}$ in $\mathit{Cheat}_{\lambda,\mathcal{U}}$ is the same as the real view of $\mathcal{A}$ and hence the algorithm $\mathcal{A}$ works as designed.

Second, note that the return $(\hat{Y}, \hat{Z}_1, \hat{Z}_2)$ of $\mathit{Cheat}_{\lambda,\mathcal{U}}$ is randomized in TABLE. Hence:

$$\sum_{i=1}^{q_\mathrm{d}} \frac{1}{q_\mathrm{d}} \Pr[\textsc{Overlook}_i] = \frac{1}{q_\mathrm{d}} \sum_{i=1}^{q_\mathrm{d}} \Pr[\textsc{Overlook}_i] = \mathbf{Adv}^{\text{twinDH-test}}_{\mathit{Cheat}_{\lambda,\mathcal{U}},\mathbb{G}}(\lambda). \tag{3}$$

Third, applying Lemma 2 to $\mathit{Cheat}_{\lambda,\mathcal{U}}$, we get:

$$\mathbf{Adv}^{\text{twinDH-test}}_{\mathit{Cheat}_{\lambda,\mathcal{U}},\mathbb{G}}(\lambda) \leq 1/p. \tag{4}$$

Combining (3) and (4), we have:

$$\Pr[\bigvee_{i=1}^{q_\mathrm{d}} \textsc{Overlook}_i] \leq \sum_{i=1}^{q_\mathrm{d}} \Pr[\textsc{Overlook}_i] \leq q_\mathrm{d} \mathbf{Adv}^{\text{twinDH-test}}_{\mathit{Cheat}_{\lambda,\mathcal{U}},\mathbb{G}}(\lambda) \leq \frac{q_\mathrm{d}}{p}. \quad \square$$

Given $(e(g,g), e(g,g)^{\alpha_1}, e(g,g)^{\alpha_2})$ as input :
Set up
Initialize the inner state and put TABLE $= \phi$;
Get a target access structure $\mathbb{A}^* \leftarrow \mathcal{A}(\lambda, \mathcal{U})$;
Compute the base $g \in \mathbb{G}$ from $(e(g,g), e)$;
Pick up $a \in \mathbb{Z}_p$ and $h_1, \ldots, h_u \in \mathbb{G}$;
Put $\mathrm{PK}_{\mathrm{cpa}} = (g, g^a, h_1, \ldots, h_u, e(g,g)^{\alpha_1})$;
Get $(\kappa^*, \psi^*_{\mathrm{cpa}}) \leftarrow \mathbf{Encap}_{\mathrm{cpa}}(\mathrm{PK}_{\mathrm{cpa}}, \mathbb{A}^*)$;
Pick up $\eta \leftarrow \mathit{HKey}(\lambda)$ and compute $\tau^* \leftarrow H_\eta(\psi^*_{\mathrm{cpa}})$;
Compute discrete logarithms $\alpha_1, \alpha_2 \in \mathbb{Z}_p$ of $e(g,g)^{\alpha_1}, e(g,g)^{\alpha_2}$ to the base $e(g,g)$;
Pick up $\mu_1, \mu_2 \leftarrow \mathbb{Z}_p$ and put $\alpha_3 = \mu_1 - \alpha_1\tau^*, \alpha_4 = \mu_2 - \alpha_2\tau^*$;
Put $\mathrm{PK} = (\mathrm{PK}_{\mathrm{cpa}}, e(g,g)^{\alpha_2}, e(g,g)^{\alpha_3}, e(g,g)^{\alpha_4}, \eta), \mathrm{MSK} = (g^{\alpha_1}, g^{\alpha_2}, g^{\alpha_3}, g^{\alpha_4})$;
Give PK to $\mathcal{A}$;
Phase 1
In the case that $\mathcal{A}$ makes a key-extraction query for $S \subset \mathcal{U}$;
Reply SK_S to $\mathcal{A}$ in the same way as **KeyGen** does using MSK;
In the case that $\mathcal{A}$ makes a decapsulation query for $(\mathbb{A}, \psi = (\psi_{\mathrm{cpa}}, d_1, d_2), S)$;
Reply $\hat{\kappa}$ to $\mathcal{A}$ in the same way as **Decap** does using MSK;
Compute $\hat{Y} = e(C', g)^{\tau - \tau^*}, \hat{Z}_1 = d_1 / e(C', g)^{\mu_1}, \hat{Z}_2 = d_2 / e(C', g)^{\mu_2}$;
Update TABLE $=$ TABLE $\cup (\hat{Y}, \hat{Z}_1, \hat{Z}_2)$;
Challenge
In the case that $\mathcal{A}$ makes a challenge instance query;
Put $d_1^* = e(C'^*, g)^{\mu_1}, d_2^* = e(C'^*, g)^{\mu_2}, \psi^* = (\psi^*_{\mathrm{cpa}}, d_1^*, d_2^*)$;
Pick up $\kappa \leftarrow \mathit{KeySp}(\lambda), b \leftarrow \{0,1\}$;
If $b = 1$ then put $\tilde{\kappa} = \kappa^*$ else put $\tilde{\kappa} = \kappa$;
Reply $(\tilde{\kappa}, \psi^*)$ to $\mathcal{A}$;
Phase 2
The same as in Phase 1;
Return
In the case that $\mathcal{A}$ returns its guess b^*;
Choose one triple $(\hat{Y}, \hat{Z}_1, \hat{Z}_2)$ from TABLE at random;
Return $(\hat{Y}, \hat{Z}_1, \hat{Z}_2)$.

