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 -, which was co-held with 2017 IEEE International Conference on Smart Computing, - SMARTCOMP 2017 -* under the title "Short CCA-Secure Ciphertext-Policy Attribute-Based Encryption". This is the full version. A related version of this paper appeared in *Advances in Science, Technology and Engineering Systems Journal (ASTESJ), Volume 3, Issue 1, pp. 261-273, 2018* under the same title.

Short CCA-Secure 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

April 19, 2018

Abstract. Chosen-ciphertext attacks are typical threat on public-key encryption schemes. We propose a technique of individually converting an attribute-based encryption scheme (ABE) which is secure against chosen-plaintext attacks into an ABE scheme which is secure against chosen-ciphertext attacks. Our technique is helpful when a Diffie-Hellman tuple to be verified is in the target group of a bilinear map. The employed technique, the Twin Diffie-Hellman Trapdoor Test of Cash, Kiltz and Shoup, results in expansion of the secret-key length and the decryption cost by a factor of four, while the public-key and the ciphertext lengths and the encryption cost remain almost the same.
Keywords: public-key cryptography, attribute-based encryption, direct chosen-ciphertext security, twin Diffie-Hellman.

1 Introduction

Access control is one of the fundamental processes and requirements in cybersecurity. Attribute-based encryption (ABE) invented by Sahai and Waters [17], where attributes mean authorized credentials, enables to realize access control *by encryption* which is conceptually close to traditional access control methods such as role-based access control (RBAC). In key-policy ABE (KP-ABE) introduced by the subsequent work of Goyal, Pandey, Sahai and Waters [11], a secret key is associated with an access policy over attributes, while a ciphertext is associated with a set of attributes. In a dual manner, in ciphertext-policy ABE (CP-ABE) [11, 3, 19], a ciphertext is associated with access policy over attributes, while a secret key is associated with a set of attributes. In a KP-ABE or CP-ABE scheme, a secret key works to decrypt a ciphertext if and only if the associated set of attributes satisfies the associated access policy. Since the proposals, it has been studied to attain certain properties such

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

as indistinguishability against chosen-plaintext attacks (IND-CPA) in the standard model [19] and adaptive security against adversary's choice of a target access policy [14].

In this paper, we work through resolving 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 choice through adversaries' attacking. Note that "provable security" which means that a cryptographic primitive is rigorously secure in a mathematical model under a certain assumption is a must requirement when we employ the primitive in a system. Moreover, the CCA security of an encryption scheme is preferable to attain because the CCA security is one of theoretically highest securities and hence the scheme can be widely used.

Let us recall the case of identity-based encryption (IBE). The CHK transformation of Canetti, Halevi and Katz [7] 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 [4] into an IBE scheme that is adaptive-ID IND-CCA secure [4]. A point of the CHK transformation is that it introduces a dummy identity $\mathtt{vk}$ that is a verification key of a one-time signature. Then a ciphertext is attached with $\mathtt{vk}$ and a signature σ, which is generated each time one executes encryption. In contrast, the direct chosen-ciphertext security technique for IBE of Boyen, Mei and Waters [6] is an individual modification 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. [20] is a generic tool for obtaining IND-CCA secure ABE scheme. It transforms any ABE scheme (with the delegatability or the verifiability [20]) 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 $\mathtt{vk}$ that is a verification key of a one-time signature. Then a ciphertext is attached with $\mathtt{vk}$ 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 target 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 [19] 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 [8]. In addition to that, we also utilize the algebraic trick of Boneh and Boyen [5] and Kiltz [12] to reply for adversary's decryption query. In total, we develop the technique to realize direct chosen-ciphertext security.

1.2 Related Works

Waters [19] pointed out that IND-CCA security would be attained by the CHK transformation. Gorantla, Boyd and Nieto [10] constructed a IND-CCA secure CP-ABKEM in the random oracle model. Yamada et al. [20] 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 of the CHK transformation, and it is versatile. Especially, it can be applied to non-pairing-based scheme.

The Waters CP-ABE [19] can be captured as a CP-ABKEM: the blinding factor can be considered as a random one-time key. This Waters CP-ABKEM is IND-CPA secure because the Waters CP-ABE is proved to be IND-CPA secure. For theoretical simplicity, we demonstrate an individual conversion of the Waters CP-ABKEM into a CP-ABKEM which is IND-CCA secure. Then we provide a CP-ABE scheme which is IND-CCA secure. As for KP-ABE, we demonstrate an individual conversion of KP-ABKEM of the Goyal, Pandey, Sahai and Waters [11], which is IND-CPA secure, into a KP-ABKEM which is IND-CCA secure. Then we provide a KP-ABE scheme which is IND-CCA secure.

