Adaptive Chosen-Ciphertext Security of Distributed Broadcast Encryption

Kwangsu Lee*

Abstract

Distributed broadcast encryption (DBE) is a specific kind of broadcast encryption (BE) where users independently generate their own public and private keys, and a sender can efficiently create a ciphertext for a subset of users by using the public keys of the subset users. Previously proposed DBE schemes have been proven in the adaptive chosen-plaintext attack (CPA) security model and have the disadvantage of requiring linear number of pairing operations when verifying the public key of a user. In this paper, we propose an efficient DBE scheme in bilinear groups and prove adaptive chosen-ciphertext attack (CCA) security for the first time. To do this, we first propose a semi-static CCA secure DBE scheme and prove the security under the q-Type assumption. Then, by modifying the generic transformation of Gentry and Waters that converts a semi-static CPA secure DBE scheme into an adaptive CPA secure DBE scheme to be applied to CCA secure DBE schemes, we propose an adaptive CCA secure DBE scheme and prove its adaptive CCA security. Our proposed DBE scheme is efficient because it requires constant size ciphertexts, constant size private keys, and linear size public keys, and the public key verification requires only a constant number of pairing operations and efficient group membership checks.

Keywords: Distributed broadcast encryption, Chosen-ciphertext security, Adaptive security, Bilinear maps.

*Sejong University, Seoul, Korea. Email: kwangsu@sejong.ac.kr.

1 Introduction

Broadcast encryption (BE) is an encryption method that efficiently transmits an encrypted message to a subset of users, and the concept of BE was first introduced by Fiat and Naor [19]. In a general public-key BE scheme, the private keys of users are generated by a trusted central authority and transmitted securely to users. Afterwards, a sender creates a ciphertext for the subset of recipients by using the public parameters, and a recipient can decrypt the ciphertext with his or her private key if his or her index is included in the subset. By combining the ciphertexts of public-key encryption (PKE), it is possible to design a trivial BE scheme with a ciphertext size that increases linearly with the size of the recipient subset. Thus, an efficient BE scheme requires that the ciphertext has a sub-linear size. The most efficient public-key BE scheme is the BGW-BE scheme designed by Boneh et al. [7] in bilinear groups with constant size ciphertexts and constant size private keys, which provides static security under the q-Type assumption. Since then, various BE schemes have been proposed in bilinear groups, multilinear maps, indistinguishability obfuscation, and lattices [3, 9, 10, 17, 21, 22, 29, 30].

Traditional PKE schemes do not require a trusted center for private key generation because individual users independently generate public and private keys. Since BE schemes require a trusted center to generate user private keys, the trust model of BE must be stronger than that of PKE. This makes it difficult to widely apply BE schemes to large-scale real-world applications where only the weak trust model is applied. Recently, the concept of distributed broadcast encryption (DBE) has been proposed [10, 31], which does not require a trusted center for user private key generation. In DBE schemes, individual users generate their own public and private keys and store the public keys in a public repository. Then, a sender creates a ciphertext using the public keys for a subset of receivers, and a receiver can decrypt the ciphertext if he or she belongs to the subset of receivers in the ciphertext. Recently, Kolonelos et al. proposed efficient DBE schemes by decentralizing the private key generation of efficient BE schemes in bilinear groups and proved the adaptive CPA security of their DBE schemes [24].

The standard security model for PKE is the CCA security model, which allows an attacker to request ciphertext decryption queries [27]. The right security model for DBE schemes is also the adaptive CCA security model, which allows an attacker to submit a challenge target set in the challenge phase and request ciphertext decryption queries. Previously proposed DBE schemes have considered adaptive CPA security, but not adaptive CCA security. In this paper, we examine the problem of designing a DBE scheme that provides adaptive CCA security.

1.1 Our Contributions

To design an adaptive CCA secure DBE scheme, we first propose a semi-static CCA secure DBE scheme. A semi-static CCA security model is a restricted model in which an attacker initially commits an initial set $\tilde{S}$ and submits a challenge target set $S^* \subseteq \tilde{S}$ in the challenge phase, allowing the attacker to request key generation and decryption queries. To construct our DBE$_{SS}$ scheme, we start from the KMW-DBE1 scheme of Kolonelos et al. [24] that is derived from the BGW-BE scheme of Boneh et al. [7] that provides CPA security. In this case, we directly use a ciphertext element to provide ciphertext integrity instead of using a signature scheme to improve efficiency, and we modify the decryption process to derive a randomized private key like the private key of IBE. In addition, we apply a batch verification method to check public key elements of the user's public key. Since a simple batch verification method does not work, we change one group element in a ciphertext from $\mathbb{G}_1$ to $\mathbb{G}_2$ and set most of the public key elements to be group elements in $\mathbb{G}_1$. Our proposed DBE$_{SS}$ scheme has constant size ciphertexts, constant size private keys, linear size public keys, and linear size public parameters. The public key verification of our scheme is very efficient

Table 1: Comparison of DBE schemes in bilinear groups

Scheme	Type	PP	USK	UPK	CT	Model	Assumption
WQZD [31]	DBE	$O(L)$	$O(L)$	$O(L^2)$	$O(1)$	AD-CPA	q-Type
KMW [24]	DBE	$O(L)$	$O(1)$	$O(L)$	$O(1)$	AD-CPA	q-Type
KMW [24]	DBE	$O(L^2)$	$O(1)$	$O(L)$	$O(1)$	AD-CPA	k-LIN
Lee [25]	DBE	$O(L)$	$O(1)$	$O(L)$	$O(1)$	AD-CPA	SD, GSD
Ours	DBE	$O(L)$	$O(1)$	$O(L)$	$O(1)$	AD-CCA	q-Type

Let L be the number of all users. We count the number of group elements to measure the size. We use symbols AD-CPA for adaptive CPA security and AD-CCA for adaptive CCA security.

since it only requires two pairing operations and efficient group membership check operations. We prove the semi-static CCA security of our DBE_{SS} scheme under the q-Type assumption and the collision resistance of hash functions.

Next, we design an adaptive CCA secure DBE_{AD} scheme by combining the semi-static CCA secure DBE_{SS} scheme designed above, a strongly unforgeable one-time signature (OTS) scheme, and the GW transformation of Gentry and Waters [22], and prove adaptive CCA security through the hybrid games. In an adaptive CCA security model, an attacker can request key generation, key corruption, and decryption queries, and the attacker submits a challenge target set S^* in the challenge phase and obtains the corresponding challenge ciphertext header and challenge session key. The attacker wins this game if he or she can distinguish whether the challenge session key is correct or random. The GW transformation is a method that converts a semi-static CPA secure DBE scheme into an adaptive CPA secure DBE scheme by doubling the ciphertext size. We modify the existing GW transformation to combine a semi-static CCA secure DBE scheme and a strongly unforgeable OTS scheme for CCA security as well. Our proposed DBE_{AD} scheme is the first DBE scheme that provides adaptive CCA security, and it also naturally satisfies active-adaptive CCA security in which an attacker registers malicious public keys. The comparison of our DBE scheme with the previous DBE schemes is given in Table 1.