Fig. 1. An Algorithm $\mathit{Cheat}_{\lambda,\mathcal{U}}$ with Unbounded Computational Power for a Proof of Claim 4.

Claim 5 *The probability that* $\textsc{Overlook}_i$ *never occurs in* TwinDH-Test*for each* i *and* Abort *occurs is negligible in* λ*. More precisely, the following inequality holds:*

$$\Pr[\Big(\bigwedge_{i=1}^{q_d} \neg \textsc{Overlook}_i\Big) \wedge \textsc{Abort}] \leq \mathbf{Adv}^{tcr}_{\mathcal{CF},\mathit{Hfam}}(\lambda). \tag{5}$$

Proof. This claim is proved by constructing a collision finder $\mathcal{CF}$ on *Hfam*. See Appendix D.

Claim 6 *The reply* $\hat{\kappa}$ *to* $\mathcal{A}$ *as an answer for a decapsulation query is correct.*

Claim 7 *The challenge instance* $\psi^* = (\psi^*_{cpa}, d_1^*, d_2^*)$ *is correctly distributed.*

Proof. These claims are proved by a direct calculation. See Appendices E and F, respectively.

Now we are ready to evaluate the advantage of $\mathcal{B}$ in the IND-sel-CPA game. That $\mathcal{A}$ wins in the IND-sel-CCA game means that $(\tilde{\kappa}, \psi^* = (\psi^*_{\mathrm{cpa}}, d_1^*, d_2^*))$ is correctly guessed. This is equivalent to that $(\tilde{\kappa}, \psi^*_{\mathrm{cpa}})$ is correctly guessed because ψ^*_{cpa} determines the consistent blinding factor $\kappa^* = e(g,g)^{\alpha s^*}$ uniquely. This means that $\mathcal{B}$ wins in the IND-sel-CPA game.

Therefore, the probability that $\mathcal{B}$ wins is equal to the probability that $\mathcal{A}$ wins, $\textsc{Overlook}_i$ never holds in TwinDH-Testfor each i and Abort never occurs. So we have:

$$\begin{aligned}
\Pr[\mathcal{B} \text{ wins}] &= \Pr[(\mathcal{A} \text{ wins}) \wedge \Big(\bigwedge_{i=1}^{q_{\mathrm{d}}} \neg\textsc{Overlook}_i\Big) \wedge (\neg\textsc{Abort})] \\
&= \Pr[\mathcal{A} \text{ wins}] - \Pr[(\mathcal{A} \text{ wins}) \wedge \neg\Big(\Big(\bigwedge_{i=1}^{q_{\mathrm{d}}} \neg\textsc{Overlook}_i\Big) \wedge (\neg\textsc{Abort})\Big)] \\
&\geq \Pr[\mathcal{A} \text{ wins}] - \Pr[\neg\Big(\Big(\bigwedge_{i=1}^{q_{\mathrm{d}}} \neg\textsc{Overlook}_i\Big) \wedge (\neg\textsc{Abort})\Big)] \\
&= \Pr[\mathcal{A} \text{ wins}] - (\Pr[\bigvee_{i=1}^{q_{\mathrm{d}}} \textsc{Overlook}_i] + \Pr[\Big(\bigwedge_{i=1}^{q_{\mathrm{d}}} \neg\textsc{Overlook}_i\Big) \wedge \textsc{Abort}]).
\end{aligned}$$