Finally, we note that there is a remarkable work of CP-ABE schemes and KP-ABE schemes with *constant-size* ciphertexts [2, 1].

1.3 Organization of the Paper

In Section 2, we survey concepts, definitions and techniques needed. In Section 3, we revisit the concept, the algorithm and the security of the twin Diffie-Hellman technique. In Section 4, we construct a CCA-secure CP-ABKEM from the Waters CPA-secure CP-ABKEM [19], and provide a security proof. Also, we describe the encryption version, a CCA-secure CP-ABE. In Section 5, we construct a CCA-secure KP-ABKEM from the Ostrovsky-Sahai-Waters CPA-secure KP-ABKEM [16], and provide a security proof. Also, we describe the encryption version, a CCA-secure KP-ABE. In Section 6, we compare efficiency of our CP-ABE and KP-ABE schemes with the original schemes, and also, with the schemes obtained by applying the generic transformation [20] to the original schemes. In Section 7, we conclude our work.

2 Preliminaries

The security parameter is denoted by λ. A prime of bit length λ is denoted by p. A multiplicative cyclic group of order p is denoted by $\mathbb{G}$. Parameters of the group $\mathbb{G}$ are generated by a probabilistic polynomial time (PPT) algorithm $\mathtt{Grp}$ on input λ: $(p, \mathbb{G}, g) \leftarrow \mathtt{Grp}(\lambda)$, where g is a generator of $\mathbb{G}$. The ring of exponent domain of $\mathbb{G}$, which consists of integers from 0 to $p-1$ with modulo p operation, is denoted by $\mathbb{Z}_p$.

2.1 Bilinear Map

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of a 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 $\mathbb{G}_T$).

The groups $\mathbb{G}$ and $\mathbb{G}_T$ with the bilinear map e is called bilinear groups. Parameters of the bilinear groups are generated by a PPT algorithm $\mathtt{BlGrp}$ on input λ: $(p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathtt{BlGrp}(\lambda)$. 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 efficiently computable (i.e. PPT in λ).

We note here that bilinear groups of an asymmetric pairing of Type 3 [9] are more efficient than the above bilinear groups of a symmetric pairing. Here Type 3 means an asymmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ without any efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. For simplicity, we describe in this paper only the case of symmetric pairing, and we just note the case of asymmetric pairing in efficiency discussion in Section 6.

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 known 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 Attribute-Based Key Encapsulation Mechanism

Ciphertext-policy attribute-based key encapsulation mechanism (CP-ABKEM). A 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$(\mathbf{PK}, \mathbb{A})$**.** 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. Note that $\mathbb{A}$ is contained in ψ.
KeyGen$(\mathbf{PK}, \mathbf{MSK}, S)$**.** 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. Note that S is contained in SK_S.
Decap$(\mathbf{PK}, \mathbf{SK}_S, \psi)$**.** 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 [10]), the chosen-ciphertext attack on a CP-ABKEM is formally defined as the indistinguishability game (IND-CCA game). In this paper, we consider the *selective game on a target access structure* (IND-sel-CCA game); that is, 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.

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

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

[3] In Gorantla, Boyd and Nieto [10], they say *encapsulation-policy* attribute-based-KEM (EP-AB-KEM) instead of saying ciphertext-policy attribute-based KEM here.

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 implicitly 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 $\mathbb{A}^*$ declared 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 $S_j \notin \mathbb{A}^* \vee \psi_j \neq \psi^*$.

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

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

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

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

Ciphertext-Policy Attribute-Based Encryption Scheme (CP-ABE). In the case of the encryption version (i.e. 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.

Key-Policy Attribute-Based Key Encapsulation Mechanism (KP-ABKEM) and Encryption Scheme (KP-ABE). The *key-policy* case is analogously defined as the case of the ciphertext-policy case. We state only the syntax and the security experiment of the key-policy ABKEM.

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$(\mathbf{PK}, S)$**.** An encapsulation algorithm Encap takes as input the public key PK and an attribute set S. It returns a random string κ and its encapsulation ψ. Note that S is contained in ψ.

KeyGen$(\mathbf{PK}, \mathbf{MSK}, \mathbb{A})$**.** A key generation algorithm KeyGen takes as input the public key PK, the master secret key MSK and an access structure $\mathbb{A}$. It returns a secret key $\mathrm{SK}_\mathbb{A}$ corresponding to S. Note that $\mathbb{A}$ is contained in $\mathrm{SK}_\mathbb{A}$.