1.2 Related Work

Chosen-Ciphertext Security. The right definition of CCA security in PKE was given by Rackoff and Simon and they showed that a CCA secure PKE scheme can be constructed by combining a CPA secure PKE scheme with a non-interactive zero-knowledge proof (NIZK) system [27]. Cramer and Shoup proposed an efficient CCA secure PKE scheme by using a hash proof system (HPS), which is a special form of zero-knowledge proof [15, 16]. Canetti et al. presented a transformation method that converts a CPA secure IBE scheme to a CCA secure PKE scheme by using an one-time signature (OTS) scheme for ciphertext integrity [13]. After that, Boneh and Katz showed that a more efficient CCA secure PKE scheme can be built by using a message authentication code (MAC) scheme instead of the OTS scheme in the CHK transformation [6]. Boyen et al. modified the CHK transformation method and showed that an efficient CCA secure KEM without additional OTS and MAC schemes can be built by using the IBE scheme directly [11]. Dodis and Katz proposed a method to ensure CCA security when a ciphertext consists of multiple independent ciphertexts [18].

Broadcast Encryption. The concept of BE was first introduced by Fiat and Naor and they proposed a BE scheme by using combinatorial methods [19]. Naor et al. proposed BE schemes that are secure against collusion attacks without maintaining state information by using binary trees [26]. Boneh et al. proposed the first efficient BE scheme with constant size ciphertexts in bilinear groups and proved static CPA security under the q-Type assumption [7]. In addition, they proposed a CCA secure BE scheme by combining the IBE private key generation method with an OTS scheme. Abdalla et al. proposed a method to design a BE scheme with constant size ciphertexts by extending the private key delegation function of the hierarchical identity-based encryption (HIBE) scheme with constant size ciphertexts [1]. Most of the existing BE schemes are proven in the static model that specifies the challenge target set in advance, but the right security model is the adaptive model that specifies the challenge target set in the challenge phase. Gentry and Waters presented an efficient transformation to design an adaptive CPA secure BE scheme by doubling the ciphertext size of a semi-static CPA secure BE scheme [22]. Recently, methods for designing efficient BE schemes with optimal parameters and ciphertext sizes using attribute-based encryption (ABE) schemes have been proposed [2, 3, 30].

Distributed Broadcast Encryption. Wu et al. introduced the concept of ad-hoc broadcast encryption (AHBE) in which individual users independently generate private and public keys, and then proposed an AHBE scheme with constant size ciphertexts and linear size private keys in bilinear groups [31]. Boneh and Zhandry defined the concept of DBE and showed that a DBE scheme can be designed using indistinguishable obfuscation [10]. DBE schemes require user indexes when generating private keys of users, but flexible broadcast encryption (FBE) schemes do not require user indexes for key generation. Garg et al. proposed a general method to convert a DBE scheme into an FBE scheme by increasing the size of users' public keys [20]. Kolonelos et al. proposed efficient DBE schemes with constant size ciphertexts and constant size private keys in bilinear groups and proved adaptive CPA security [24]. Champion and Wu designed the first DBE scheme in lattices and proved the static CPA security [14].

2 Preliminaries

In this section, we review bilinear groups with complexity assumptions, symmetric-key encryption, and strongly unforgeable one-time signatures.

2.1 Bilinear Groups and Complexity Assumptions

A bilinear group generator $\mathcal{G}$ takes as input a security parameter λ and outputs a tuple $(p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e)$ where p is a random prime and $\mathbb{G}, \hat{\mathbb{G}}$, and $\mathbb{G}_T$ be three cyclic groups of prime order p. Let g and $\hat{g}$ be generators of $\mathbb{G}$ and $\hat{\mathbb{G}}$, respectively. The bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ has the following properties:

1. Bilinearity: $\forall u \in \mathbb{G}, \forall \hat{v} \in \hat{\mathbb{G}}$ and $\forall a, b \in \mathbb{Z}_p$, $e(u^a, \hat{v}^b) = e(u, \hat{v})^{ab}$.
2. Non-degeneracy: $\exists g \in \mathbb{G}, \hat{g} \in \hat{\mathbb{G}}$ such that $e(g, \hat{g})$ has order p in $\mathbb{G}_T$.

We say that $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ are asymmetric bilinear groups with no efficiently computable isomorphisms if the group operations in $\mathbb{G}, \hat{\mathbb{G}}$, and $\mathbb{G}_T$ as well as the bilinear map e are all efficiently computable, but there are no efficiently computable isomorphisms between $\mathbb{G}$ and $\hat{\mathbb{G}}$.

Assumption 1 (Bilinear Diffie-Hellman Exponent, BDHE [7])**.** Let $(p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e)$ be an asymmetric bilinear group generated by $\mathcal{G}(1^\lambda)$. Let $g, \hat{g}$ be random generators of $\mathbb{G}, \hat{\mathbb{G}}$ respectively. The ℓ-bilinear Diffie-

Hellman exponent (ℓ-BDHE) assumption is that if the challenge tuple

$$D = \left((p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e), g, g^a, \ldots, g^{a^\ell}, g^c, \hat{g}, \hat{g}^a, \ldots, \hat{g}^{a^\ell}, \hat{g}^{a^{\ell+2}}, \ldots, \hat{g}^{a^{2\ell}}\right) \text{ and } Z$$

are given, no PPT algorithm $\mathcal{A}$ can distinguish $Z = Z_0 = e(g,\hat{g})^{a^{\ell+1}c}$ from $Z = Z_1 = e(g,\hat{g})^d$ with more than a negligible advantage. The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_{\mathcal{A}}^{BDHE}(\lambda) = \left|\Pr[\mathcal{A}(D, Z_0) = 0] - \Pr[\mathcal{A}(D, Z_1) = 0]\right|$ where the probability is taken over random choices of $a, c, d \in \mathbb{Z}_p$.

2.2 Symmetric Key Encryption

Symmetric key encryption (SKE) is a cryptographic technique that uses the same symmetric key for encryption and decryption algorithms. The traditional security model for SKE is the indistinguishability under chosen-plaintext attack (IND-CPA) model, but we define the one-message indistinguishability (OMI) model that considers a single challenge ciphertext. If an SKE scheme is IND-CPA secure, then it is also OMI secure.

Definition 2.1 (Symmetric Key Encryption)**.** A symmetric key encryption (SKE) scheme consists of three algorithms **GenKey**, **Encrypt**, and **Decrypt**, which are defined as follows:

GenKey(1^λ): The key generation algorithm takes as input a security parameter λ. It outputs a symmetric key K.

Encrypt(K, M): The encryption algorithm takes as input a symmetric key K and a message M. It outputs a ciphertext C.

Decrypt(K, C): The decryption algorithm takes as input a symmetric key K and a ciphertext C. It outputs a message M or a special symbol $\perp$.

The correctness property of SKE is defined as follows: For all K generated by **GenKey**(1^λ) and any message M, it is required that **Decrypt**$(K,$ **Encrypt**$(K, M)) = M$.

