Short CCA-Secure Attribute-Based Encryption

Hiroaki Anada¹, Seiko Arita²

¹Department of Information Security, University of Nagasaki, 851–2195, Japan
²Graduate School of Information Security, Institute of Information Security, 221–0835, Japan

ARTICLE INFO

Article history:
Received: 16 November, 2017
Accepted: 16 January, 2018
Online: 31 January, 2018

Keywords:
Role-based access control
Attribute-based encryption
Direct chosen-ciphertext security
Twin Diffie-Hellman

ABSTRACT

Chosen-ciphertext attacks (CCA) are typical threat on public-key encryption schemes. We show direct chosen-ciphertext security modification in the case of attribute-based encryption (ABE), where an ABE scheme secure against chosen-plaintext attacks (CPA) is converted into an ABE scheme secure against CCA by individual techniques. Our modification works in the setting that the Diffie-Hellman tuple to be verified in decryption is in the target group of a bilinear map. The employed techniques result 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.

1 Introduction

Access control is one of the fundamental processes and requirements in cybersecurity. Attribute-based encryption (ABE) invented by Sahai and Waters [1], where attributes mean authorized credentials, enables to realize access control which is functionally close to role-based access control (RBAC), but by encryption. In key-policy ABE (KP-ABE) introduced by the subsequent work of Goyal, Pandey, Sahai and Waters [2], 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) [2][3][4], a ciphertext is associated with an 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. The remarkable feature of ABE is attribute privacy; that is, in decryption, no information about the access policy and the identity of the secret key owner in the case of KP-ABE (or, the attributes and the identity of the secret key owner in the case of CP-ABE) leaks except the fact that the set of attributes satisfies the access policy. Since the proposals, it has been studied to attain certain properties such as indistinguishability against chosen-plaintext attacks (IND-CPA) in the standard model [3] and adaptive security against adversary’s choice of a target access policy [5].

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 collect decryption results of ciphertexts of its choice through adversaries’ attacking. Note that “provable security” of a cryptographic primitive is now a must requirement when we employ the primitive in a system, where it means that an appropriately defined security is polynomially reduced to the hardness of a computational problem. Moreover, the CCA security of an encryption scheme is preferable to attain because the CCA security is one of the theoretically highest securities and hence the scheme can be used widely.

To capture the idea of our approach, 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 [8] into an IBE scheme that is adaptive-ID IND-CCA secure [8]. 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, the direct chosen-ciphertext security technique for IBE of

---

¹Hiroaki Anada, 1–1–1, Manabino, Nagayo-cho, Nishisonogi-gun, Nagasaki, +81-95-813-5113 & anada@sun.ac.jp

²This paper is an extension of the work originally presented in SMARTCOMP 2017 [5]. The schemes of KP-ABKEM and KP-ABE have been newly proposed.

https://dx.doi.org/10.25046/aj030132
Boyen, Mei and Waters [9] is 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 in [10] is a generic tool for obtaining IND-CCA secure ABE scheme. It transforms any ABE scheme (with the delegatability or the verifiability [10]) 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 $\sigma$. Notice here that discussing direct chosen-plaintext security modification for ABE (in the standard model) is a missing piece. One of the reasons 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 contribution is that we fill in the missing piece; we demonstrate direct chosen-plaintext security modification in the case of the Waters CP-ABE scheme [4] and the KP-ABE scheme of Ostrovsky, Sahai and Waters [11]. To overcome the above obstacle, we employ the technique of the Twin Diffie-Hellman Trapdoor Test of Cash, Kiltz and Shoup [12]. In addition, we also utilize the algebraic trick of Boneh and Boyen [13] and Kiltz [14] to reply for adversary’s decryption queries.

1.2 Related Works

Waters [4] pointed out that IND-CCA security would be attained by the CHK transformation. Gorantla, Boyd and Nieto [15] constructed a IND-CCA secure CP-ABKEM in the random oracle model. In [10], the authors 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 [4] can be captured as a CP-ABKEM: the binding 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 Ostrovsky, 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 [16][17]. Our direct chosen-ciphertext security modification is not constant-size ciphertexts but a different approach for easier implementation in engineering.

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 CPA-secure CP-ABKEM from the Waters CPA-secure CP-ABKEM [4], and provide a security proof. Also, we describe the encryption version, a CPA-secure CP-ABE. In Section 5, we construct a CPA-secure KP-ABKEM from the Ostrovsky-Sahai-Waters CPA-secure KP-ABKEM [11], and provide a security proof. Also, we describe the encryption version, a CPA-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 [10] to the original schemes. In Section 7, we conclude our work.

2 Preliminaries

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

2.1 Bilinear Map

We remark first that our description in the subsequent sections is in the setting of a symmetric bilinear map for simplicity, but we can employ an asymmetric bilinear map instead for better efficiency as is noted in Section 6. Let $G$ and $G_T$ be two multiplicative cyclic groups of prime order $p$. Let $g$ be a generator of $G$ and $e$ be a bilinear map, $e : G \times G \rightarrow G_T$. The bilinear map $e$ has the following properties:

1. Bilinearity: for all $u, v \in G$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq id_{G_T}$ ( $\cdot$ the identity element of the group $G_T$).

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

Hereafter we assume that the group operation in $G$ and $G_T$ and the bilinear map $e : G \times G \rightarrow G_T$ are computable in PT in $\lambda$. 

www.astesj.com
2.2 Access Structure

Let \( U = \{x_1, \ldots, x_u\} \) be a set of attributes, or simply set \( U = \{1, \ldots, u\} \) by numbering. An access structure, which corresponds to an access policy, is defined as a collection \( A \) of non-empty subsets of \( U \); that is, \( A \subseteq 2^U \setminus \emptyset \). An access structure \( A \) is called monotone if for any \( B \in A \) and \( B \subset C \), \( C \in A \) holds. The sets in \( A \) are called authorized sets, and the sets not in \( A \) 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 \( \Pi \) over the attribute universe \( U \) 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 \( \Pi \) and a function \( \rho \) which maps each row index \( i \) of \( M \) to an attribute in \( U = \{1, \ldots, u\} : \rho : \{1, \ldots, l\} \to U \).

To make shares, we first choose a random vector \( \vec{v} = (s, v_2, \ldots, v_n) \in \mathbb{Z}_p^n \) such that \( s \) is a secret to be shared. For \( i = 1 \) to \( l \), we calculate each share \( \lambda_i = \vec{v} \cdot M_i \), where \( M_i \) denotes the \( i \)-th row vector of \( M \) and \( \cdot \) denotes the inner product. LSSS \( \Pi = (M, \rho) \) defines an access structure \( A \) through \( \rho \).

Suppose that an attribute set \( S \) satisfies \( A \in \mathcal{A} \) 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 \( l \) that the attribute universe \( M \). If \( S \) does not satisfy \( A \) \( S \notin \mathcal{A} \), 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).\(^2\)

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

\( \text{Encap}(PK, A) \). An encapsulation algorithm Encap takes as input the public key PK and an access structure \( A \). It returns a random string \( \kappa \) and its encapsulation \( \psi \). Note that \( A \) is contained in \( \psi \).