Decap$(\mathbf{PK}, \mathbf{SK}_\mathbb{A}, \psi)$**.** 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 $\mathrm{SK}_\mathbb{A}$. It first checks whether $S \in \mathbb{A}$. If the check result is False, it puts $\hat{\kappa} = \perp$. It returns a decapsulation result $\hat{\kappa}$.

[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 explicitly.

Chosen-Ciphertext Attack on KP-ABKEM. The *selective game on a target attribute set* (IND-sel-CCA game) is defined by the following experiment.

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

2.5 Target Collision Resistant Hash Functions

Target collision resistant (TCR) hash functions [15] are treated as a family. Let us denote a function family as $\mathit{Hfam}(\lambda) = \{H_\mu\}_{\mu \in \mathit{HKey}(\lambda)}$. Here $\mathit{HKey}(\lambda)$ is a hash key space, $\mu \in \mathit{HKey}(\lambda)$ 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).

$$
\begin{aligned}
&\mathbf{Experiment}^{\text{tcr}}_{\mathit{Hfam},\mathcal{CF}}(\lambda)\\
&m^* \leftarrow \mathcal{CF}(\lambda), \mu \leftarrow \mathit{HKey}(\lambda), m \leftarrow \mathcal{CF}(\mu)\\
&\text{If } m^* \neq m \wedge H_\mu(m^*) = H_\mu(m)\\
&\quad \text{then return } \textsc{Win} \text{ else return } \textsc{Lose}.
\end{aligned}
$$

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

$$
\begin{aligned}
&\mathbf{Adv}^{\text{tcr}}_{\mathit{Hfam},\mathcal{CF}}(\lambda)\\
&\stackrel{\text{def}}{=} \Pr[\mathbf{Experiment}^{\text{tcr}}_{\mathit{Hfam},\mathcal{CF}}(\lambda) \text{ returns } \textsc{Win}].
\end{aligned}
$$

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

TCR hash function families can be constructed based on the existence of a one-way function [15].

3 The Twin Diffie-Hellman Technique Revisited

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 [8] will be used in the security proof to decide whether a tuple is a twin DH tuple or not.

Lemma 1 ("Trapdoor Test" [8]) *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.

$$
\begin{aligned}
&\mathbf{Experiment}^{\text{twinDH-test}}_{\mathtt{Grp},\mathit{Cheat}}(\lambda)\\
&\quad (p, \mathbb{G}, g_0) \leftarrow \mathtt{Grp}(\lambda)\\
&\quad (g, X_1) \leftarrow \mathbb{G}^2, (r, s) \leftarrow \mathbb{Z}_p^2, X_2 = X_1^{-r} g^s\\
&\quad \mathbb{G}^3 \ni (\hat{Y}, \hat{Z}_1, \hat{Z}_2) \leftarrow \mathit{Cheat}(g, X_1, X_2)\\
&\quad \text{If } \hat{Z}_1{}^r \hat{Z}_2 = \hat{Y}^s \wedge (g, X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2) \text{ is NOT a twin DH tuple,}\\
&\qquad \text{then return } \textsc{Win} \text{ else return } \textsc{Lose}
\end{aligned}
$$

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

$$
\begin{aligned}
&\mathbf{Adv}^{\text{twinDH-test}}_{\mathtt{Grp},\mathit{Cheat}}(\lambda)\\
&\stackrel{\text{def}}{=} \Pr[\mathbf{Experiment}^{\text{twinDH-test}}_{\mathtt{Grp},\mathit{Cheat}}(\lambda) \text{ returns } \textsc{Win}].
\end{aligned}
$$

Now we are ready to complement Lemma 1.

Lemma 2 (Complement for "Trapdoor Test" [8])
For any algorithm Cheat *with unbounded computational power,* $\mathbf{Adv}^{\mathit{twinDH\text{-}test}}_{\mathtt{Grp},\mathrm{Cheat}}(\lambda)$ *is at most* $1/p$.

For a proof of Lemma 2, see Appendix A.

We remark here that in the succeeding sections we use "Trapdoor Test" for the target group $\mathbb{G}_T$ of bilinear groups generated as $(p, \mathbb{G}, \mathbb{G}_T, g_0, e) \leftarrow \mathtt{BlGrp}(\lambda)$.

4 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 [19].

Overview of Our Technique The Waters CP-ABE is proved to be secure in the IND-sel-CPA game [19]. 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 [8] and the algebraic trick of Boneh and Boyen [5] and Kiltz [12].

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 are unable to use the bilinear map instead because the tuple to be verified is in the target 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.