Definition 2.2 (One-Message Indistinguishability)**.** The one-message indistinguishability (OMI) of SKE is defined in terms of the following experiment between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$ where 1^λ is given as input:

1. **Setup**: $\mathcal{C}$ obtains a symmetric key K by running **GenKey**(1^λ) and keeps K to itself.
2. **Challenge**: $\mathcal{A}$ submits challenge messages M_0^*, M_1^* where $|M_0^*| = |M_1^*|$. $\mathcal{C}$ flips a random coin $\mu \in \{0,1\}$ and obtains CT^* by running **Encrypt**(K, M_μ^*). It gives CT^* to $\mathcal{A}$.
3. **Guess**: $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$. $\mathcal{C}$ outputs 1 if $\mu = \mu'$ or 0 otherwise.

The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_{SKE,\mathcal{A}}^{OMI}(\lambda) = \left|\Pr[\mu = \mu'] - \frac{1}{2}\right|$ where the probability is taken over all the randomness of the experiment. An SKE scheme is OMI secure if for all probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible in the security parameter λ.

2.3 One-Time Signature

One-time signature (OTS) is a special kind of public-key signature (PKS) that allows an attacker to obtain at most one signature. The security model of OTS is strong unforgeability, which allows an attacker to obtain at most one signature from a signing oracle, and the message queried to the signing oracle is also allowed to be a forged message if the forged signature by the attacker is different from a signature received from the signing oracle.

Definition 2.3 (One-Time Signature)**.** A one-time signature (OTS) scheme consists of three algorithms **GenKey**, **Sign**, and **Verify**, which are defined as follows:

GenKey(1^λ): The key generation algorithm takes as input a security parameter λ. It outputs a signing key SK and a verification key VK.

Sign(SK, M): The signing algorithm takes as input a signing key SK and a message M. It outputs a signature σ.

Verify(VK, σ, M): The verification algorithm takes as input a verification key VK, signature σ, and a message M. It outputs 1 if the signature is valid and 0 otherwise.

The correctness property of OTS is defined as follows: For all (SK, VK) generated by **GenKey**(1^λ) and any message M, it is required that **Verify**$(VK, \textbf{Sign}(SK, M), M) = 1$.

Definition 2.4 (Strong Unforgeability)**.** The strong unforgeability (SUF) of OTS is defined in terms of the following experiment between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$ where 1^λ is given as input:

1. **Setup**: $\mathcal{C}$ first generates a key pair (SK, VK) by running **GenKey**(1^λ) and gives VK to $\mathcal{A}$.
2. **Signature Query**: $\mathcal{A}$ requests at most one signature query on a message M. $\mathcal{C}$ generates a signature σ by running **Sign**(SK, M) and gives σ to $\mathcal{A}$.
3. **Output**: Finally, $\mathcal{A}$ outputs a forged pair (σ^*, M^*). $\mathcal{C}$ outputs 1 if the forged pair satisfies the following conditions, or outputs 0 otherwise: 1) **Verify**$(VK, \sigma^*, M^*) = 1$, 2) $(\sigma^*, M^*) \neq (\sigma, M)$ where (σ, M) is the pair of the signature query.

The advantage of $\mathcal{A}$ is defined as $\textbf{Adv}^{SUF}_{OTS,\mathcal{A}}(\lambda) = \Pr[\mathcal{C} = 1]$ where the probability is taken over all the randomness of the experiment. An OTS scheme is SUF secure if for all probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible in the security parameter λ.

Efficient PKS schemes that provide strong unforgeability in bilinear groups include the BLS-PKS scheme of Boneh et al. [8] in the random oracle model and the BB-PKS scheme of Boneh and Boyen [4] without the random oracle model.

3 Distributed Broadcast Encryption

In this section, we define the syntax of DBE and the security model of DBE.

3.1 Definition

In a DBE scheme, a trusted center runs the setup algorithm to generate public parameters. Each user generates his or her own private key and public key by running the key generation algorithm and discloses the public key to a public directory. A sender runs the encryption algorithm to create a ciphertext header and a session key using a subset of recipients and their public keys. After that, a receiver can derive the same session key by running the decryption algorithm using the recipients' public keys and its own private key If the index of a receiver is included in the subset of the ciphertext header. A more detailed syntax of a DBE scheme is given as follows:

Definition 3.1 (Distributed Broadcast Encryption)**.** A distributed broadcast encryption (DBE) scheme consists of five algorithms **Setup**, **GenKey**, **IsValid**, **Encaps**, and **Decaps**, which are defined as follows:

Setup$(1^\lambda, 1^L)$: This algorithm takes as input a security parameter 1^λ, and the number users L. It outputs public parameters PP.

GenKey(i, PP): This algorithm takes as input a user index $i \in [L]$ and public parameters PP. It outputs a private key USK_i and a public key UPK_i.

IsValid(j, UPK_j, PP): This algorithm takes as input an index j, a public key UPK_j, and the public parameters PP. It outputs 1 or 0 depending on the validity of keys.

Encaps$(S, \{(j, UPK_j)\}_{j\in S}, PP, AU)$: This algorithm takes as input a set $S \subseteq [L]$, public keys $\{(j, UPK_j)\}_{j\in S}$, public parameters PP, and an optional auxiliary input AU. It outputs a ciphertext header CH and a session key CK.

Decaps$(S, CH, i, USK_i, \{(j, UPK_j)\}_{j\in S}, PP, AU)$: This algorithm takes as input a set S, a ciphertext header CH, an index i, a private key USK_i for the index i, public keys $\{(j, UPK_j)\}_{j\in S}$, public parameters PP, and an optional auxiliary input AU. It outputs a session key CK.

The correctness of DBE is defined as follows: For all PP generated by **Setup**$(1^\lambda, 1^L)$, all (USK_i, UPK_i) generated by **GenKey**(i, PP), all UPK_j such that **IsValid**(j, UPK_j, PP), all $S \subseteq [L]$, it is required that

- If $i \in S$, then $CK = CK'$ where $(CH, CK) =$ **Encaps**$(S, \{(j, UPK_j)\}_{j\in S}, PP, AU)$ and $CK' =$ **Decaps**$(S, CH, i, USK_i, \{(j, UPK_j)\}_{j\in S}, PP, AU)$.

Remark 1. We modify the syntax of DBE to accept an auxiliary input AU to the encryption and decryption algorithms. This AU field is optional and served as a label.