\( \text{KeyGen}(PK, MSK, S) \). 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 \( S \). Note that \( S \) is contained in SK\( S \).

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

Chosen-Ciphertext Attack on CP-ABKEM. According to previous works (for example, see \cite{13}), 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 \( A^* \) before \( A \) receives a public key PK, which is defined as the following experiment.

\begin{align*}
\text{Experiment}^{\text{IND-sel-CCA}}_{\mathcal{A}, \text{CP-ABKEM}}(\lambda, U) \in \{A, \bot\} \quad \text{Setup}(\lambda, U) \\
\in \{A, \bot\} \quad \text{KeyGen}(PK, MSK, \cdot, \text{Decap}(PK, SK, \cdot, \cdot)(PK)) \\
\in \{A, \bot\} \quad \text{Encap}(PK, A^*, \kappa) \quad \text{KeySp}(A) \quad b \leftarrow \{0, 1\} \\
\in \{A, \bot\} \quad b' \leftarrow \text{KeyGen}(PK, MSK, \cdot, \cdot, \text{Decap}(PK, SK, \cdot, \cdot)(\kappa, \psi')) \\
\in \{A, \bot\} \quad b' = b \quad \text{then Win else return Lose}
\end{align*}

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 KeyGen(PK, MSK, \cdot) for the secret key SK\( S_i \). Here we do not require any input attribute sets \( S_i \) and \( S_j \) to be distinct. Another is decapsulation queries. Indicating a pair \( (S_j, \psi_j) \) of an attribute set and an encapsulation, \( \mathcal{A} \) queries its decapsulation oracle Decap(PK, SK\( S_j \), \cdot) for the decapsulation result \( \hat{\kappa}_j \). Here an access structure \( A_j \), which is used to generate an encapsulation \( \psi_j \), is implicitly included in \( \psi_j \). In the case that \( S \notin A \), \( \hat{\kappa}_j = \bot \) 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 \( \lambda \).

The access structure \( A^* \) declared by \( \mathcal{A} \) is called a target access structure. Two restrictions are imposed on \( \mathcal{A} \) concerning \( A^* \). In key-extraction queries, each attribute set \( S_i \) must satisfy \( S_i \notin A^* \). In decapsulation queries, each pair \( (S_j, \psi_j) \) must satisfy \( S_j \notin A^* \lor \psi_j \neq \psi'^* \).

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

\[ \text{Adv}^{\text{IND-sel-CCA}}_{\mathcal{A}, \text{CP-ABKEM}}(\lambda, U) \stackrel{\text{def}}{=} \Pr[\text{Experiment}^{\text{IND-sel-CCA}}_{\mathcal{A}, \text{CP-ABKEM}}(\lambda, U) \text{ returns Win}] \]

CP-ABKEM is called selectively secure against chosen-ciphertext attacks if, for any PPT adversary \( \mathcal{A} \) and for any attribute universe \( U \), Adv\( ^{\text{IND-sel-CCA}}_{\mathcal{A}, \text{CP-ABKEM}}(\lambda, U) \) is negligible in \( \lambda \). Here we must distinguish the two cases; the case that \( U \) is small (i.e. \( |U| = u \) is bounded by a polynomial of \( \lambda \)) and the case that \( U \) is large (i.e. \( u \)

\(^2\)In Gorantla, Boyd and Nieto \cite{13}, they say encryption-policy attribute-based KEM (EP-AB-KEM) instead of saying ciphertext-policy attribute-based KEM here.
is not necessarily bounded by a polynomial of \( \lambda \). We assume the small case in this paper.

In the indistinguishability game against chosen-plaintext attack (IND-CPA game), the adversary \( A \) issues no decapsulation query (that is, \( q_3 = 0 \).

Ciphertext-Text-Attribute-Based Encryption Scheme (CP-ABE). In the case of the encryption version (i.e., CP-ABE), Encap(\( PK, A \)) and Decap(\( PK, SK_A, \psi \)) are replaced by PPT algorithms Encrypt(\( PK, A \)) and Decrypt(\( PK, SK_A, CT \)), 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 \( A \) 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 \( A \). In Guess phase, the adversary \( A \) returns \( b' \in \{0, 1\} \). If \( b' = b \), then \( A \) wins in the IND-CCA game. Otherwise, \( A \) 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.

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

\[ \text{Encap}(PK, S) \] An encapsulation algorithm Encap takes as input the public key PK and an attribute set \( S \). It returns a random string \( \kappa \) and its encapsulation \( \psi \). Note that \( S \) is contained in \( \psi \).

\[ \text{KeyGen}(PK, MSK, A) \] A key generation algorithm KeyGen takes as input the public key PK, the master secret key MSK and an access structure \( A \). It returns a secret key \( SK_A \) corresponding to \( S \). Note that \( A \) is contained in \( SK_A \).

\[ \text{Decap}(PK, SK_A, \psi) \] A decapsulation algorithm Decap takes as input the public key PK, an encapsulation (we also call it a ciphertext according to context) \( \psi \) and a secret key \( SK_A \). It first checks whether \( S \in A \). If the check result is False, it puts \( \kappa = \bot \). It returns a decapsulation result \( \kappa \).

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

\[ \text{Experiment}_{\text{ind-sel-cca}}(\lambda, U) \]

\[ S' \leftarrow A(\lambda, U), (PK, MSK) \leftarrow \text{Setup}(\lambda, U) \]

\[ \epsilon \leftarrow \text{KeyGen}(PK, MSK, A), \text{Decap}(PK, SK_A, \psi)(PK) \]

\[ (\kappa', \psi') \leftarrow \text{Encap}(PK, S'), \kappa \leftarrow \text{KeySp}(\lambda), b \leftarrow \{0, 1\} \]

If \( b = 1 \) then \( \kappa = \kappa' \) else \( \kappa = \kappa \)

\[ b' \leftarrow A(\text{KeyGen}(PK, MSK, A), \text{Decap}(PK, SK_A, \psi), (\kappa, \psi')) \]

If \( b' = b \) then Win else return Lose.

2.5 Target Collision Resistant Hash Functions

Target collision resistant (TCR) hash functions [18] are treated as a family. Let us denote a function family as \( \text{Hfam}(\lambda) = \{H_1(\cdot) | \text{HKey}(\lambda)\} \). Here \( \text{HKey}(\lambda) \) is a hash key space, \( \mu \in \text{HKey}(\lambda) \) is a hash key and \( H_\mu \) is a function from \( [0, 1]^n \) to \( [0, 1]^m \). We may assume that \( H_\mu \) from \( [0, 1]^n \) to \( Z_p^m \), where \( p \) is a prime of length \( \lambda \).

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

\[ \text{Experiment}_{\text{CF,Hfam}}(\lambda) \]

\[ m' \leftarrow CF(\lambda), m \leftarrow \text{HKey}(\lambda) \]

If \( m' \neq m \wedge H_\mu(m') = H_\mu(m) \)

then return Win else return Lose.