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 $\mathtt{CP\text{-}ABKEM}_{\mathrm{cpa}}$. Likewise, we distinguish parameters and algorithms of $\mathtt{CP\text{-}ABKEM}_{\mathrm{cpa}}$ by the index $_{\mathrm{cpa}}$. For theoretical simplicity, we first develop a KEM `CP-ABKEM`.

4.1 Our Construction

Our `CP-ABKEM` consists of the following four PPT algorithms (Setup, Encap, KeyGen, Decap). Roughly speaking, the Waters original scheme $\mathtt{CP\text{-}ABKEM}_{\mathrm{cpa}}$ (the first scheme in [19]) 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 $\mathtt{BlGrp}(\lambda)$ to get bilinear groups $(p, \mathbb{G}, \mathbb{G}_T, g, e)$. These parameters become public. 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$(\mathbf{PK}, \mathbb{A})$**.** 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 index 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 randomness, and chooses 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.

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

KeyGen$(\mathbf{MSK}, \mathbf{PK}, S)$**.** 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 generates the secret key SK_S as follows.

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

Decap$(\mathbf{PK}, \psi, \mathbf{SK}_S)$**.** The decapsulation algorithm Decap takes as input the public key PK, an encapsulation ψ for an 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 $\hat{\kappa} = \perp$. Otherwise, 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.

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

4.2 Security and Proof

Theorem 1 *If the Waters* $\mathtt{CP\text{-}ABKEM}_{cpa}$ *[19] is selectively secure against chosen-plaintext attacks and an employed hash function family* Hfam *has target collision resistance, then our* $\mathtt{CP\text{-}ABKEM}$ *is selectively secure against chosen-ciphertext attacks. More precisely, for any given PPT adversary* $\mathcal{A}$ *that attacks* $\mathtt{CP\text{-}ABKEM}$ *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* $\mathtt{CP\text{-}ABKEM}_{cpa}$ *in the IND-sel-CPA game and a PPT target collision finder* $\mathcal{CF}$ *on* Hfam *that satisfy the following tight reduction.*

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

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 $\text{CP-ABKEM}_{\text{cpa}}$ 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 PK_{cpa} for $\text{CP-ABKEM}_{\text{cpa}}$, which consists of the following components.

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

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 $(\tilde{\kappa}, \psi^*_{\text{cpa}})$. It consists of the following components.

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

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

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

Note we have implicitly set relations in the exponent domain:

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

A public key PK for CP-ABKEM become:

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

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 $\text{CP-ABKEM}_{\text{cpa}}$ for a single attribute set S:

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

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

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

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
  $\tau \leftarrow H_\eta(\psi_{\mathrm{cpa}})$;
  $\hat{Y} = e(C', g)^{\tau - \tau^*}, \hat{Z}_1 = d_1 / e(C', g)^{\mu_1}, \hat{Z}_2 = d_2 / e(C', g)^{\mu_2}$;
  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.

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

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:

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

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:*

$$\begin{cases} \hat{Z_1}^{\gamma_1}\hat{Z_2} = \hat{Y}^{\gamma_2} \text{ holds} \\ \text{and} \\ (e(g,g), e(g,g)^{\alpha_1}, e(g,g)^{\alpha_2}, \hat{Y}, \hat{Z_1}, \hat{Z_2}) \text{ is NOT a twin DH tuple.} \end{cases}$$

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 *Cheat* 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.

Given $(e(g,g), e(g,g)^{\alpha_1}, e(g,g)^{\alpha_2})$ as input :
Set up
Initialize the inner state, put TABLE $= \phi$;
Get a target access structure $\mathbb{A}^* \leftarrow \mathcal{A}(\lambda, \mathcal{U})$;
Compute the base $g \in \mathbb{G}$ from $(e(g,g), e)$;
Choose $a \in \mathbb{Z}_p$ 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 $(\kappa^*, \psi^*_{\mathrm{cpa}}) \leftarrow \mathbf{Encap}_{\mathrm{cpa}}(\mathrm{PK}_{\mathrm{cpa}}, \mathbb{A}^*)$;
Choose $\eta \leftarrow \mathit{HKey}(\lambda)$ 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 $e(g,g)$;
Choose $\mu_1, \mu_2 \leftarrow \mathbb{Z}_p$, 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 $\mathcal{A}$ issues 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 $\mathcal{A}$ issues 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 $\mathcal{A}$ issues a challenge instance query;
Put $d_1^* = e(C'^*, g)^{\mu_1}, d_2^* = e(C'^*, g)^{\mu_2}, \psi^* = (\psi^*_{\mathrm{cpa}}, d_1^*, d_2^*)$;
Choose $\kappa \leftarrow \mathit{KeySp}(\lambda), b \leftarrow \{0,1\}$;
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 *Cheat* with unbounded computational power for a proof of Claim 4.