Substituting (2), (5) and advantages into the above, we have:

$$\mathbf{Adv}^{\text{ind-sel-cpa}}_{\mathcal{B},\mathtt{CP\text{-}ABKEM}_{\mathrm{cpa}}}(\lambda,\mathcal{U}) \geq \mathbf{Adv}^{\text{ind-sel-cca}}_{\mathcal{A},\mathtt{CP\text{-}ABKEM}}(\lambda,\mathcal{U}) - \frac{q_{\mathrm{d}}}{p} - \mathbf{Adv}^{\mathrm{tcr}}_{\mathcal{CF},Hfam}(\lambda). \quad \square$$

3.3 Discussion

Encryption Version. It is straightforward to construct our encryption scheme `CP-ABE` from `CP-ABKEM`. The IND-sel-CCA security of `CP-ABE` is proved based on IND-sel-CPA security of the Waters *KEM* $\mathtt{CP\text{-}ABKEM}_{\mathrm{cpa}}$. See Appendix G.

The Case of Adaptive Game on a Target Access Structure. Lewko, Okamoto, Sahai, Takashima and Waters [LOS+10] converted the Waters scheme [Wat11] into the one that attain the security against adversary's adaptive choice of a target access structure.

We can apply the same conversion as in Section 3.1 to their scheme [LOS+10]. In addition, the IND-CCA security of their scheme can be proved along the way as in Section 3.2, but in the *random oracle model* for the hash function H used in encapsulation and decapsulation.

4 Conclusions

We developed a technique of direct chosen-ciphertext security for ABE in the standard model in the case of the Waters scheme ($\mathtt{CP\text{-}ABKEM}_{\mathrm{cpa}}, \mathtt{CP\text{-}ABE}_{\mathrm{cpa}}$). We utilized the Twin Diffie-Hellman Trapdoor Test of Cash, Kiltz and Shoup and the algebraic trick of Boneh and Boyen [BB04] and Kiltz [Kil06]. Our technique is helpful when a Diffie-Hellman tuple to be verified is in a terminal group of a bilinear map. It results in expansion of secret key length and decryption cost of computation by a factor of four, while public key length, ciphertext length and encryption cost of computation are almost the same as those of the Waters.

References

[AA11] Hiroaki Anada and Seiko Arita. Identification schemes from key encapsulation mechanisms. In *Progress in Cryptology - AFRICACRYPT 2011 - 4th International Conference on Cryptology in Africa, Dakar, Senegal, July 5-7, 2011. Proceedings*, pages 59–76, 2011.

[ALdP11] Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*, pages 90–108, 2011.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 223–238, 2004.

[BMW05] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In *Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, November 7-11, 2005*, pages 320–329, 2005.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 207–222, 2004.

[CKS08] David Cash, Eike Kiltz, and Victor Shoup. The twin diffie-hellman problem and applications. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, pages 127–145, 2008.

[CZF11] Cheng Chen, Zhenfeng Zhang, and Dengguo Feng. Efficient ciphertext policy attribute-based encryption with constant-size ciphertext and constant computation-cost. In *Provable Security - 5th International Conference, ProvSec 2011, Xi'an, China, October 16-18, 2011. Proceedings*, pages 84–101, 2011.

[GBN10] M. Choudary Gorantla, Colin Boyd, and Juan Manuel González Nieto. Attribute-based authenticated key exchange. In *Information Security and Privacy - 15th Australasian Conference, ACISP 2010, Sydney, Australia, July 5-7, 2010. Proceedings*, pages 300–317, 2010.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 89–98, 2006.