Then we define CF’s advantage over Hfam in the game of target collision resistance as follows.

\[ \text{Adv}_{\text{CF,Hfam}}(\lambda) \]

\[ \epsilon \leftarrow \text{Pr}[\text{Experiment}_{\text{CF,Hfam}}(\lambda) \text{ returns Win}]. \]

We say that \( Hfam \) is a TCR function family if, for any PPT algorithm \( CF \), \( \text{Adv}_{\text{CF,Hfam}}(\lambda) \) is negligible in \( \lambda \).

TCR hash function families can be constructed based on the existence of a one-way function [18].

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 \((g, g^{x_1}, g^{y}, g^{x_1+y}, g^{x_2}, g^{x_2+y})\) 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 (“Trapdoor Test” [12]) Let \( X_1, r, s \) be mutually independent random variables, where \( X_i \) 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 \( Y, Z_1, Z_2 \) are random variables taking values in \( \mathbb{G} \), each of which is defined independently of \( r \). Then the probability that the truth value of \( Z_1^r Z_2 - Y^s \) does not agree with the truth value of \((g, X_1, X_2, Y, Z_1, Z_2)\) being a twin DH tuple is at most 1/p. Moreover, if \((g, X_1, X_2, Y, Z_1, Z_2)\) is a twin DH tuple, then \( Z_1^r Z_2 = Y^s \) 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 \((g, X_1, X_2, Y, Z_1, Z_2)\) which passes the “Trapdoor Test” but which is not a twin DH tuple.
Let us define the advantage of Cheat over \( G \) as follows.

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

Now we are ready to complement Lemma 1.

Lemma 2 (Complement for “Trapdoor Test” [12])

For any algorithm Cheat with unbounded computational power, \( \text{Adv}_{\text{Cheat},G}^{\text{twinDH-test}}(\lambda) \) is at most \( 1/p \).

For a proof of Lemma 2 see Appendix A.

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 [4].

Overview of Our Modification

The Waters CP-ABE is proved to be secure in the IND-sel-CPA game [4]. 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 [12] and the algebraic trick of Boneh and Boyen [13] and Kiltz [14].

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 both the 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 \( e(g,g)^{\alpha t} \) 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 CP-ABKEM_rpa. Likewise, we distinguish parameters and algorithms of CP-ABKEM_rpa by the index rpa. 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 CP-ABKEM_{rpa} (the first scheme in [4]) corresponds to the case \( k = 1 \) below excluding the “check sum” \( (d_1, d_2) \).

Setup(\( \lambda, \mu \)). Setup takes as input the security parameter \( \lambda \) and the attribute universe \( \mathcal{U} = \{1, \ldots, u\} \).

It runs Grp(\( \lambda \)) to get \( (p, G, G_T, g, e) \), where \( G \) and \( G_T \) are cyclic groups of order \( p \), \( e : G \times G \rightarrow G_T \) is a bilinear map and \( g \) is a generator of \( G \). These become public parameters. Then Setup chooses \( u \) random group elements \( h_1, \ldots, h_u \in G \) that are associated with the \( u \) attributes. In addition, it chooses random exponents \( \alpha_k \in \mathbb{Z}_p, k = 1, \ldots, 4 \).

Decap(\( \mathbf{PK}, \psi, \mathbf{SK}_S \)). The decapsulation algorithm Decap takes as input the public key PK and an LSSS access structure \( \mathbf{A} = (M, \rho) \), where \( M \) is an \( n \times l \) matrix and \( \rho \) is the function which maps each row index \( i \) of \( M \) to an attribute in \( \mathcal{U} = \{1, \ldots, u\} \). Decap 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 Decap forms a vector \( \hat{\mathbf{v}} = (s, y_2, \ldots, y_n) \).

For \( i = 1 \) to \( l \), it calculates \( \lambda_i = \hat{\mathbf{v}} \cdot M_i \), where \( M_i \) denotes the \( i \)-th row vector of \( M \). In addition, Decap chooses random values \( r_1, \ldots, r_l \in \mathbb{Z}_p \). Then, a pair of a random one-time key and its encapsulation \((\kappa, \psi)\) is computed as follows.

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 generates the secret key \( \mathbf{SK}_S \) as follows.

For \( k = 1 \) to \( 4 \) : \( K_k = g^{\alpha_1} g^\gamma^{t_k}, L_k = g^\gamma \)

For \( x \in S : K_{k,x} = h_k^{\gamma x} \)

\( \mathbf{SK}_S = \{ (K_k, L_k, (K_{k,x}; x \in S)) | k = 1, \ldots, 4 \} \).
luation \( \hat{\kappa} \) is computed as follows.

Parse \( \psi \) into \( (\psi_{\text{cpa}} = (A, C^\prime, ((C_i, D_i); i = 1, \ldots, l), d_1, d_2); \)  
\[ \tau \leftarrow H_{\psi}(\psi_{\text{cpa}}); \]
For \( k = 1 \) to \( 4 \):
\[ \kappa_k^* = e(C^\prime, K_k^I) \prod_{i \in S} e(L_k^I, C_i) e(D_i, K_{\psi(i)})^{a_1} = e(g, g)^{a_3} \]
If \( \kappa_1^* \neq d_1 \lor \kappa_2^* \neq d_2 \), then put \( \hat{\kappa} = \bot \), else put \( \hat{\kappa} = \kappa_4^* \).

4.2 Security and Proof

**Theorem 1** If the Waters CP-ABKEM \( \psi \) is selectively secure against chosen-plaintext attacks and an employed hash function family H stands against collision resistance, then our CP-ABKEM is selectively secure against chosen-ciphertext attacks. More precisely, for any given PPT adversary \( A \) that attacks CP-ABKEM in the IND-sel-CCA game where decapsulation queries are at most \( q_4 \) times, and for any small attribute universe \( \mathcal{U} \), there exists a PPT adversary \( B \) that attacks CP-ABKEM in the IND-sel-CPA game and a PPT target collision finder \( CF \) on Hfdm that satisfy the following tight reduction.

\[
\text{Adv}^{\text{ind-sel-cpa}}_{A,\text{CP-ABKEM}}(\lambda, \mathcal{U}) 
\leq \text{Adv}^{\text{ind-sel-cpa}}_{B,\text{CP-ABKEM}}(\lambda, \mathcal{U}) + \text{Adv}^\text{cf}_{CF,Hfdm}(\lambda) + \frac{q_4}{p} 
\]

**Proof.** Given any adversary \( A \) that attacks our scheme CP-ABKEM in the IND-sel-CCA game, we construct an adversary \( B \) that attacks the Waters scheme CP-ABKEM in the IND-sel-CPA game as follows.

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

**Set up.** In return to outputting \( A^* \), \( B \) receives the public key \( PK_{\text{cpa}} \) for CP-ABKEM, which consists of the following components.

\[ PK_{\text{cpa}} = (g, g^\lambda, h_1, \ldots, h_u, e(g, g)^\lambda). \]

To set up a public key \( PK \) for CP-ABKEM, \( B \) (herein) needs a challenge instance: \( B \) queries its challenger and gets a challenge instance \((\hat{\kappa}, \psi_{\text{cpa}}^\prime)\). It consists of the following components.

\[ \hat{\kappa} = (g, g)^\lambda \]

OR a random one-time key \( \kappa \in \text{KeysP}^\prime(\lambda), \psi_{\text{cpa}}^\prime = (A^*, C^* = g^{\lambda \tau}; ((C_i, D'_i); i = 1, \ldots, l^*)) \).