First, note that the view of $\mathcal{A}$ in *Cheat* 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 *Cheat* is randomized in TABLE. Hence:

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

We note here that the advantage $\mathbf{Adv}^{\text{twinDH-test}}_{\mathtt{Grp}, \mathit{Cheat}}(\lambda)$ of *Cheat* is evaluated according to "Trapdoor Test" in Section 3 for the *target group* $\mathbb{G}_T$ of bilinear groups $(p, \mathbb{G}, \mathbb{G}_T, g, e)$ generated by $\mathtt{BlGrp}(\lambda)$. In this case the group generation algorithm $\mathtt{Grp}$ is the following one.

$$\mathtt{Grp}(\lambda) : (p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathtt{BlGrp}(\lambda), g_T := e(g, g), \text{Return } (p, \mathbb{G}_T, g_T).$$

Third, applying Lemma 2 to *Cheat*, we get:

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

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

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

□

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

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

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 two claims are proved by a direct calculation. See Appendices E and F, respectively. □

Evaluation of the Advantage of $\mathcal{B}$. 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:

$$\begin{aligned}&\mathbf{Adv}^{\text{ind-sel-cpa}}_{\texttt{CP-ABKEM}_{\text{cpa}},\mathcal{B}}(\lambda,\mathcal{U})\\ \geq&\mathbf{Adv}^{\text{ind-sel-cca}}_{\texttt{CP-ABKEM},\mathcal{A}}(\lambda,\mathcal{U})-\frac{q_{\text{d}}}{p}-\mathbf{Adv}^{\text{tcr}}_{Hfam,\mathcal{CF}}(\lambda).\end{aligned}$$

This is what we should prove in Theorem 1. □

4.3 Encryption Version from KEM.

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* $\texttt{CP-ABKEM}_{\text{cpa}}$.
Setup$(\lambda,\mathcal{U})$. The same as Setup of CP-ABKEM.
Encrypt$(\text{PK},\mathbb{A},m)$. 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 $\text{CT}=(C=m\kappa,\psi=(C',((C_i,D_i);i=1,\ldots,l),d_1,d_2))$.
KeyGen(MSK,PK,S). The same as KeyGen of CP-ABKEM.
Decrypt$(\text{PK},\text{CT},\text{SK}_S)$. The same as Decap of CP-ABKEM except that Decrypt divides out C by the decapsulated blinding factor $\hat{\kappa}$. Decrypt returns the result $\hat{m}$.

4.4 Security and Proof

Theorem 2 *If the Waters* $\texttt{CP-ABKEM}_{cpa}$ *[19] is selectively secure against chosen-plaintext attacks and an employed hash function family Hfam has target collision resistance, then our* CP-ABE *is selectively secure against chosen-ciphertext attacks. More precisely, for any given PPT adversary* $\mathcal{A}$ *that attacks* CP-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* $\texttt{CP-ABKEM}_{cpa}$ *in the IND-sel-CPA game and a PPT target collision finder* $\mathcal{CF}$ *on Hfam that satisfy the following inequality.*

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

Proof. Given any adversary $\mathcal{A}$ that attacks our scheme CP-ABE in the IND-sel-CCA game, we construct an adversary $\mathcal{B}$ that attacks the Waters KEM $\texttt{CP-ABKEM}_{\text{cpa}}$ in the IND-sel-CPA game as follows.
Commit to a Target Access Structure. The same as that of CP-ABKEM.
Set up. In return to outputting $\mathbb{A}^*$, $\mathcal{B}$ receives the public key PK_{cpa} for $\texttt{CP-ABKEM}_{\text{cpa}}$. To set up a public key PK for CP-ABE, $\mathcal{B}$ *herein* needs a challenge instance: $\mathcal{B}$ queries its challenger and gets a challenge instance $(\tilde{\kappa},\psi^*_{\text{cpa}})$. The rest of procedure is the same as that of CP-ABKEM, and $\mathcal{B}$ inputs PK into $\mathcal{A}$.
Phase 1. The same as that of CP-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 $\mathcal{A}$.

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

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;

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

□

5 Securing the Ostrovsky-Sahai-Waters KP-ABKEM against Chosen-Ciphertext Attacks

In this section, we describe our direct chosen-ciphertext security technique by applying it to the Ostrovsky-Sahai-Waters KP-ABE [16].

Overview of Our Technique The Ostrovsky-Sahai-Waters KP-ABE is proved to be secure in the IND-sel-CPA game [16]. 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 [8] and the algebraic trick of Boneh and Boyen [5] and Kiltz [12].