3.2 Security Model

The semi-static CPA security of BE was defined by Gentry and Waters [22]. We extend the existing semi-static CPA security model to the semi-static CCA security model by adding a decryption oracle. In the semi-static CCA security model, an attacker first submits an initial set $\tilde{S}$, and a challenger generates public parameters by running the setup algorithm. In the query phase, the attacker can request key generation and decryption queries with some constrains, and the challenger processes these queries by executing the key generation algorithm or the decryption algorithm. In the challenge phase, the attacker submits a set $S^* \subseteq \tilde{S}$, the challenger obtains a ciphertext header CH^* and a session key CK^* using the encryption algorithm, and it sets $CK_0^* = CK^*$ and CK_1^* with random. After that, the challenger flips a random coin μ and sends

the challenge CH^*, CK^*_μ to the attacker. Afterwards, the attacker can additionally request decryption queries except the challenge ciphertext header. Finally, the attacker wins this game if it correctly guesses the random coin of the challenger. The detailed definition of the security model is given as follows:

Definition 3.2 (Semi-Static CCA Security)**.** The semi-static CCA (SS-CCA) security of DBE is defined in terms of the following experiment between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$ where 1^λ and 1^L are given as input:

1. **Init**: $\mathcal{A}$ initially commits an initial set $\tilde{S} \subseteq [L]$.
2. **Setup**: $\mathcal{C}$ obtains public parameters PP by running $\mathbf{Setup}(1^\lambda, 1^L)$ and gives PP to $\mathcal{A}$.
3. **Query Phase 1**: $\mathcal{A}$ adaptively requests key generation and decryption queries. These queries are processed as follows:
   - Key Generation: For all $j \in \tilde{S}$, $\mathcal{C}$ generates (USK_j, UPK_j) by running $\mathbf{GenKey}(j, PP)$ and gives $\{(j, UPK_j)\}_{j \in \tilde{S}}$ to $\mathcal{A}$.
   - Decryption: $\mathcal{A}$ issues this query on (S, CH, i, AU) such that $S \subseteq \tilde{S}$ and $i \in S$. $\mathcal{C}$ responds with $\mathbf{Decaps}(S, CH, i, USK_i, \{(j, UPK_j)\}_{j \in S}, PP, AU)$.
4. **Challenge**: $\mathcal{A}$ submits a challenge set $S^* \subseteq \tilde{S}$. $\mathcal{C}$ obtains a ciphertext tuple (CH^*, CK^*) by running $\mathbf{Encaps}(S^*, \{(j, UPK_j)\}_{j \in S^*}, PP, -)$. It sets $CK^*_0 = CK^*$ and $CK^*_1 = RK$ by selecting a random RK. It flips a random coin $\mu \in \{0,1\}$ and gives (CH^*, CK^*_μ) to $\mathcal{A}$.
5. **Query Phase 2**: $\mathcal{A}$ continues to request decryption queries. These queries are processed as follows:
   - Decryption: $\mathcal{A}$ issues this query on (S, CH, i, AU) such that $S \subseteq \tilde{S}$, $i \in S$, and $CH \neq CH^*$. $\mathcal{C}$ responds with $\mathbf{Decaps}(S, CH, i, USK_i, \{(j, UPK_j)\}_{j \in S}, PP, AU)$.
6. **Guess**: Finally, $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$, and wins the game if $\mu = \mu'$.

The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}^{SS\text{-}CCA}_{DBE,\mathcal{A}}(\lambda) = \left| \Pr[\mu = \mu'] - \frac{1}{2} \right|$ where the probability is taken over all the randomness of the experiment. A DBE scheme is SS-CCA secure if for all probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible in the security parameter λ.

Adaptive CPA security of DBE is a relaxed security model in which an attacker can freely specify a challenge set S^* in the challenge phase. We define the adaptive CCA security model by adding a decryption oracle to the adaptive CPA security model. In the adaptive CCA security model, unlike the semi-static CCA model defined above, the attacker does not initially submit an initial set $\tilde{S}$. In the query phase, the challenger manages a key generation query set KQ, a key corruption query set CQ, and a decryption query set DQ for the queries requested by the attacker. In the challenge phase, the attacker submits a challenge set $S^* \subseteq KQ \setminus CQ$, and the challenge ciphertext header and the session key are transmitted to the attacker. After that, the attacker additionally requests decryption queries and wins the game if it can correctly guess the coin selected by the challenger. The detailed definition of the security model is given as follows:

Definition 3.3 (Adaptive CCA Security)**.** The adaptive CCA (AD-CCA) security of DBE is defined in terms of the following experiment between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$ where 1^λ and 1^L are given as input:

1. **Setup**: $\mathcal{C}$ obtains public parameters PP by running $\mathbf{Setup}(1^\lambda, 1^L)$ and gives PP to $\mathcal{A}$. It prepares KQ, CQ, and DQ as empty set.

2. **Query Phase 1**: $\mathcal{A}$ adaptively requests key generation, key corruption, and decryption queries. These queries are processed as follows:

   - Key Generation: $\mathcal{A}$ issues this query on an index $i \in [L]$ such that $i \notin KQ$. $\mathcal{C}$ creates (USK_i, UPK_i) by running **GenKey**(i, PP), adds i to KQ, and responds UPK_i to $\mathcal{A}$.
   - Key Corruption: $\mathcal{A}$ issues this query on an index $i \in [L]$ such that $i \in KQ \setminus (CQ \cup DQ)$. $\mathcal{C}$ adds i to CQ and responds USK_i to $\mathcal{A}$.
   - Decryption: $\mathcal{A}$ issues this query on (S, CH, i, AU) such that $S \subseteq KQ \setminus CQ$ and $i \in S$. $\mathcal{C}$ obtains CK by running **Decaps**$(S, CH, i, USK_i, \{(j, UPK_j)\}_{j \in S}, PP, AU)$, adds i to DQ, and responds CK to $\mathcal{A}$.

3. **Challenge**: $\mathcal{A}$ submits a challenge set $S^* \subseteq KQ \setminus CQ$. $\mathcal{C}$ obtains a ciphertext tuple (CH^*, CK^*) by running **Encaps**$(S^*, \{(j, UPK_j)\}_{j \in S^*}, PP, -)$. It sets $CK_0^* = CK^*$ and $CK_1^* = RK$ by selecting a random RK. It flips a random coin $\mu \in \{0,1\}$ and gives (CH^*, CK_μ^*) to $\mathcal{A}$.

4. **Query Phase 2**: $\mathcal{A}$ continues to request decryption queries. These queries are processed as follows:

   - Decryption: $\mathcal{A}$ issues this query on (S, CH, i, AU) such that $S \subseteq KQ \setminus CQ$, $i \in S$, and $CH \neq CH^*$. $\mathcal{C}$ obtains CK by running **Decaps**$(S, CH, i, USK_i, \{(j, UPK_j)\}_{j \in S}, PP, AU)$, adds i to DQ, and responds CK to $\mathcal{A}$.

5. **Guess**: Finally, $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$, and wins the game if $\mu = \mu'$.

The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_{DBE,\mathcal{A}}^{AD\text{-}CCA}(\lambda) = \left| \Pr[\mu = \mu'] - \frac{1}{2} \right|$ where the probability is taken over all the randomness of the experiment. A DBE scheme is AD-CCA secure if for all probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible in the security parameter λ.

The active-adaptive CCA security model is the extension of the adaptive CCA security model that allows an attacker to register malicious public keys [24]. In the active-adaptive security model, the index belonging to the set of maliciously registered public keys is not included in the challenge set, so it is generally known that an adaptive CPA secure DBE scheme also satisfies active-adaptive CPA security. For the same reason, the adaptive CCA secure DBE scheme is also active-adaptive CCA secure.

Definition 3.4 (Active-Adaptive CCA Security)**.** The active-adaptive CCA (AA-CCA) security of DBE is defined in terms of the following experiment between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$ where 1^λ and 1^L are given as input:

1. **Setup**: $\mathcal{C}$ obtains public parameters PP by running **Setup**$(1^\lambda, 1^L)$ and gives PP to $\mathcal{A}$. It prepares KQ, CQ, MQ, and DQ as empty set.