[GZC+12] Aijun Ge, Rui Zhang, Cheng Chen, Chuangui Ma, and Zhenfeng Zhang. Threshold ciphertext policy attribute-based encryption with constant size ciphertexts. In *Information Security and Privacy - 17th Australasian Conference, ACISP 2012, Wollongong, NSW, Australia, July 9-11, 2012. Proceedings*, pages 336–349, 2012.

[HLR10] Javier Herranz, Fabien Laguillaumie, and Carla Ràfols. Constant size ciphertexts in threshold attribute-based encryption. In *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*, pages 19–34, 2010.

[Kil06] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, pages 581–600, 2006.

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, pages 62–91, 2010.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 33–43, 1989.

[SW05] 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, 2005.

[Wat11] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*, pages 53–70, 2011.

[YAHK11] Shota Yamada, Nuttapong Attrapadung, Goichiro Hanaoka, and Noboru Kunihiro. Generic constructions for chosen-ciphertext secure attribute based encryption. In *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*, pages 71–89, 2011.

Appendix

A Proof of Lemma 2

Only one point to be complemented to the original proof (in [CKS08]) is that even for any algorithm $\mathcal{A}$ with unbounded computational power, the statement holds. This is because, conditioning on input fixed values (g, X_1, X_2), $\mathcal{A}$ at most reduces two-dimensional freedom $(r, s) \in \mathbb{Z}_p^2$ into one-dimensional freedom $r \in \mathbb{Z}_p$ even if $\mathcal{A}$ correctly guesses the relation $s = rx_1 + x_2$. □

B Proof of Claim 1

For the index 3, we have implicitly set $t_3 = t_{\mathrm{cpa},3}(-\tau^*)$ and we get:

$$\begin{aligned}
K_3 &= g^{\mu_1} K_{\mathrm{cpa},3}^{-\tau^*} = g^{\mu_1}(g^{\alpha_1} g^{a t_{\mathrm{cpa},3}})^{-\tau^*} = g^{\mu_1 - \alpha_1 \tau^*} g^{a t_3} = g^{\alpha_3} g^{a t_3}, \\
L_3 &= L_{\mathrm{cpa},3}^{-\tau^*} = (g^{t_{\mathrm{cpa},3}})^{-\tau^*} = g^{t_3}, \\
K_{3,x} &= K_{\mathrm{cpa},3,x}^{-\tau^*} = (h_x^{t_{\mathrm{cpa},3}})^{-\tau^*} = h_x^{t_3}, x \in S.
\end{aligned}$$

For the index 4, we have implicitly set $t_4 = t_{\mathrm{cpa},4}(\gamma_1 \tau^*)$ and we get:

$$\begin{aligned}
K_4 &= g^{\mu_2 - \gamma_2 \tau^*} K_{\mathrm{cpa},4}^{\gamma_1 \tau^*} = g^{\mu_2 - \gamma_2 \tau^*}(g^{\alpha_1} g^{a t_{\mathrm{cpa},4}})^{\gamma_1 \tau^*} = g^{\mu_2 - \gamma_2 \tau^*} g^{\alpha_1 \gamma_1 \tau^*} g^{a t_4} \\
&= g^{\mu_2 - (\gamma_2 - \alpha_1 \gamma_1)\tau^*} g^{a t_4} = g^{\mu_2 - \alpha_2 \tau^*} g^{a t_4} = g^{\alpha_4} g^{a t_4}, \\
L_4 &= L_{\mathrm{cpa},4}^{\gamma_1 \tau^*} = (g^{t_{\mathrm{cpa},4}})^{\gamma_1 \tau^*} = g^{t_4}, \\
K_{4,x} &= K_{\mathrm{cpa},4,x}^{\gamma_1 \tau^*} = (h_x^{t_{\mathrm{cpa},4}})^{\gamma_1 \tau^*} = h_x^{t_4}, x \in S. \quad \square
\end{aligned}$$