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 are unable to use the bilinear map instead because the tuple to be verified is in the target 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.

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

5.1 Our Construction

Our KP-ABKEM consists of the following four PPT algorithms (Setup, Encap, KeyGen, Decap). Roughly speaking, the Ostrovsky-Sahai-Waters original scheme $\texttt{KP-ABKEM}_{\mathrm{cpa}}$ (the first scheme in [16]) 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 $\texttt{BlGrp}(\lambda)$ to get bilinear groups $(p, \mathbb{G}, \mathbb{G}_T, g, e)$. These parameters become public. 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 $\eta \in \mathit{HKey}(\lambda)$. The public key is published as $\mathrm{PK} = (g, g^a, h_1, \ldots, h_u, e(g,g)^{a\alpha_1}, \ldots, e(g,g)^{a\alpha_4}, \eta)$. The authority sets $\mathrm{MSK} = (\alpha_1, \ldots, \alpha_4)$ as the master secret key.

Encap$(\mathbf{PK}, S)$**.** The encapsulation algorithm Encap takes as input the public key PK and a set S of attributes. Encap first chooses a random value $s \in \mathbb{Z}_p$ that is the encryption randomness. Then, a pair of a random one-time key and its encapsulation (κ, ψ) is computed as follows.

$$
\begin{aligned}
&\text{Put } C' = g^s; \text{For } x \in S : C_x = h_x^s \\
&\psi_{\mathrm{cpa}} = (S, C', (C_x; x \in S)), \tau \leftarrow H_\eta(\psi_{\mathrm{cpa}}); \\
&\text{For } k = 1 \text{ to } 4 : \kappa_k = e(g,g)^{a\alpha_k s}; d_1 = \kappa_1^\tau \kappa_3, d_2 = \kappa_2^\tau \kappa_4; \\
&(\kappa, \psi) = (\kappa_1, (\psi_{\mathrm{cpa}}, d_1, d_2)).
\end{aligned}
$$

KeyGen$(\mathbf{MSK}, \mathbf{PK}, \mathbb{A})$**.** The key generation algorithm KeyGen takes as input the master secret key MSK, 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 index i of M to an attribute in $\mathcal{U} = \{1, \ldots, u\}$. For $k = 1$ to 4, KeyGen first chooses random values $y_{k,2}, \ldots, y_{k,n} \in \mathbb{Z}_p$ and forms a vector $\boldsymbol{v}_k = (\alpha_k, y_{k,2}, \ldots, y_{k,n})$. Then, for $i = 1$ to l, it calculates $\lambda_{k,i} = \boldsymbol{v}_k \cdot M_i$, where M_i denotes the i-th row vector of M, and it chooses random values $r_{k,i} \in \mathbb{Z}_p$. KeyGen generates the secret key $\mathrm{SK}_{\mathbb{A}}$ as follows.

$$
\begin{aligned}
&\text{For } k = 1 \text{ to } 4 : \text{For } l = 1 \text{ to } l : \\
&\quad K_{k,i} = g^{a\lambda_{k,i}} h_{\rho(i)}^{r_{k,i}}, L_{k,i} = g^{r_{k,i}} \\
&\mathrm{SK}_{\mathbb{A}} = (((K_{k,i}, L_{k,i}); i = 1, \ldots, l) k = 1, \ldots, 4).
\end{aligned}
$$

Decap$(\mathbf{PK}, \psi, \mathbf{SK}_{\mathbb{A}})$**.** The decapsulation algorithm Decap takes as input the public key PK, an encapsulation ψ for an attribute set S and a private key $\mathrm{SK}_{\mathbb{A}}$ for an access structure $\mathbb{A} = (M, \rho)$. It first checks whether $S \in \mathbb{A}$. If the result is FALSE, put $\hat{\kappa} = \perp$. Otherwise, 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.

Parse ψ into $(\psi_{\mathrm{cpa}} = (S, C', (C_x; x \in S)), d_1, d_2)$;
$\tau \leftarrow H_\eta(\psi_{\mathrm{cpa}})$;
For $k = 1$ to 4 :
$\hat{\kappa_k} = \prod_{i \in I_S} (e(C', K_{k,i})/e(L_{k,i}, C_{\rho(i)}))^{\omega_i} = e(g,g)^{a\alpha_k s}$
If $\hat{\kappa_1}^{\tau}\hat{\kappa_3} \neq d_1 \vee \hat{\kappa_2}^{\tau}\hat{\kappa_4} \neq d_2$,
then put $\hat{\kappa} = \perp$, else put $\hat{\kappa} = \hat{\kappa_1}$.