Then \( B \) makes the rest of the PPTs of \( PK \) as follows.

Choose \( \eta \leftarrow H_{\psi}(\psi_{\text{cpa}}^\prime) \);  
Put \( e(g, g)^\eta = e(g, g)^\eta \);  
Choose \( \gamma_1, \gamma_2 \leftarrow Z_p \), put \( e(g, g)^\gamma_2 = e(g, g)^{\gamma_1}/e(g, g)^{a_1} \gamma_1 \);  
Choose \( \mu_1, \mu_2 \leftarrow Z_p \), put \( e(g, g)^{\mu_2} = e(g, g)^{\mu_1}/e(g, g)^{a_2} \gamma_2 \).

Note we have implicitly set relations in the exponent domain:

\[
\begin{align*}
\alpha_2 &= \gamma_2 - \alpha_1 \gamma_1, \\
\alpha_3 &= \mu_1 - \alpha_1 \tau, \\
\alpha_4 &= \mu_2 - \alpha_2 \tau = \mu_2 - (\gamma_2 - \alpha_1 \gamma_1) \tau. 
\end{align*}
\]

A public key PK for CP-ABKEM become:

\[
PK = (PK_{\text{cpa}}, e(g, g)^\gamma_1, e(g, g)^{\mu_1}, e(g, g)^{\gamma_2}, \eta). 
\]

Then \( B \) inputs PK into \( A \). Note that PK determines the corresponding MSK uniquely.

**Phase 1.** \( B \) answers for two types of \( A \)’s queries as follows.

(1) **Key-Extraction Queries.** In the case that \( A \) issues a key-extraction query for an attribute set \( S \subseteq \mathcal{U} \), \( B \) has to simulate \( A \)’s challenger. To do so, \( B \) issues key-extraction queries to \( B \)’s challenger for \( S \) repeatedly up to four times. As replies, \( B \) gets four secret keys of the Waters CP-ABKEM for a single attribute set \( S \):

\[
SK_{\text{cpa}, s,k} = (K_{\text{cpa}, k}, L_{\text{cpa}, k}, (K_{\text{cpa}, k,s}; x \in S)) \mid k = 1, \ldots, 4. 
\]

We remark that, according to the randomness in the key-generation algorithm of the Waters CP-ABKEM, all four secret keys \( SK_{\text{cpa}, s,k} \) are random and mutually independent. To reply a secret key \( SK_s \) of our CP-ABKEM to \( A \), \( B \) converts the four secret keys as follows.

\[
\begin{align*}
K_1 &= K_{\text{cpa}, 1}, \\
L_1 &= L_{\text{cpa}, 1}, \\
K_{1,x} &= K_{\text{cpa}, 1,x}, x \in S; \\
L_2 &= L_{\text{cpa}, 2}, \\
K_{2,x} &= K_{\text{cpa}, 2,x}, x \in S; \\
L_3 &= L_{\text{cpa}, 3}, \\
K_{3,x} &= K_{\text{cpa}, 3,x}, x \in S; \\
L_4 &= L_{\text{cpa}, 4}, \\
K_{4,x} &= K_{\text{cpa}, 4,x}, x \in S; \\
K &= K_{\text{cpa}, 4}. 
\end{align*}
\]

Then \( B \) replies \( SK_s = ((K_{k}, L_{k}, (K_{k,s}; x \in S)); k = 1, \ldots, 4) \) to \( A \).

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

If \( S \subseteq A \) then put \( \hat{\kappa} = \bot \), else

\[
\tau \leftarrow H_{\psi}(\psi_{\text{cpa}}); \\
\hat{\psi} = e(C^\prime, g)^{\gamma - \tau}; \\
\hat{Z}_1 = d_1/e(C^\prime, g)^{\mu_1}; \\
\hat{Z}_2 = d_2/e(C^\prime, g)^{\gamma_2}; \\
\end{align*}
\]

If \( \hat{Z}_1 \neq \hat{Z}_2 \neq \hat{Y} \) (\( \vdots \): call this checking TwinDH-Test) then put \( \hat{\kappa} = \hat{\psi} \) else

If \( \tau = \tau^* \) then abort (\( \vdots \): call this case Abort) else put \( \hat{\kappa} = \hat{\psi}_{\text{cpa}}^\prime, d_1, d_2 \).

**Challenge.** In the case that \( A \) queries its challenger for a challenge instance, \( B \) makes a challenge instance as follows.

\[
\begin{align*}
\hat{\psi} &= (\psi_{\text{cpa}}^\prime, d_1^*, d_2^*) \\
\psi^* &= (\psi_{\text{cpa}}^\prime, d_1^*, d_2^*). 
\end{align*}
\]
H. Anada et al. / Advances in Science, Technology and Engineering Systems Journal Vol. 3, No. 1, 261-273 (2018)

Then \( B \) feeds \((\hat{x}, \psi')\) to \( A \) as a challenge instance. 

**Phase 2.** The same as in Phase 1.

**Guess.** In the case that \( A \) returns \( A \)'s guess \( \hat{b} \), \( B \) returns \( \hat{b} \) itself as \( B \)'s guess.

In the above construction of \( B \), \( B \) can perfectly simulate the real view of \( A \) until the case \( \text{Abort} \) happens, except for a negligible case, and hence the algorithm \( A \) 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 \( SK_\lambda = ((K_\lambda, L_\lambda, (K_{\lambda,x} ; x \in S)); k = 1, \ldots, 4) \) for a key-extraction query of \( A \) is a perfect simulation.

**Proof.** We must consider the implicit relations \( \mathbb{I} \). For the index 2, we have implicitly set the randomness \( t_2 = l_{\text{cpa}, 2}(-\gamma_1) \) and we get:

\[
K_2 = g^{t_2} t_{\text{cpa}, 2}^{-\gamma_2} = g^{t_2} (g^{a_1} g^{a_2^{\alpha}})^{-\gamma_1} = g^{t_2 - \alpha_1} g^{a_2^{\alpha}} = g^{a_3} g^{a_4},
\]

\[
L_2 = l_{\text{cpa}, 2}^{-\gamma_1} t_{\text{cpa}, 2}^{-\gamma_2} = g^{t_3},
\]

\[
K_{2,x} = K_{\text{cpa}, 2,x}^{-\gamma_2} = (h^{t_{\text{cpa}, 2}})^{-\gamma_2} = h^{t_{\text{cpa}, 2}} x \in S.
\]

For the index 3 and 4, see Appendix \( \mathbb{E} \).

**Claim 2** \((e(g, g), e(g, g)^{a_1}, e(g, g)^{a_3}, \hat{Y}, \hat{Z}_1, \hat{Z}_2)\) is a twin Diffie-Hellman tuple if and only if \((e(g, g), e(g, g)^{a_1}, e(g, g)^{a_3}, e(g, g)^{a_2}, e(C', g), d_1, d_2)\) is a twin Diffie-Hellman tuple.