C Proof of Claim 2

Suppose that we are given a twin DH tuple $(e(g,g), e(g,g)^{\alpha_1}, e(g,g)^{\alpha_2}, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$. Then, $d_i / e(C', g)^{\mu_i} = (e(g,g)^{\alpha_i})^{s(\tau - \tau^*)}, i = 1, 2$. So, using the implicit relations (1), we have:

$$\begin{aligned}
d_i &= e(g,g)^{\alpha_i s(\tau - \tau^*)} e(g^s, g)^{\mu_i} \\
&= (e(g,g)^{\alpha_i(\tau - \tau^*)} e(g,g)^{\mu_i})^s \\
&= (e(g,g)^{\alpha_i(\tau - \tau^*)} e(g,g)^{\alpha_i \tau^* + \alpha_{(i+2)}})^s \\
&= (e(g,g)^{\alpha_i \tau} e(g,g)^{\alpha_{(i+2)}})^s, i = 1, 2.
\end{aligned}$$

This means that $(e(g,g), e(g,g)^{\alpha_1 \tau} e(g,g)^{\alpha_3}, e(g,g)^{\alpha_2 \tau} e(g,g)^{\alpha_4}, e(C', g), d_1, d_2)$ is a twin Diffie-Hellman tuple.

The converse is also verified by the reverse calculation. □

D Proof of Claim 5

To reduce to the target collision resistance of an employed hash function family *Hfam*, we construct a PPT target collision finder $\mathcal{CF}$ that attacks *Hfam* using $\mathcal{A}$ as a subroutine. The construction is shown in Fig.2. (Remark that the case Collision is defined in Fig.2.)

Note that the view of $\mathcal{A}$ in $\mathcal{CF}$ is the same as the real view of $\mathcal{A}$ until the case Collision occurs and hence the algorithm $\mathcal{A}$ works as designed.

Given λ as input :

Set up

Initialize inner state;
Choose a polynomial size attribute universe $\mathcal{U}$ at random;
Get a target access structure $\mathbb{A}^* \leftarrow \mathcal{A}(\lambda, \mathcal{U})$;
Run $\mathbf{Setup}_{\mathrm{cpa}}(\lambda, \mathcal{U})$ to get $(p, \mathbb{G}, \mathbb{G}_T, g, e), \mathrm{PK}_{\mathrm{cpa}}, \mathrm{MSK}_{\mathrm{cpa}}$;
Get $(\kappa^*, \psi^*_{\mathrm{cpa}}) \leftarrow \mathbf{Encap}_{\mathrm{cpa}}(\mathrm{PK}_{\mathrm{cpa}}, \mathbb{A}^*)$;
Output ψ^*_{cpa};
Receive, in return, $\eta \leftarrow \mathit{HKey}(\lambda)$ and compute $\tau^* \leftarrow H_\eta(\psi^*_{\mathrm{cpa}})$;
Pick up $\alpha_2, \alpha_3, \alpha_4 \leftarrow \mathbb{Z}_p$;
Put $\mathrm{PK} = (\mathrm{PK}_{\mathrm{cpa}}, e(g,g)^{\alpha_2}, e(g,g)^{\alpha_3}, e(g,g)^{\alpha_4}, \eta), \mathrm{MSK} = (g^{\alpha_1}, g^{\alpha_2}, g^{\alpha_3}, g^{\alpha_4})$;
Give PK to $\mathcal{A}$;

Phase 1

In the case that $\mathcal{A}$ makes a key-extraction query for $S \subset \mathcal{U}$;
  Reply SK_S to $\mathcal{A}$ in the same way as **KeyGen** does using MSK;
In the case that $\mathcal{A}$ makes a decapsulation query for $(S, \psi = (\psi_{\mathrm{cpa}}, d_1, d_2))$;
  Reply $\hat{\kappa}$ to $\mathcal{A}$ in the same way as **Decap** does using MSK;
  If $\hat{\kappa} \neq \perp$ and $\tau = \tau^*$ (: call this case Collision)
    then return ψ_{cpa} and stop;

Challenge

In the case that $\mathcal{A}$ makes a challenge instance query;
  Using MSK, put $d_1^* = e(g^{\alpha_1}, C'^*)^{\tau^*} e(g^{\alpha_3}, C'^*), d_2^* = e(g^{\alpha_2}, C'^*)^{\tau^*} e(g^{\alpha_4}, C'^*)$,
    $\psi^* = (\psi^*_{\mathrm{cpa}}, d_1^*, d_2^*)$;
  Pick up $\kappa \leftarrow \mathit{KeySp}(\lambda), b \leftarrow \{0,1\}$;
  If $b = 1$ then put $\tilde{\kappa} = \kappa^*$ else put $\tilde{\kappa} = \kappa$;
  Reply $(\tilde{\kappa}, \psi^*)$ to $\mathcal{A}$;

Phase 2

The same as in Phase 1;

Return

In the case that $\mathcal{A}$ returns its guess b^*;
  Stop.