2. **Query Phase 1**: $\mathcal{A}$ adaptively requests key generation, key corruption, malicious corruption, and decryption queries. These queries are processed as follows:

   - Key Generation: $\mathcal{A}$ issues this query on an index $i \in [L]$ such that $i \notin KQ \wedge i \notin MQ$. $\mathcal{C}$ creates (USK_i, UPK_i) by running **GenKey**(i, PP), adds i to KQ, and responds UPK_i to $\mathcal{A}$.
   - Key Corruption: $\mathcal{A}$ issues this query on an index $i \in [L]$ such that $i \in KQ \wedge i \notin CQ$. $\mathcal{C}$ adds i to CQ and responds with USK_i to $\mathcal{A}$.
   - Malicious Corruption: $\mathcal{A}$ issues this query on an index $i \in [L]$ such that $i \notin KQ \wedge i \notin MQ$. $\mathcal{C}$ adds i to MQ and stores UPK_i.

- Decryption: $\mathcal{A}$ issues this query on (S, CH, i, AU) such that $S \subseteq KQ \setminus (CQ \cup MQ)$ and $i \in S$. $\mathcal{C}$ responds with $\mathbf{Decaps}(S, CH, i, USK_i, \{(j, UPK_j)\}_{j \in S}, PP, AU)$.

3. **Challenge**: $\mathcal{A}$ submits a challenge set $S^* \subseteq KQ \setminus (CQ \cup MQ)$. $\mathcal{C}$ obtains a ciphertext tuple (CH^*, CK^*) by running $\mathbf{Encaps}(S^*, \{(j, UPK_j)\}_{j \in S^*}, PP, -)$. It sets $CK_0^* = CK^*$ and $CK_1^* = RK$ by selecting a random RK. It flips a random coin $\mu \in \{0,1\}$ and gives (CH^*, CK_μ^*) to $\mathcal{A}$.

4. **Query Phase 2**: $\mathcal{A}$ continues to request decryption queries. These queries are processed as follows:

   - Decryption: $\mathcal{A}$ issues this query on (S, CH, i, AU) such that $S \subseteq KQ \setminus (CQ \cup MQ)$, $i \in S$, and $CH \neq CH^*$. $\mathcal{C}$ responds with $\mathbf{Decaps}(S, CH, i, USK_i, \{(j, UPK_j)\}_{j \in S}, PP, AU)$.

5. **Guess**: Finally, $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$, and wins the game if $\mu = \mu'$.

The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}_{DBE,\mathcal{A}}^{AA\text{-}CCA}(\lambda) = \left| \Pr[\mu = \mu'] - \frac{1}{2} \right|$ where the probability is taken over all the randomness of the experiment. A DBE scheme is AA-CCA secure if for all probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible in the security parameter λ.

Lemma 3.1 ([24]). *Let Π_{AD} be an adaptively secure DBE scheme. Then Π_{AD} is also active-adaptively secure.*

4 DBE Construction

In this section, we propose CCA secure efficient DBE schemes in bilinear groups.

4.1 SS-CCA Construction

We first construct a semi-static CCA-secure DBE scheme. To design this DBE scheme, we start from the KMW-DBE1 scheme of Kolonelos et al. [24], which is a DBE scheme modified from the BGW-BE scheme of Boneh et al. [7]. The KMW-DBE1 scheme provides CPA security with constant-size ciphertexts, constant-size private keys, linear-size public keys, and linear-size public parameters. One way to design a CCA-secure DBE scheme is to use the OTS scheme to provide ciphertext integrity and bind the OTS public key to the ciphertext. However, this method has a disadvantage in that the ciphertext size increases because the OTS public key and OTS signature are included in the ciphertext. To reduce the ciphertext size of the DBE scheme, we apply the BMW technique of Boyen et al. [11], which provides ciphertext integrity by directly using the element included in the ciphertext header. However, the BMW technique is not simply applied to the DBE scheme that supports the distributed private key generation. To solve this problem, we modify the BMW technique to provide CCA even for DBE schemes where individual users have public keys by adding additional public parameters.

In addition, the existing KMW-DBE1 scheme has a disadvantage that it requires L pairing operations to check whether the user public key is correct where L is the maximum number of users. One way to improve the user public key verification is to use a batch verification method [12]. However, it is difficult to apply efficient batch verification because the group membership check in $\mathbb{G}_2$ is inefficient when the public key contains many $\mathbb{G}_2$ group elements. To solve this problem, we change the ciphertext structure so that the ciphertext consists of $\mathbb{G}_2$ and $\mathbb{G}_1$ group elements instead of $\mathbb{G}_1$ group elements. Because of this change, we can set most of the user public key elements to be $\mathbb{G}_1$ group elements. In this case, public key verification using batch verification is very efficient because two pairing operations are needed for batch verification and

the group membership check in $\mathbb{G}_1$ is very efficient. The detailed description of our DBE scheme for the semi-static CCA security is as follows:

DBE_{SS}.Setup$(1^\lambda, 1^L)$**:** Let λ be a security parameter and L be the maximum number of users. It first generates asymmetric bilinear groups $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ of prime order p with random generators $g, \hat{g}$ of $\mathbb{G}, \hat{\mathbb{G}}$ respectively. It selects a random exponent $\alpha \in \mathbb{Z}_p$ and sets $\{A_k = g^{\alpha^k}\}_{k=1}^{2L+2}$, $\{\hat{A}_k = \hat{g}^{\alpha^k}\}_{k=1}^{L+1}$. It also selects a random exponent $\beta \in \mathbb{Z}_p$ and sets $B = g^\beta, \{B_k = A_k^\beta\}_{2 \le k \le L+1}$. Next, it chooses two hash functions $\mathbf{H}_1$ and $\mathbf{H}_2$ such that $\mathbf{H}_1 : \hat{\mathbb{G}} \to \mathbb{Z}_p$ and $\mathbf{H}_2 : \mathbb{G}_T \to \{0,1\}^\lambda$. It outputs public parameters

$$PP = \Big((p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e),\ g,\ \hat{g},\ \{A_k\}_{1 \le k \ne L+2 \le 2L+2},\ \{\hat{A}_k\}_{1 \le k \le L+1},\ B,\ \{B_k\}_{2 \le k \le L+1},$$
$$\Omega = e(A_{L+2}, \hat{g}),\ \mathbf{H}_1,\ \mathbf{H}_2 \Big).$$

DBE_{SS}.GenKey(i, PP)**:** Let $i \in [L]$. It selects a random exponent $\gamma_i \in \mathbb{Z}_p$ and builds a private key USK_i and a public key UPK_i as

$$USK_i = \Big(K_i = A_{L+2-i}^{\gamma_i} \Big),\ UPK_i = \Big(V_i = g^{\gamma_i},\ \hat{V}_i = \hat{g}^{\gamma_i},\ \{V_{i,k} = A_k^{\gamma_i}\}_{2 \le k \ne L+2-i \le L+1} \Big).$$

DBE_{SS}.IsValid(j, UPK_j, PP)**:** Let $UPK_j = (V_j, \hat{V}_j, \{V_{j,k}\})$. It first checks that $UPK_j = (V_j, \hat{V}_j, \{V_{j,k}\}) \in \mathbb{G} \times \hat{\mathbb{G}} \times \mathbb{G}^{2L}$. It chooses small random exponents $\delta_0, \{\delta_k\}_{k=2}^{2L+1}$ where δ_0, δ_k are elements of ℓ_b bit from $\mathbb{Z}_p$ and checks that

$$e\Big(V_j^{\delta_0} \prod_{2 \le k \ne L+2-j \le 2L+2} V_{j,k}^{\delta_k}, \hat{g} \Big) \stackrel{?}{=} e\Big(g^{\delta_0} \prod_{2 \le k \ne L+2-j \le 2L+2} A_k^{\delta_k}, \hat{V}_j \Big).$$

If the equation holds, then it outputs 1. Otherwise, it outputs 0.