**Proof.** This claim can be proved by a short calculation. See Appendix \( \mathbb{C} \).

**Claim 3** \( (e(g, g), e(g, g)^{a_1}, e(g, g)^{a_3}, \hat{Y}, \hat{Z}_1, \hat{Z}_2) \) 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 \( \mathbb{I} \).

**Claim 4** Consider the following event which we name as \( \text{Overlook}_i \):

In the \( i \)-th TwinDH-Test, the following condition holds:

\[
\hat{Z}_1^{\gamma_1} \hat{Z}_2 = \hat{Y}^{\gamma_2} \text{ holds and}
\]

\[(e(g, g), e(g, g)^{a_1}, e(g, g)^{a_3}, \hat{Y}, \hat{Z}_1, \hat{Z}_2) \]

is NOT a twin DH tuple.

Then, for at most \( q_{\text{d}} \) times decapsulation queries of \( A \), the probability that at least one \( \text{Overlook}_i \) occurs is negligible in \( \lambda \). More precisely, the following inequality holds:

\[
\Pr \left[ \bigvee_{i=1}^{q_{\text{d}}} \text{Overlook}_i \right] \leq \frac{q_{\text{d}}}{p}.
\]

**Proof.** To apply Lemma \( \mathbb{2} \) we construct an algorithm \( \text{Cheat}_\lambda \) with unbounded computational power, which takes as input \((e(g, g), e(g, g)^{a_1}, e(g, g)^{a_3})\) and returns \((\hat{Y}, \hat{Z}_1, \hat{Z}_2)\) employing the adversary \( A \) as a subroutine.

First, note that the view of \( A \) in \( \text{Cheat}_\lambda \) is the same as the real view of \( A \) and hence the algorithm \( A \) works as designed.

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

\[
\sum_{i=1}^{q_{\text{d}}} \frac{1}{q_{\text{d}}} \Pr[\text{Overlook}_i] = \frac{1}{q_{\text{d}}} \sum_{i=1}^{q_{\text{d}}} \Pr[\text{Overlook}_i] = \frac{\text{Adv}_{\text{TwinDH-test},G}(\lambda)}{q_{\text{d}}}. \tag{3}
\]

Third, applying Lemma \( \mathbb{2} \) to \( \text{Cheat}_\lambda \), we get:

\[
\text{Adv}_{\text{TwinDH-test},G}(\lambda) \leq \frac{1}{p}. \tag{4}
\]

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

\[
\Pr \left[ \bigvee_{i=1}^{q_{\text{d}}} \text{Overlook}_i \right] \leq \sum_{i=1}^{q_{\text{d}}} \Pr[\text{Overlook}_i] = \frac{q_{\text{d}}}{p} \frac{\text{Adv}_{\text{TwinDH-test},G}(\lambda)}{q_{\text{d}}} = \frac{\text{Adv}_{\text{TwinDH-test},G}(\lambda)}{p}.
\]

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

\[
\Pr \left[ \bigvee_{i=1}^{q_{\text{d}}} \text{Overlook}_i \right] \leq \text{Adv}_{\text{TwinDH-test},G}(\lambda). \tag{5}
\]

**Proof.** This claim is proved by constructing a collision finder \( \text{CF} \) on \( \text{HHam} \). See Appendix \( \mathbb{D} \).

**Claim 6** The reply \( \hat{x} \) to \( A \) as an answer for a decapsulation query is correct.

**Claim 7** The challenge instance \( \psi' = (\psi_{\text{cpa}}, d_1', d_2') \) is correctly distributed.

**Proof.** These two claims are proved by a direct calculation. See Appendices \( \mathbb{E} \) and \( \mathbb{F} \), respectively.

**Evaluation of the Advantage of \( B \).** Now we are ready to evaluate the advantage of \( B \) in the IND-sel-CPA game. That \( A \) wins in the IND-sel-CCA game means that \((\hat{x}, \psi' = (\psi_{\text{cpa}}, d_1', d_2'))\) is correctly guessed. This is equivalent to that \((\hat{x}, \psi_{\text{cpa}}')\) is correctly guessed because \( \psi_{\text{cpa}}' \) determines the consistent binding factor \( \kappa' = e(g, g)^{a_{\alpha}} \) uniquely. This means that \( B \) wins in the IND-sel-CPA game.

Therefore, the probability that \( B \) wins is equal to the probability that \( A \) wins, \( \text{Overlook}_i \) never holds in TwinDH-Test for each \( i \) and \( \text{Abort} \) never occurs. So
Given \((e(g,g), e(g,g)^{a_1}, e(g,g)^{a_2})\) as input:

**Set up**

Initialize the inner state, put \(\text{TABLE} = \phi\);
Get a target access structure \(A^* \sim \mathcal{A}(\lambda, \mathcal{U})\);
Compute the base \(g \in G\) from \((e(g,g), e)\);
Choose \(a \in \mathbb{Z}_p\) and \(h_1, \ldots, h_u \in G\);
Put \(PK_{\text{cpa}} = (g, g^a, h_1, \ldots, h_u, e(g,g)^{a_1})\);
Get \((\kappa^*, \psi^*) \leftarrow \text{Encap}_{\text{cpa}}(PK_{\text{cpa}}, A^*)\);
Choose \(\eta \leftarrow HKey(\lambda)\) and compute \(\tau^* \leftarrow H_p(\psi^*_{\text{cpa}})\);
Compute discrete logarithms \(\alpha_1, \alpha_2 \in \mathbb{Z}_p\) of \((e(g,g)^{a_1}, e(g,g)^{a_2})\) to the base \(e(g,g)\);
Choose \(\mu_1, \mu_2 \sim \mathbb{Z}_p\), put \(\alpha_3 = \mu_1 - \alpha_1 \tau^*, \alpha_4 = \mu_2 - \alpha_2 \tau^*\);
Put \(PK = (PK_{\text{cpa}}, e(g,g)^{a_2}, e(g,g)^{a_3}, e(g,g)^{a_4}, \eta)\), \(\text{MSK} = (g^{a_1}, g^{a_2}, g^{a_3}, g^{a_4})\);
Give PK to \(A\);

**Phase 1**

In the case that \(A\) makes a key-extraction query for \(S \subseteq \mathcal{U}\);
Reply \(SK_S \to A\) in the same way as \(\text{KeyGen}\) does using MSK;
In the case that \(A\) makes a decapsulation query for \((A, \psi = (\psi_{\text{cpa}}, d_1, d_2), S)\);
Reply \(\kappa \to A\) in the same way as \(\text{Decap}\) does using MSK;
Compute \(\hat{Y} = e(C', g)^{\tau^*}, \hat{Z}_1 = d_1/e(C', g)^{\mu_1}, \hat{Z}_2 = d_2/e(C', g)^{\mu_2}\);
Update \(\text{TABLE} = \text{TABLE} \cup (\hat{Y}, \hat{Z}_1, \hat{Z}_2)\);

**Challenge**

In the case that \(A\) makes a challenge instance query:
Put \(d_1^* = e(C'', g)^{\mu_1}, d_2^* = e(C'', g)^{\mu_2}, \psi^* = (\psi_{\text{cpa}}, d_1^*, d_2^*)\);
Choose \(\kappa \leftarrow \text{KeySp}(\lambda), b \leftarrow \{0, 1\}\);
If \(b = 1\) then put \(\hat{\kappa} = \kappa^*\) else put \(\hat{\kappa} = \kappa\);
Reply \((\hat{\kappa}, \psi^*) \to A\);

**Phase 2**

The same as in Phase 1;

**Return**

In the case that \(A\) returns its guess \(b^*\);
Choose one triple \((\hat{Y}, \hat{Z}_1, \hat{Z}_2)\) from \(\text{TABLE}\) at random;
Return \((\hat{Y}, \hat{Z}_1, \hat{Z}_2)\).