Fig. 2. A PPT Collision Finder $\mathcal{CF}$ that attacks *Hfam* for the proof of Claim 5.

To evaluate the probability in Claim 5, we consider the following two cases.
Case 1: the case that Abort ($\tau = \tau^*$) occurs in $\mathcal{B}$ in Phase 1. In this case, the target τ^* has not been given to $\mathcal{A}$. So $\mathcal{A}$ needs to guess τ^* to cause a collision $\tau = \tau^*$. Hence:

$$\Pr[\text{Phase 1} \wedge \Big(\bigwedge_{i=1}^{q_{\mathrm{d}}} \neg \text{Overlook}_i\Big) \wedge \text{Abort}] \leq \Pr[\text{Phase 1} \wedge \text{Collision}]. \tag{6}$$

Case 2: the case that Abort ($\tau = \tau^*$) occurs in $\mathcal{B}$ in Phase 2. In this case, if, in addition to $\tau = \tau^*$, it occurred that $\psi_{\mathrm{cpa}} = \psi^*_{\mathrm{cpa}}$ (and hence $C' = C'^*$), then it would occur that $\psi = \psi^*$. This is because the following two tuples are equal twin DH tuples by the fact that Overlook_i never occurs:

$$(e(g,g), e(g,g)^{\alpha_1 \tau} e(g,g)^{\alpha_3}, e(g,g)^{\alpha_2 \tau} e(g,g)^{\alpha_4}, e(C',g), d_1, d_2),$$
$$(e(g,g), e(g,g)^{\alpha_1 \tau^*} e(g,g)^{\alpha_3}, e(g,g)^{\alpha_2 \tau^*} e(g,g)^{\alpha_4}, e(C'^*,g), d_1^*, d_2^*).$$

Hence both $S \in \mathbb{A}$ and $\psi = \psi^*$ would occur. This is ruled out in decapsulation query; a contradiction. So we have $\psi_{\mathrm{cpa}} \neq \psi^*_{\mathrm{cpa}}$; that is, a collision:

$$\psi_{\mathrm{cpa}} \neq \psi^*_{\mathrm{cpa}} \wedge H_\eta(\psi_{\mathrm{cpa}}) = \tau = \tau^* = H_\eta(\psi^*_{\mathrm{cpa}}).$$

Therefore, if Overlook_i never occurs for each i, then only decapsulation queries for which $(e(g,g), e(g,g)^{\alpha_1}, e(g,g)^{\alpha_2}, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$ are certainly twin DH tuples have the chance to cause a collision $\tau = \tau^*$,

as is the case in $\mathcal{CF}$. Hence we have:

$$\Pr[\text{Phase 2} \wedge \Big(\bigwedge_{i=1}^{q_{\mathrm{d}}} \neg\textsc{Overlook}_i\Big) \wedge \textsc{Abort}] \leq \Pr[\text{Phase 2} \wedge \textsc{Collision}]. \tag{7}$$

Taking a sum of both sides of (6) and (7), we get:

$$\Pr[\Big(\bigwedge_{i=1}^{q_{\mathrm{d}}} \neg\textsc{Overlook}_i\Big) \wedge \textsc{Abort}] \leq \Pr[\textsc{Collision}] = \mathbf{Adv}^{\mathrm{tcr}}_{\mathcal{CF},Hfam}(\lambda). \quad \square \tag{8}$$

E Proof of Claim 6

It is enough to prove that

$$\begin{aligned}&\text{When } (e(g,g), e(g,g)^{\alpha_1}, e(g,g)^{\alpha_2}, \hat{Y}, \hat{Z}_1, \hat{Z}_2) \text{ is a twin DH tuple,}\\ &\hat{\kappa} = \hat{Z_1}^{1/(\tau-\tau^*)} = e(g,g)^{\alpha_1 s} \text{ holds.}\end{aligned}$$