DBE_{SS}.Encaps$(S, \{(j, UPK_j)\}_{j \in S}, PP, AU)$**:** Let $UPK_j = (V_j, \hat{V}_j, \{V_{j,k}\})$. It selects a random exponent $t \in \mathbb{Z}_p$ and computes a tag $\omega = \mathbf{H}_1(\hat{g}^t \| AU)$. It outputs a ciphertext header

$$CH = \Big(\hat{C}_1 = \hat{g}^t,\ C_2 = \big(A_{L+1}^\omega B \prod_{j \in S} A_j V_j \big)^t \Big)$$

and a session key $CK = \mathbf{H}_2(\Omega^t)$.

DBE_{SS}.Decaps$(S, CH, i, USK_i, \{(j, UPK_j)\}_{j \in S}, PP, AU)$**:** Let $CH = (\hat{C}_1, C_2)$, $USK_i = K_i$, and $UPK_j = (V_j, \hat{V}_j, \{V_{j,k}\})$. If $i \notin S$, it outputs $\perp$. It computes a tag $\omega = \mathbf{H}_1(\hat{C}_1 \| AU)$ and checks the validity of the ciphertext

$$e\big(C_2, \hat{g}\big) \stackrel{?}{=} e\Big(A_{L+1}^\omega B \prod_{j \in S} A_j V_j, \hat{C}_1 \Big).$$

If the equation fails, then it outputs $\perp$. It selects a random exponent $r \in \mathbb{Z}_p$ and builds decryption components

$$D_1 = K_i \cdot \Big(A_{L+1}^\omega B \prod_{j \in S} A_j V_j \Big)^r,\ \hat{D}_2 = \hat{A}_{L+2-i} \cdot \hat{g}^r,$$
$$D_3 = A_{2L+3-i}^\omega B_{L+2-i},\ D_4 = \prod_{j \in S \setminus \{i\}} A_{L+2-i+j} V_{j,L+2-i}.$$

It outputs a session key $CK = \mathbf{H}_2(e(C_2, \hat{D}_2) \cdot e(D_1 \cdot D_3 \cdot D_4, \hat{C}_1)^{-1})$.

4.2 AD-CCA Construction

Now, we construct an adaptive CCA-secure DBE scheme from the semi-static CCA-secure DBE scheme. The GW transformation of Gentry and Waters [22] is used to convert a semi-static CPA-secure BE or DBE scheme into the adaptive CPA-secure BE or DBE scheme. However, the GW transformation is applied to the CPA-secure BE or DBE scheme, and it needs additional modification to be applied to the CCA-secure BE or DBE scheme. We derive the adaptive CCA-secure DBE scheme by combining the semi-static CCA-secure DBE scheme, the OTS scheme, and the GW transformation. Here, we use the OTS signature for ciphertext integrity, and we modify the encryption and decryption algorithms of the underlying DBE scheme to additionally receive a label as input to bind the OTS public key with the DBE ciphertext. This modification of algorithms to receive an additional label as input was widely used in the design of existing CCA-secure PKE schemes [23, 28]. The detailed description of our DBE scheme for adaptive CCA security is given as follows:

$\mathbf{DBE}_{AD}$.Setup$(1^\lambda, 1^L)$**:** Let λ be a security parameter and L be the number of users. It obtains PP_{SS} by running **$\mathbf{DBE}_{SS}$.Setup**$(1^\lambda, 1^{2L})$. It outputs public parameters $PP = PP_{SS}$.

$\mathbf{DBE}_{AD}$.GenKey(i, PP)**:** Let $i \in [L]$. It generates two key pairs $(USK_{SS,2i}, UPK_{SS,2i})$ and $(USK_{SS,2i-1}, UPK_{SS,2i-1})$ by running **$\mathbf{DBE}_{SS}$.GenKey**$(2i, PP)$ and **$\mathbf{DBE}_{SS}$.GenKey**$(2i-1, PP)$ respectively. It selects a random bit $u_i \in \{0,1\}$ and erases $USK_{SS,2i-(1-u_i)}$ completely. It outputs a private key $USK_i = (USK_{SS,2i-u_i}, u_i)$ and a public key $UPK_i = (UPK_{SS,2i}, UPK_{SS,2i-1})$.

$\mathbf{DBE}_{AD}$.IsValid(j, UPK_j, PP)**:** Let $UPK_j = (UPK_{SS,2j}, UPK_{SS,2j-1})$. It checks that **$\mathbf{DBE}_{SS}$.IsValid**$(2j, UPK_{SS,2j}, PP_{SS}) = 1$ and **$\mathbf{DBE}_{SS}$.IsValid**$(2j-1, UPK_{SS,2j-1}, PP_{SS}) = 1$. If it passes all checks, then it outputs 1. Otherwise, it outputs 0.

$\mathbf{DBE}_{AD}$.Encaps$(S, \{(j, UPK_j)\}_{j\in S}, PP, AU)$**:** Let $S \subseteq [L]$ and $UPK_j = (UPK_{SS,2j}, UPK_{SS,2j-1})$.

1. It generates (SK, VK) by running **OTS.GenKey**(1^λ).
2. It selects random bits $z = \{z_j\}_{j\in S}$ where $z_j \in \{0,1\}$. Next, it defines two sets $S_0 = \{2j - z_j\}_{j\in S}$ and $S_1 = \{2j - (1-z_j)\}_{j\in S}$.
3. It obtains ciphertext pairs $(CH_{SS,0}, CK_{SS,0})$ and $(CH_{SS,1}, CK_{SS,1})$ by running **$\mathbf{DBE}_{SS}$.Encaps**$(S_0, \{(k, UPK_{SS,k}\}_{k\in S_0}, PP_{SS}, VK)$ and **$\mathbf{DBE}_{SS}$.Encaps**$(S_1, \{(k, UPK_{SS,k}\}_{k\in S_1}, PP_{SS}, VK)$ respectively.
4. It selects a random message $CK \in \{0,1\}^\lambda$. It obtains ciphertexts CT_0 and CT_1 by running **SKE.Encrypt**$(CK_{SS,0}, CK)$ and **SKE.Encrypt**$(CK_{SS,1}, CK)$ respectively.
5. It sets $CM = (CH_{SS,0}, CH_{SS,1}, CT_0, CT_1, z)$ and calculates σ by running **OTS.Sign**(SK, CM).
6. It outputs a ciphertext header $CH = (CM, \sigma, VK)$ and a session key CK.