Figure 1: An Algorithm \(\text{Cheat}_{\lambda, \mathcal{U}}\) with Unbounded Computational Power for a Proof of Claim \(\square\)

**4.3 Encryption Scheme from KEM**

It is straightforward to construct our encryption scheme \(\text{CP-ABE}\) from \(\text{CP-ABKEM}\). The IND-sel-CCA security of \(\text{CP-ABE}\) is proved based on IND-sel-CPA security of the Waters \(\text{KEM CP-ABKEM}_{\text{cpa}}\).

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

\textbf{Encrypt}(PK, A, m). The same as Encap of \(\text{CP-ABKEM}\) except that Encrypt multiplies \(m\) by the blinding factor \(\kappa\) in the group \(G_T\). Encrypt returns \(CT = (C = mk, \psi = (\psi_{\text{cpa}}, d_1, d_2))\).

\textbf{KeyGen}(MSK, PK, S). The same as KeyGen of \(\text{CP-ABKEM}\).

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

**4.4 Security and Proof**

\textbf{Theorem 2} If the Waters \(\text{CP-ABKEM}_{\text{cpa}}\) \([4]\) is selectively secure against chosen-plaintext attacks and an employed hash function family \(\text{Hfam}\) has target collision resistance, then our \(\text{CP-ABE}\) is selectively secure against chosen-ciphertext attacks. More precisely, for any given PPT ad-
versary $A$ that attacks CP-ABE in the IND-sel-CCA game where decryption queries are at most $q_A$ times, and for any small attribute universe $U$, there exist a PPT adversary $B$ that attacks CP-ABKEM$_{cpa}$ in the IND-sel-CPA game and a PPT target collision finder $CF$ on $H_{Ham}$ that satisfy the following inequality.

$$
\text{Adv}^{\text{ind-sel-cca}}_{A,\text{CP-ABE}}(\lambda, U) \\
\leq 2\left(\text{Adv}^{\text{ind-sel-cpa}}_{B,\text{CP-ABKEM}_{cpa}}(\lambda, U) + \text{Adv}^{\text{ter CF,Ham}}_{\text{CF,Ham}}(\lambda) + \frac{q_A}{p}\right).
$$

**Proof.** Given any adversary $A$ that attacks our scheme CP-ABE in the IND-sel-CCA game, we construct an adversary $B$ that attacks the Waters KEM CP-ABKEM$_{cpa}$ 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 $A^*$, $B$ receives the public key $PK_{cpa}$ for CP-ABKEM$_{cpa}$. To set up a public key $PK$ for CP-ABE, $B$ herein needs a challenge instance: $B$ queries its challenger and gets a challenge instance $(\kappa, \psi_{cpa})$. The rest of procedure is the same as that of CP-ABKEM, and $B$ inputs $PK$ into $A$.

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

**Challenge.** In the case that $A$ submits two plaintexts $(m_0, m_1)$ of equal length, $B$ makes a challenge ciphertext $CT^*$ as follows and feeds $CT^*$ to $A$.

- Choose $b^* \leftarrow \{0, 1\}$, put $C^* = m_{b^*}, \kappa$;
- $d_1^* = e(C^*, g)^\mu_1, d_2^* = e(C^*, g)^\mu_2$;
- $CT^* = (C^*, \psi^* = (\psi_{cpa}, d_1^*, d_2^*))$.

**Phase 2.** The same as in Phase 1.

**Guess.** In the case that $A$ returns $A$’s guess $\hat{b}$, $B$ returns $\hat{b}'$ as $B$’s guess.

**Evaluation of the Advantage of $B$.** A standard argument deduces a loss of tightness by a factor of $1/2$.

That is;

$$
\text{Adv}^{\text{ind-sel-cpa}}_{B,\text{CP-ABKEM}_{cpa}}(\lambda, U) \\
\geq \frac{1}{2}\text{Adv}^{\text{ind-sel-cca}}_{A,\text{CP-ABE}}(\lambda, U) - \frac{q_A}{p} - \text{Adv}^{\text{ter CF,Ham}}_{\text{CF,Ham}}(\lambda).
$$

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

In this section, we describe our direct chosen-ciphertext security modification by applying it to the Ostrovsky-Sahai-Waters KP-ABE [11].

**Overview of Our Modification** The Ostrovsky-Sahai-Waters KP-ABE is proved to be secure in the IND-sel-CPA game [11]. 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 [12] and the algebraic trick of Boneh and Boyen [13] and Kiltz [14].

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 $e(g, g)^{\eta_{\text{msk}}}$ 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 KP-ABKEM$_{cpa}$. Likewise, we distinguish parameters and algorithms of KP-ABKEM$_{cpa}$ by the index $cpa$. 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 KP-ABKEM$_{cpa}$ (the first scheme in [11]) corresponds to the case $k = 1$ below excluding the “check sum” $(d_1, d_2)$.

**Setup($\lambda, U$).** Setup takes as input the security parameter $\lambda$ and the attribute universe $U = \{1,\ldots, u\}$. It runs $\text{Grp}(\lambda)$ to get $(p, G, G_T, g, e)$, where $G$ and $G_T$ are cyclic groups of order $p$, $G \rightarrow G_T$ is a bilinear map and $g$ is a generator of $G$. These become public parameters. Then Setup chooses $u$ random group elements $h_1, \ldots, h_u \in G$ that are associated with the $u$ attributes. In addition, it chooses random exponents $a_k \in Z_p$, $k = 1, \ldots, 4, a \in Z_p$ and a hash key $\eta \in H_{Ham}(\lambda)$. The public key is published as $PK = (g, g^s, h_1, \ldots, h_u, e(g, g)^{a_1}, \ldots, e(g, g)^{a_4}, \eta)$. The authority sets $\text{MSK} = (a_1, \ldots, a_4)$ as the master secret key.

**Encap(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 $s \in Z_p$ that is the encryption randomness. Then, a pair of a random one-time key and its encapsulation $(\psi, \psi)$ is computed as follows.

- Put $C' = g^s$; For $x \in S: C_x = h_x^s$;
- $\psi_{cpa} = (S, C', (C_x; x \in S)), \tau \leftarrow H_{Ham}(\psi_{cpa})$;
- For $k = 1$ to $4: \kappa_k = e(g, g)^{a_k}; d_1 = \kappa_1^s \kappa_3, d_2 = \kappa_2^s \kappa_4$;
- $(\psi, \psi) = (\kappa_1, (\psi_{cpa}, d_1, d_2))$.