This is deduced as follows:

$$\hat{\kappa} = (d_1/e(C',g)^{\mu_1})^{1/(\tau-\tau^*)} = ((e(g,g)^{\alpha_1})^{s(\tau-\tau^*)})^{1/(\tau-\tau^*)} = e(g,g)^{\alpha_1 s}. \quad \square$$

F Proof of Claim 7

A direct calculation with equalities (1) shows:

$$d_i^* = e(C'^*,g)^{\mu_i} = e(g,g)^{s^*(\alpha_i\tau^*+\alpha_{(i+2)})} = e(g,g)^{\alpha_i s^* \tau^*} e(g,g)^{\alpha_{(i+2)} s^*}, i = 1,2.$$

Hence $\psi^* = (\psi^*_{\mathrm{cpa}}, d_1^*, d_2^*)$ is legitimate and correctly distributed. $\square$

G Our Ciphertext-Policy Attribute-Based Encryption Scheme

In this section, we describe our `CP-ABE`. Only differences between `CP-ABKEM` and `CP-ABE` are stated below.

G.1 Our Construction

Setup($\lambda, \mathcal{U}$). The same as Setup of `CP-ABKEM`.

Encrypt(PK, $\mathbb{A}$, m). The same as Encap of `CP-ABKEM` except that Encrypt multiplies m by the blinding factor κ in the group $\mathbb{G}_T$. Encrypt returns $\mathrm{CT} = (C = m\kappa, \psi = (C', ((C_i, D_i); i = 1, \ldots, l), d_1, d_2))$.

KeyGen(MSK, PK, S). The same as KeyGen of `CP-ABKEM`.

Decrypt(PK, CT, SK_S). The same as Decap of `CP-ABKEM` except that Decrypt divides out C by the decapsulated blinding factor $\hat{\kappa}$. Decrypt returns the result $\hat{m}$.

G.2 Security and its Proof

Theorem 2 *If the Waters* $\mathtt{CP\text{-}ABKEM}_{cpa}$ *[Wat11] is selectively secure against chosen-plaintext attacks and an employed hash function family* Hfam *has target collision resistance, then our* $\mathtt{CP\text{-}ABE}$ *is selectively secure against chosen-ciphertext attacks. More precisely, for any given PPT adversary* $\mathcal{A}$ *that attacks* $\mathtt{CP\text{-}ABE}$ *in the IND-sel-CCA game where decryption queries are at most* q_d *times, and for any attribute universe* $\mathcal{U}$*, there exist a PPT adversary* $\mathcal{B}$ *that attacks* $\mathtt{CP\text{-}ABKEM}_{cpa}$ *in the IND-sel-CPA game and a PPT target collision finder* $\mathcal{CF}$ *on* Hfam *that satisfy the following inequality.*

$$\mathbf{Adv}^{ind\text{-}sel\text{-}cca}_{\mathcal{A},\mathtt{CP\text{-}ABE}}(\lambda,\mathcal{U}) \leq 2\big(\mathbf{Adv}^{ind\text{-}sel\text{-}cpa}_{\mathcal{B},\mathtt{CP\text{-}ABKEM}_{cpa}}(\lambda,\mathcal{U}) + \mathbf{Adv}^{tcr}_{\mathcal{CF},Hfam}(\lambda) + \frac{q_d}{p}\big).$$

Proof. Given any adversary $\mathcal{A}$ that attacks our scheme $\mathtt{CP\text{-}ABE}$ in the IND-sel-CCA game, we construct an adversary $\mathcal{B}$ that attacks the Waters KEM $\mathtt{CP\text{-}ABKEM}_{\mathrm{cpa}}$ in the IND-sel-CPA game as follows.

Commit a Target Access Structure. The same as that of $\mathtt{CP\text{-}ABKEM}$.