5.2 Security and Proof

Theorem 3 *If the Ostrovsky-Sahai-Waters* KP-ABKEM$_{cpa}$ *[16] is selectively secure against chosen-plaintext attacks and an employed hash function family* Hfam *has target collision resistance, then our* KP-ABKEM *is selectively secure against chosen-ciphertext attacks. More precisely, for any given PPT adversary* $\mathcal{A}$ *that attacks* KP-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* KP-ABKEM$_{cpa}$ *in the IND-sel-CPA game and a PPT target collision finder* $\mathcal{CF}$ *on* Hfam *that satisfy the following tight reduction.*

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

Proof. We will omit the description of the proof because the proof goes analogously to the case of CP-ABKEM in Section 4.2. □

5.3 Encryption Version from KEM.

It is straightforward to construct our encryption scheme KP-ABE from KP-ABKEM. The IND-sel-CCA security of KP-ABE is proved based on IND-sel-CPA security of the Waters *KEM* KP-ABKEM$_{\mathrm{cpa}}$.
Setup$(\lambda, \mathcal{U})$**.** The same as Setup of KP-ABKEM.
Encrypt$(\mathbf{PK}, \mathbb{A}, m)$**.** The same as Encap of KP-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$(\mathbf{MSK}, \mathbf{PK}, S)$**.** The same as KeyGen of KP-ABKEM.
Decrypt$(\mathbf{PK}, \mathbf{CT}, \mathbf{SK}_S)$**.** The same as Decap of KP-ABKEM except that Decrypt divides out C by the decapsulated blinding factor $\hat{\kappa}$. Decrypt returns the result $\hat{m}$.

5.4 Security and Proof

Theorem 4 *If the Ostrovsky-Sahai-Waters* KP-ABKEM$_{cpa}$ *[16] is selectively secure against chosen-plaintext attacks and an employed hash function family* Hfam *has target collision resistance, then our* KP-ABE *is selectively secure against chosen-ciphertext attacks. More precisely, for any given PPT adversary* $\mathcal{A}$ *that attacks* KP-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* KP-ABKEM$_{cpa}$ *in the IND-sel-CPA game and a PPT target collision finder* $\mathcal{CF}$ *on* Hfam *that satisfy the following inequality.*

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

Proof. We will omit the description of the proof because the proof goes in the same way as the case of CP-ABE in Section 4.4. □

Table 1. Efficiency comparison of IND-sel-CCA secure ABKEMs with the original IND-sel-CPA secure ABKEMs [19, 16].

Scheme	L(PK)	L(SK$_S$)	L(CT)	C(Enc)	C(Dec)
Generic transform [20], CP-ABE	$+4\lambda^2(\mathbb{G})$	$+4\lambda^2(\mathbb{G})$	$+3\lambda^2$(bit)	$+2\lambda^2$exp.($\mathbb{G}$)	$+2\lambda^2$pair.(e)
Our individual modification (CP-ABE)	$+3(\mathbb{G}_T)$	$\times 4$	$+2(\mathbb{G}_T)$	$+4$exp.($\mathbb{G}_T$)	$\times 4$
Generic transform [20], KP-ABE	$+4\lambda^2(\mathbb{G})$	$+0$	$+3\lambda^2$(bit)	$+2\lambda^2$exp.($\mathbb{G}$)	$+2\lambda^2$pair.(e)
Our individual modification (KP-ABE)	$+3(\mathbb{G}_T)$	$\times 4$	$+2(\mathbb{G}_T)$	$+4$exp.($\mathbb{G}_T$)	$\times 4$

1) λ is the security parameter. (For instance, $\lambda = 224$ or 256.)
2) L(data) denotes length of data, C(algorithm) denotes computational amount of algorithm.
3) $+$ and $\times$ mean increment and multiplier to the length or computational amount of the Waters CP-ABKEM$_{\mathrm{cpa}}$.
4) ($\mathbb{G}$), ($\mathbb{G}_T$) and (bit) mean elements in $\mathbb{G}$, elements in $\mathbb{G}_T$ and bits, respectively.
5) 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.

6 Efficiency Discussion

First, we remark that our individual modification to attain CCA security is applicable when a Diffie-Hellman tuple to be verified is in the target group of a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Especially, it is applicable even when an original CPA secure scheme is based on asymmetric pairing [9], $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. For example, the Type 3 version [9] of the Waters CP-ABE scheme [19] can be found in [21]. Detailed discussions and results on real implementations are found for the case of CPA-secure ABE schemes [18, 21]. We note that the efficiency comparison below enables to guess the implementation results of our modification technique.