**KeyGen(MSK, PK, A).** The key generation algorithm KeyGen takes as input the master secret key MSK, the public key PK and an LSSS access structure $A = (M, \rho)$, where $M$ is an $l \times n$ matrix and $\rho$ is the function which maps each row index $i$ of $M$ to an attribute in $U = \{1, \ldots, u\}$. For $k = 1$ to $4$, KeyGen first chooses...
random values \( y_k, y_{k+1}, \ldots, y_{n-1} \in \mathbb{Z}_p \) and forms a vector \( \vec{y} = (y_k, y_{k+1}, \ldots, y_{n-1}) \). Then, for \( i = 1 \) to \( l \), it calculates \( \lambda_{ki} = \vec{y} \cdot M_i \), where \( M_i \) denotes the \( i \)-th row vector of \( M \), and it chooses random values \( r_{ki} \in \mathbb{Z}_p^l \). KeyGen generates the secret key \( SK_A \) as follows.

\[
\text{KeyGen}(MSK, PK, S). \quad \text{The same as KeyGen of KP-ABKEM.}
\]

\[
\text{Decrypt}(PK, CT, SK_K). \quad \text{The same as Decrypt of KP-ABKEM except that Decrypt divides out } C \text{ by the decapsulated blinding factor } \hat{\kappa}. \text{ Decrypt returns the result } \hat{m}.
\]

5.4 Security and Proof

Theorem 4 If the Ostrovsky-Sahai-Waters KP-ABKEM\_cpa [17] 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 \( A \) that attacks KP-ABE in the IND-sel-CCA game where decapsulation queries are at most \( q_{d} \) times, and for any small attribute universe \( U \), there exists a PPT adversary \( B \) that attacks KP-ABKEM\_cpa in the IND-sel-CPA game and a PPT target collision finder \( CF \) on \( Hfam \) that satisfy the following inequality.

\[
Adv_{\text{ind-sel-cca}}^{\text{ind-sel-cpa}}(\lambda, U) \\
\leq\frac{Adv_{\text{ind-sel-cpa}}^{\text{ind-sel-cca}}(\lambda, U) + Adv_{CF, Hfam}^{\text{tcr}}(\lambda) + \frac{q_{d}}{p}}{p}.
\]

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 5.4.

6 Efficiency Discussion

First of all, 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 \( e: G \times G \rightarrow G_T \). Especially, it is applicable even when an original CPA secure scheme is based on asymmetric pairing [19], \( e: G_1 \times G_2 \rightarrow G_T \). For example, the Type 3 version [19] of the Waters CP-ABE scheme [4] can be found in [20]. Detailed discussions and results on real implementations are found for the case of CPA-secure ABE schemes [21, 20]. We note here that the efficiency comparison below enables to guess the implementation results of CCA-secure ABE schemes via our modification.

We compare the efficiency of our CP-ABE with the original Waters CP-ABE\_cpa and our KP-ABE with the original Ostrovsky-Sahai-Waters KP-ABE\_cpa. We also compare the efficiency of our schemes with the CCA-secure CP-ABE and KP-ABE schemes obtained by the generic transformation in [10]. Here the generic transformation [10] is considered in the case of a small attribute universe, the delegation type [10] and the Laport one-time signature [22]. Table 1 shows these comparison. Note that a hash function is applied to the computation below in the case that an attribute universe

\[
\text{Set}(\lambda, U). \quad \text{The same as Setup of KP-ABKEM.}
\]

\[
\text{Encrypt}(PK, A, m). \quad \text{The same as Encaps of KP-ABKEM except that Encrypt multiplies } m \text{ by the blinding factor } \kappa \text{ in the group } G_T. \text{ Encrypt returns CT } = (C = m\kappa, \psi = (\psi_{\text{cpa}}, d_1, d_2)).
\]

www.astesj.com
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 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 by a factor of four. In the case of the Waters scheme (CP-ABKEM\textsubscript{CPA} and the Ostrovsky-Sahai-Waters KP-ABE\textsubscript{CPA}).

\begin{table}[h]
\centering
\begin{tabular}{|l|c|c|c|c|}
\hline
\textbf{Scheme} & \textbf{L(PK)} & \textbf{L(SK)} & \textbf{L(CT)} & \textbf{C(Enc)} & \textbf{C(Dec)} \\
\hline
Generic transform \cite{10}, CP-ABE & \(4\lambda^2(G)\) & \(4\lambda^2(G)\) & \(3\lambda^2(\text{bit})\) & \(2\lambda^2\text{exp}(G)\) & \(2\lambda^2\text{pair}(e)\) \\
Our individual modification (CP-ABE) & \(3(G_T)\) & \(4\times\lambda(G_T)\) & \(2\times\lambda(G_T)\) & \(4\text{exp}(G_T)\) & \(4\times\text{pair}(e)\) \\
Generic transform \cite{10}, KP-ABE & \(4\lambda^2(G)\) & \(0\) & \(3\lambda^2(\text{bit})\) & \(2\lambda^2\text{exp}(G)\) & \(2\lambda^2\text{pair}(e)\) \\
Our individual modification (KP-ABE) & \(3(G_T)\) & \(4\times\lambda(G_T)\) & \(2\times\lambda(G_T)\) & \(4\text{exp}(G_T)\) & \(4\times\text{pair}(e)\) \\
\hline
\end{tabular}
\caption{Efficiency comparison of IND-sel-CCA secure ABEs \cite{10} and ours with the original IND-sel-CPA secure ABEs \cite{4,11}.}
\end{table}

1) \(\lambda\) is the security parameter. (For instance, \(\lambda = 224\) or 256.)
2) \(L(\text{data})\) denotes the length of the data. \(C(\text{algorithm})\) denotes the computational amount of the algorithm.
3) \(+\) and \(\times\) mean the increment and the multiplier to the length or to the computational amount of the Waters scheme \(\text{CP-ABKEM}_{\text{CPA}}\) and the Ostrovsky-Sahai-Waters KP-ABE\textsubscript{CPA}.
4) \((G), (G_T)\) and \((\text{bit})\) mean that the lengths are evaluated in the number of elements in \(G\), elements in \(G_T\) and bits, respectively.
5) \(\text{exp}(G)\) and \(\text{pair}(e)\) mean the computational amount of one exponentiation in \(G\) and one pairing computation by the map \(e\), respectively.

7 Conclusion

We demonstrated direct chosen-ciphertext security modification for ABE in the standard model in the case of the Waters scheme (CP-ABKEM\textsubscript{CPA}, CP-ABE\textsubscript{CPA}) and the Ostrovsky-Sahai-Waters scheme (KP-ABKEM\textsubscript{CPA}, KP-ABE\textsubscript{CPA}). We utilized the twin Diffie-Hellman Trapdoor Test of Cash, Kiltz and Shoup and the algebraic trick of Boneh and Boyen \cite{13} and Kiltz \cite{14}. Our modification worked for the setting that the Diffie-Hellman tuple to be verified in decryption was in the target group of the bilinear map. We compared the efficiency of our CPA-secure ABE schemes with the original CPA-secure ABE schemes and with the CPA-secure ABE schemes obtained by the versatile generic transformation.

Acknowledgment This work was supported by JSPS KAKENHI Grant Number JP15K00029.

References

[1] 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.

[2] 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, October 30 - November 3, 2006, pages 89–98, 2006.