Set up. In return to outputting $\mathbb{A}^*$, $\mathcal{B}$ receives the public key $\mathrm{PK}_{\mathrm{cpa}}$ for $\mathtt{CP\text{-}ABKEM}_{\mathrm{cpa}}$. To set up a public key PK for $\mathtt{CP\text{-}ABE}$, $\mathcal{B}$ *herein* needs a challenge instance: $\mathcal{B}$ queries its challenger and gets a challenge instance $(\tilde{\kappa}, \psi^*_{\mathrm{cpa}})$. The rest of procedure is the same as that of $\mathtt{CP\text{-}ABKEM}$, and $\mathcal{B}$ inputs PK into $\mathcal{A}$.

Phase 1. The same as that of $\mathtt{CP\text{-}ABKEM}$ except that $\mathcal{B}$ replies a decrypted message $\hat{m}$ to $\mathcal{A}$ for a decryption query.

Challenge. In the case that $\mathcal{A}$ submits two plaintexts (m^*_0, m^*_1) of equal length, $\mathcal{B}$ makes a challenge ciphertext CT^* as follows and feeds CT^* to $\mathcal{A}$.

$$\begin{aligned}&\text{Pick up } b' \leftarrow \{0,1\} \text{ and put } \tilde{C}^* = m^*_{b'}\tilde{\kappa};\\ &\text{Put } d^*_1 = e(C'^*, g)^{\mu_1}, d^*_2 = e(C'^*, g)^{\mu_2};\\ &\text{Put } \mathrm{CT}^* = (\tilde{C}^*, \psi^* = (\psi^*_{\mathrm{cpa}}, d^*_1, d^*_2)).\end{aligned}$$

Phase 2. The same as in Phase 1.

Guess. In the case that $\mathcal{A}$ returns $\mathcal{A}$'s guess $\tilde{b}$, $\mathcal{B}$ returns $\tilde{b}$ as $\mathcal{B}$'s guess.

Evaluation of the Advantage of $\mathcal{B}$. A standard argument deduces a loss of tightness by a factor of 1/2. That is;

$$\mathbf{Adv}^{\text{ind-sel-cpa}}_{\mathcal{B},\mathtt{CP\text{-}ABKEM}_{\mathrm{cpa}}}(\lambda,\mathcal{U}) \geq \frac{1}{2}\mathbf{Adv}^{\text{ind-sel-cca}}_{\mathcal{A},\mathtt{CP\text{-}ABE}}(\lambda,\mathcal{U}) - \frac{q_{\mathrm{d}}}{p} - \mathbf{Adv}^{\mathrm{tcr}}_{\mathcal{CF},Hfam}(\lambda). \quad \square$$

H Target Collision Resistant Hash Functions

Target collision resistant (TCR) hash functions [NY89] are treated as a family. Let us denote a function family as $Hfam(\lambda) = \{H_\mu\}_{\mu \in HKey(\lambda)}$. Here $HKey(\lambda)$ is a hash key space, $\mu \in HKey(\lambda)$ is a hash key and H_μ is a function from $\{0,1\}^*$ to $\{0,1\}^\lambda$. We may assume that H_μ is from $\{0,1\}^*$ to $\mathbb{Z}_p$, where p is a prime of length λ.

Given a PPT algorithm $\mathcal{CF}$, a collision finder, we consider the following experiment (the target collision resistance game).

$\mathbf{Experiment}^{\mathrm{tcr}}_{\mathcal{CF},Hfam}(\lambda)$
$m^* \leftarrow \mathcal{CF}(\lambda), \mu \leftarrow HKey(\lambda), m \leftarrow \mathcal{CF}(\mu)$
If $m^* \neq m \wedge H_\mu(m^*) = H_\mu(m)$ then return Win else return Lose.

Then we define *$\mathcal{CF}$'s advantage over Hfam in the game of target collision resistance* as follows.

$$\mathbf{Adv}^{\mathrm{tcr}}_{\mathcal{CF},Hfam}(\lambda) \stackrel{\mathrm{def}}{=} \Pr[\mathbf{Experiment}^{\mathrm{tcr}}_{\mathcal{CF},Hfam}(\lambda) \text{ returns Win}].$$

We say that *Hfam* is *a TCR function family* if, for any PPT algorithm $\mathcal{CF}$, $\mathbf{Adv}^{\mathrm{tcr}}_{\mathcal{CF},Hfam}(\lambda)$ is negligible in λ.

TCR hash function families can be constructed based on the existence of a one-way function [NY89].