$\mathbf{DBE}_{AD}$.Decaps$(S, CH, i, USK_i, \{(j, UPK_j)\}_{j\in S}, PP, AU)$**:** Let $CH = (CM, \sigma, VK)$ and $USK_i = (USK_{SS,2i-u_i}, u_i)$ where $CM = (CH_{SS,0}, CH_{SS,1}, CT_0, CT_1, z)$. If $i \notin S$, it outputs $\perp$.

1. It checks that $1 \stackrel{?}{=}$ **OTS.Verify**(VK, σ, CM). If this check fails, it outputs $\perp$.
2. It derives two sets $S_0 = \{2j - z_j\}_{j\in S}$ and $S_1 = \{2j - (1-z_j)\}_{j\in S}$. If $z_i = u_i$, then it sets $S' = S_0, CH'_{SS} = CH_{SS,0}, CT' = CT_0$. Otherwise, it sets $S' = S_1, CH'_{SS} = CH_{SS,1}, CT' = CT_1$.
3. It obtains CK'_{SS} by running **$\mathbf{DBE}_{SS}$.Decaps**$(S', CH'_{SS}, 2i-u_i, USK_{SS,2i-u_i}, \{(k, UPK_{SS,k}\}_{k\in S'}, PP_{SS}, VK)$.
4. It obtains CK by running **SKE.Decrypt**(CK'_{SS}, CT') and outputs a session key CK.

4.3 Correctness

Theorem 4.1. *The above DBE$_{SS}$ scheme is correct.*

Proof. To show the correctness of this scheme, we first show that the same session key can be derived. If $i \in S$, then we can derive the following equation

$$
\begin{aligned}
& e(C_2, \hat{A}_{L+2-i}) \\
&= e(\big(A_{L+1}^{\omega} B \prod_{j \in S} A_j V_j\big)^t, \hat{A}_{L+2-i}) \\
&= e(\big(A_{L+1}^{\omega} B \cdot A_i V_i \cdot \prod_{j \in S \setminus \{i\}} A_j V_j\big)^t, \hat{A}_{L+2-i}) \\
&= e((A_{L+1}^{\omega} B)^t, \hat{A}_{L+2-i}) \cdot e(A_i^t, \hat{A}_{L+2-i}) \cdot e(V_i^t, \hat{A}_{L+2-i}) \cdot e(\big(\prod_{j \in S \setminus \{i\}} A_j V_j\big)^t, \hat{A}_{L+2-i}) \\
&= e(A_{2L+3-i}^{\omega} B_{L+2-i}, \hat{g}^t) \cdot e(A_{L+2}, \hat{g}^t) \cdot e(\hat{A}_{L+2-i}^{\gamma_i}, \hat{g}^t) \cdot e(\big(\prod_{j \in S \setminus \{i\}} A_{L+2-i+j} A_{L+2-i}^{\gamma_j}, \hat{g}^t\big) \\
&= e(D_3, \hat{g}^t) \cdot \Omega^t \cdot e(K_i, \hat{g}^t,) \cdot e(D_4, \hat{g}^t) \\
&= \Omega^t \cdot e(K_i \cdot D_3 \cdot D_4, \hat{C}_1).
\end{aligned}
$$

By using the above equation, we can obtain that the same element that is used to derive a session key as follows

$$
\begin{aligned}
& e(C_2, \hat{D}_2) \cdot e(D_1 \cdot D_3 \cdot D_4, \hat{C}_1)^{-1} \\
&= e(C_2, A_{L+2-i}) \cdot e(C_2, \hat{g}^r) \cdot e(D_1 \cdot D_3 \cdot D_4, \hat{C}_1)^{-1} \\
&= \Omega^t \cdot e(K_i \cdot D_3 \cdot D_4, \hat{C}_1) \cdot e(\big(A_{L+1}^{\omega} B \prod_{j \in S} A_j V_j\big)^t, \hat{g}^r) \cdot e(D_1^{-1}, \hat{C}_1) \cdot e((D_3 \cdot D_4)^{-1}, \hat{C}_1) \\
&= \Omega^t \cdot e(K_i \cdot \big(A_{L+1}^{\omega} B \prod_{j \in S} A_j V_j\big)^r, \hat{C}_1) \cdot e(D_1^{-1}, \hat{C}_1) = \Omega^t.
\end{aligned}
$$

Next, we show that the verification of a public key is correct. Since our public key verification uses the small exponent test, the validity of this batch verification can be checked by the previous work of Camenisch et al. [12], we omit the detailed analysis of this batch verification. □

Theorem 4.2. *The above DBE$_{AD}$ scheme is correct.*

The correctness of the DBE$_{AD}$ scheme can be easily derived from the correctness of the DBE$_{SS}$, SKE, and OTS schemes. We omit the proof of this theorem.

5 Security Analysis

In this section, we prove the semi-static CCA security and adaptive CCA security of the proposed DBE schemes.

5.1 Semi-Static Security Analysis

The basic idea of the semi-static CCA proof is to use the partitioning technique that divides the ciphertext header space into two regions: a challenge ciphertext header region and a decryption oracle ciphertext header region. The challenge ciphertext header CH^* is associated with a tag ω^*, and the decryption oracle ciphertext CH is associated with a tag ω. If $\omega^* \neq \omega$, a simulator first derives a decryption key to be used for decrypting the ciphertext header CH and perform the decryption of CH to handle the decryption query. In other words, the simulator can decrypt the ciphertext header requested by the attacker by using a method like the IBE private key derivation for an identity ω. If $\omega^* = \omega$ holds, the simulator can find the collisions of the collision-resistant hash function, which ensures that such an event will not occur. The detailed description of the semi-static CCA proof is given as follows:

Theorem 5.1. *The above DBE$_{SS}$ scheme is SS-CCA secure if the $(L+1)$-BDHE assumption holds and $\boldsymbol{H}_1$ is a collision-resistant hash function.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that breaks the DBE$_{SS}$ scheme with a non-negligible advantage. A simulator $\mathcal{B}_1$ that solves the $L+1$-BDHE assumption using $\mathcal{A}$ is given: a challenge tuple $D = (g, g^a, \ldots, g^{a^{L+1}}, g^{a^{L+3}}, \ldots, g^{2L+2}, g^b, \hat{g}, \hat{g}^a, \ldots, \hat{g}^{a^{L+1}}, \hat{g}^b)$ and Z where $Z = Z_0 = e(g,\hat{g})^{a^{L+2}b}$ or $Z = Z_1 = e(g,\hat{g})^c$. Then $\mathcal{B}_1$ that interacts with $\mathcal{A}$ is described as follows:

Init: $\mathcal{A}$ commits an initial set $\tilde{S}$.