[3] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In 2007 IEEE Symposium on Security and Privacy (S& P 2007), 20-23 May 2007, Oakland, California, USA, pages 321–334, 2007.

[4] 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.

[5] 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.

[6] Hiroaki Anada and Seiko Arita. Short cca-secure ciphertext-policy attribute-based encryption. In 2017 IEEE International Conference on Smart Computing, SMARTCOMP 2017, Hong Kong, China, May 29-31, 2017, pages 1–6, 2017.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] Rafail Ostrovsky, Amit Sahai, and Brent 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 193–203, 2007.
Appendix

A Proof of Lemma 2

The only one point to be complemented to the original proof (in [12]) is that even for any algorithm $A$ with unbounded computational power, the statement holds. This is because, conditioning on the input fixed values ($g, X_1, X_2$), $A$ only reduces the two-dimensional freedom $(r, s)$ to $Z_p^2$ into the one-dimensional freedom $r \in Z_p$ even if $A$ correctly guesses the relation $s = r x_1 + x_2$. □

B Proof of Claim 1

For the index 3, we have implicitly set $t_3 = t_{cpa,3}(−τ)$ and we get:

$$K_3 = g^{\mu_1} K_{cpa,3} = g^{\mu_1} (g^{\alpha_1} g^{\alpha 2} \cdot r_3)^{t_3} = g^{\alpha_1 - \alpha_2} g^{\alpha 2},$$

$$L_3 = l_{cpa,3} = (g^{\alpha_2} x_3)^{t_3} = g^{\alpha 2},$$

$$K_{3,x} = r_{cpa,3,x} = k_{2(x^{\cdot -1})} = k_{2(x^1)}, x \in S.$$

For the index 4, we have implicitly set $t_4 = t_{cpa,4}(−τ)$ and we get:

$$K_4 = g^{\mu_4} K_{cpa,4} = g^{\mu_4} (g^{\alpha_4} g^{\alpha 2} \cdot r_4)^{t_4} = g^{\alpha_4 - \alpha 2} g^{\alpha 2},$$

$$L_4 = l_{cpa,4} = (g^{\alpha 4} x_4)^{t_4} = g^{\alpha 4},$$

$$K_{4,x} = r_{cpa,4,x} = k_{2(x^{\cdot -1})} = k_{2(x^1)}, x \in S.$$ □

C Proof of Claim 2

Suppose that we are given a twin DH tuple $(e(g, g), e(g, g)^{\gamma_1})$, $(e(g, g)^{\gamma 2}, Y, Z_1, Z_2)$. Then, $d_1(e(C^*, g)^{\gamma_1})$ and $d_2(e(C^*, g)^{\gamma 2})$ occur in Case 3. Hence, we get

$$(e(g, g)^{\gamma_1})^{t_3} = (e(g, g)^{\mu_1} (g^{\alpha_1} g^{\alpha 2} \cdot r_3)^{t_3}) = (e(g, g)^{\gamma_1}) x_3 = (e(g, g)^{\gamma_1}) k_{2(x^1)}, x \in S.$$
Given $\lambda$ as input:

**Set up**

Initialize inner state;
Choose a polynomial size attribute universe $U$ at random;
Get a target access structure $A^* \leftarrow A(\lambda, U)$;
Run $\text{Setup}_{\text{cpa}}(\lambda, U)$ to get $(p, G, G_T, g, e, \text{PK}_{\text{cpa}}, \text{MSK}_{\text{cpa}})$;
Get $(\kappa^*, \psi^*_{\text{cpa}}) \leftarrow \text{Encap}_{\text{cpa}}(\text{PK}_{\text{cpa}}, A^*)$;
Output $\psi^*_{\text{cpa}}$;
Receive, in return, $\eta \leftarrow H\text{Key}(\lambda)$ and compute $\tau^* \leftarrow H_2(\psi^*_{\text{cpa}})$;
Choose $\alpha_2, \alpha_3, \alpha_4 \leftarrow Z_p$;
Put $\text{PK} = (\text{PK}_{\text{cpa}}, e(g, g)^{\alpha_2}, e(g, g)^{\alpha_3}, e(g, g)^{\alpha_4}, \eta)$, $\text{MSK} = (g^{\alpha_1}, g^{\alpha_2}, g^{\alpha_3}, g^{\alpha_4})$;
Give $\text{PK}$ to $A$;

**Phase 1**

In the case that $A$ makes a key-extraction query for $S \subseteq U$;
Reply $\text{SK}_S$ to $A$ in the same way as $\text{KeyGen}$ does using $\text{MSK}$;
In the case that $A$ makes a decapsulation query for $(S, \psi = (\psi^*_{\text{cpa}}, d_1, d_2))$;
Reply $\kappa$ to $A$ in the same way as $\text{Decap}$ does using $\text{MSK}$;
If $\kappa \neq \perp$ and $\tau = \tau^*$ (call this case $\text{COLLISION}$)
then return $\psi^*_{\text{cpa}}$ and stop;

**Challenge**

In the case that $A$ makes a challenge instance query;
Using $\text{MSK}$, put $d'_1 = e(g^{\alpha_1}, C^\kappa)^{\tau^*} e(g^{\alpha_3}, C^\eta), d'_2 = e(g^{\alpha_2}, C^\kappa)^{\tau^*} e(g^{\alpha_4}, C^\eta),$
$\psi' = (\psi^*_{\text{cpa}}, d'_1, d'_2);$
Choose $\kappa \leftarrow \text{KeySp}(\lambda), b \leftarrow \{0, 1\};$
If $b = 1$ then put $\hat{\kappa} = \kappa^*$ else put $\hat{\kappa} = \kappa$;
Reply $(\hat{\kappa}, \psi')$ to $A$;

**Phase 2**

The same as in Phase 1;

**Return**

In the case that $A$ returns its guess $b^*$;
Stop.

Figure 2: A PPT Collision Finder $\text{CF}$ that attacks $H\text{fam}$ for the proof of Claim 5.

Taking a sum of both sides of equations 1 and 7, we get:

$$\Pr[\bigwedge_{i=1}^{2d} \overline{\text{Overlook}_i} \land \text{Abort}] \leq \Pr[\text{Collision}] = \text{Adv}_{\text{CF}, H\text{fam}}(\lambda).$$

\[\square\]

### E Proof of Claim 6

It is enough to prove that:

When $(r(g, g), e(g, g)^{\alpha_1}, e(g, g)^{\alpha_2}, \tilde{Y}, Z_1, Z_2)$ is a twin DH tuple,
$\hat{e} = Z_1^{1/(r - \tau^*)} = e(g, g)^{\alpha_1 s}$ holds.

This is deduced as follows:

$$\hat{e} = (d_1/e(C_i, g)^{\beta_1})^{1/(\tau - \tau^*)} = (e(g, g)^{\alpha_1})^{1/(\tau - \tau^*)} = e(g, g)^{\alpha_1 s} .$$

\[\square\]

### F Proof of Claim 7

A direct calculation with equalities 1 shows:

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

Hence $\psi^* = (\psi^*_{\text{cpa}}, d'_1, d'_2)$ is legitimate and correctly distributed. \[\square\]