We compare efficiency of our CP-ABE to the original Waters CP-ABE$_{\mathrm{cpa}}$, and our KP-ABE to the original Ostrovsky-Sahai-Waters KP-ABE$_{\mathrm{cpa}}$. We also compare efficiency of the CP-ABE and KP-ABE obtained by the generic transformation of Yamada et al. [20]. Here the generic transformation [20] is considered in the case of a small attribute universe [11], the delegation type [20] and the Lamport one-time signature [13]. Table 1 shows these comparison. Note that a hash function is applied to generate a message digest of bit-length λ before signing by a secret key of the one-time signature. Note also, for simplicity, we evaluate the lengths and the amounts of computation below in the case that an access structure $\mathbb{A}$ is "all-AND" and an attribute map ρ is injective (i.e. "single-use" that is opposed to "multi-use").

Our individual modification results in expansion of the length of a secret-key and the amount of decryption computation by a factor of four, while the length of a public-key, the length of a ciphertext and the amount of encryption computation are almost the same as those of the original CPA-secure schemes. In the case that the size of an attribute set is up to ($\frac{2}{3}$ of) the square of the security parameter λ, the amount of decryption computation of our CP-ABE and KP-ABE are smaller than those of the CP-ABE and KP-ABE obtained by the generic transformation [20], respectively.

7 Conclusion

We developed a technique of direct chosen-ciphertext security for ABE in the standard model in the case of the Waters scheme (CP-ABKEM$_{\mathrm{cpa}}$, CP-ABE$_{\mathrm{cpa}}$) and the Ostrovsky-Sahai-Waters scheme (KP-ABKEM$_{\mathrm{cpa}}$, KP-ABE$_{\mathrm{cpa}}$). We utilized the Twin Diffie-Hellman Trapdoor Test of Cash, Kiltz and Shoup and the algebraic trick of Boneh and Boyen [5] and Kiltz [12]. Our technique is helpful when a Diffie-Hellman tuple to be verified is in the target group of the 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 the original CPA secure schemes. In comparison with the versatile generic transformation, the amount of decryption computation of our individual modification is smaller when the size of an attribute set is up to about the square of the security parameter λ.

References

1. N. Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 557–577, 2014.
2. N. Attrapadung, B. Libert, and E. 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.
3. J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pages 321–334, 2007.
4. D. Boneh and X. 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.
5. D. Boneh and X. 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.
6. X. Boyen, Q. Mei, and B. 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.
7. R. Canetti, S. Halevi, and J. 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.
8. D. Cash, E. Kiltz, and V. 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.
9. S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
10. M. C. Gorantla, C. Boyd, and J. M. G. 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.
11. V. Goyal, O. Pandey, A. Sahai, and B. 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.
12. E. 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.
13. L. Lamport. Constructing digital signatures from a one-way function. Technical report, Oct. 1979.
14. A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. 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.
15. M. Naor and M. 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.
16. R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 195–203, 2007.
17. A. Sahai and B. 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.
18. A. H. Sánchez and F. Rodríguez-Henríquez. NEON implementation of an attribute-based encryption scheme. In *Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings*, pages 322–338, 2013.

19. B. 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.
20. S. Yamada, N. Attrapadung, G. Hanaoka, and N. 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.
21. E. Zavattoni, L. J. D. Perez, S. Mitsunari, A. H. Sánchez-Ramírez, T. Teruya, and F. Rodríguez-Henríquez. Software implementation of an attribute-based encryption scheme. *IEEE Trans. Computers*, 64(5):1429–1441, 2015.

Appendix

A Proof of Lemma 2

The only one point to be complemented to the original proof (in [8]) 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:

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

□

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. (Note 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.

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:

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

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 $\tau^* \leftarrow H_\eta(\psi^*_{\mathrm{cpa}})$;
Choose $\alpha_2, \alpha_3, \alpha_4 \leftarrow \mathbb{Z}_p$;
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 $d_1^* = e(g^{\alpha_1}, C'^*)^{\tau^*} e(g^{\alpha_3}, C'^*), d_2^* = e(g^{\alpha_2}, C'^*)^{\tau^*} e(g^{\alpha_4}, C'^*)$,
$\psi^* = (\psi^*_{\mathrm{cpa}}, d_1^*, d_2^*)$;
Choose $\kappa \leftarrow \mathit{KeySp}(\lambda), b \leftarrow \{0,1\}$;
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.

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 $\textsc{Overlook}_i$ 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 $\textsc{Overlook}_i$ 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:

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

□

E Proof of Claim 6

It is enough to prove that

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

This is deduced as follows:

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

□

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. □