Setup: $\mathcal{B}_1$ sets $\hat{C}_1^* = \hat{g}^b$ and computes $\omega^* = \mathbf{H}_1(\hat{C}_1^*)$. It selects a random exponent β' and implicitly sets $\beta = -\omega^* a^{L+1} + \beta'$. It creates public parameters

$$g,\ \hat{g},\ \left\{A_k = g^{a^k}\right\}_{1\leq k\neq L+2\leq 2L+2},\ \left\{\hat{A}_k = \hat{g}^{a^k}\right\}_{1\leq k\leq L+1},$$
$$B = A_{L+1}^{-\omega^*} g^{\beta'},\ \left\{B_k = A_{L+1+k}^{-\omega^*} A_k^{\beta'}\right\}_{2\leq k\leq L+1},\ \Omega = e(A_{L+1}, \hat{A}_1).$$

Query Phase 1: For each index $i \in \tilde{S}$, it selects random $\gamma_i' \in \mathbb{Z}_N$ and creates a public key by implicitly setting $\gamma_i = \gamma_i' - \alpha^i$ as

$$UPK_i = \left(V_j = A_j^{-1} g^{\gamma_j'},\ \hat{V}_j = \hat{A}_j^{-1} \hat{g}^{\gamma_j'},\ \left\{V_{j,k} = A_{k+j}^{-1} A_k^{\gamma_j'}\right\}_{2\leq k\neq L+2-j\leq L+1}\right).$$

It gives $\{(j, UPK_j)\}_{j\in\tilde{S}}$ to $\mathcal{A}$.

For a decryption query on $(S, CH = (\hat{C}_1, C_2), i, AU)$ such that $S \subseteq \tilde{S}$, $\mathcal{B}_1$ proceeds as follows: It first computes $\omega = \mathbf{H}_1(\hat{C}_1 \| AU)$ and checks the validity of the ciphertext header by using pairing. If the check fails, then it responds with $\perp$. If $\omega = \omega^*$, then it sets $\mathsf{Bad} = 1$ and aborts the simulation. Otherwise ($\omega \neq \omega^*$), it selects a random exponent $r' \in \mathbb{Z}_p$ and derives decryption components by implicitly setting $r = a/(\omega - \omega^*) + r'$ as

$$D_1 = A_{L+2-i}^{\gamma_i'} (A_1^{\beta'} \prod_{j\in S} A_{j+1} V_{j,1})^{1/(\omega-\omega^*)} \left(A_{L+1}^{\omega} B \prod_{j\in S} A_j V_j\right)^{r'},$$
$$D_2 = A_{L+2-i} A_1^{1/(\omega-\omega^*)} g^{r'},$$
$$D_3 = A_{2L+3-i}^{\omega} B_{L+2-i},\ D_4 = \prod_{j\in S\setminus\{i\}} A_{L+2-i+j} V_{j,L+2-i}.$$

It responds a session key $CK = \mathbf{H}_2(e(C_2, D_2) \cdot e(D_1 \cdot D_3 \cdot D_4, \hat{C}_1)^{-1})$ to $\mathcal{A}$.

Challenge: $\mathcal{A}$ submits a challenge set $S^* \subseteq \tilde{S}$. $\mathcal{B}_1$ implicitly sets $t = b$ and creates a ciphertext tuple

$$CH^* = \left(\hat{C}_1^* = \hat{g}^b,\ C_2^* = \left(g^b\right)^{\beta' + \sum_{j \in S^*} \gamma'_j}\right),\ CK^* = \mathbf{H}_2(Z).$$

It sets $CK_0^* = CK^*$ and $CK_1^* = \mathbf{H}_2(R)$ by selecting a random element $R \in \mathbb{G}_T$. It flips a random bit $\mu \in \{0,1\}$ and gives (CH^*, CK_μ^*) to $\mathcal{A}$.

Query Phase 2: The decryption queries are handled as the same as query phase 1.

Guess: Finally, $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$. $\mathcal{B}_1$ outputs 0 if $\mu = \mu'$ or 1 otherwise.

The decryption components are correctly distributed as

$$\begin{aligned}
D_1 &= K_i \left(A_{L+1}^{\omega} B \prod_{j \in S} A_j V_j\right)^r = A_{L+2-i}^{\gamma'_i - \alpha^i} \left(A_{L+1}^{\omega - \omega^*} g^{\beta'} \prod_{j \in S} A_j V_j\right)^{a/(\omega - \omega^*) + r'} \\
&= A_{L+2-i}^{\gamma'_i} A_{L+2}^{-1} A_{L+2} \left(g^{\beta'} \prod_{j \in S} A_j V_j\right)^{a/(\omega - \omega^*)} \left(A_{L+1}^{\omega} B \prod_{j \in S} A_j V_j\right)^{r'} \\
&= A_{L+2-i}^{\gamma'_i} (A_1^{\beta'} \prod_{j \in S} A_{j+1} V_{j,1})^{1/(\omega - \omega^*)} \left(A_{L+1}^{\omega} B \prod_{j \in S} A_j V_j\right)^{r'}, \\
D_2 &= A_{L+2-i} g^r = A_{L+2-i} g^{a/(\omega - \omega^*) + r'} = A_{L+2-i} A_1^{1/(\omega - \omega^*)} g^{r'}.
\end{aligned}$$

The challenge ciphertext header are also correctly distributed as

$$C_2^* = \left(A_{L+1}^{\omega^*} B \prod_{j \in S^*} A_j V_j\right)^t = \left(A_{L+1}^{\omega^*} A_{L+1}^{-\omega^*} g^{\beta'} \prod_{j \in S^*} g^{\alpha^j} g^{\gamma'_j - \alpha^j}\right)^t = \left(g^t\right)^{\beta' + \sum_{j \in S^*} \gamma'_j}.$$

Suppose there exists an adversary $\mathcal{A}$ that breaks the DBE$_{SS}$ scheme with a non-negligible advantage. A simulator $\mathcal{B}_2$ that breaks the CRHF using $\mathcal{A}$ is given the description of $\mathbf{H}_1$. Then $\mathcal{B}_2$ that interacts with $\mathcal{A}$ is described as follows:

Init: $\mathcal{A}$ commits an initial set $\tilde{S}$.

Setup: $\mathcal{B}_2$ creates public parameters by running the setup algorithm except that $\mathbf{H}_1$ is replaced by the given hash function. It selects a random exponent $t \in \mathbb{Z}_p$, sets $\hat{C}_1^* = \hat{g}^t$, and computes $\omega^* = \mathbf{H}_1(\hat{C}_1^*)$.

Query Phase 1: For all $j \in \tilde{S}$, $\mathcal{B}_2$ generates (USK_j, UPK_j) by simply running the key generation algorithm. It gives $\{(j, UPK_j)\}_{j \in \tilde{S}}$ to $\mathcal{A}$. For a decryption query on $(S, CH = (\hat{C}_1, C_2), i, AU)$, it first computes $\omega = \mathbf{H}_1(\hat{C}_1 \| AU)$. If $\omega = \omega^*$, then it sets $\mathsf{Bad} = 1$ and outputs a collision pair $(\hat{C}_1^*, \hat{C}_1 \| AU)$. Otherwise, it responds with a session key by simply running the decryption algorithm.

Challenge: To create a ciphertext (CH^*, CK^*), $\mathcal{B}_2$ simply runs the encapsulation algorithm except that it uses the exponent t chosen in the setup. It sets $CK_0^* = CK^*$ and $CK_1^* = RK$ by selecting a random RK. It flips a coin $\mu \in \{0,1\}$ and gives (CH^*, CK_μ^*) to $\mathcal{A}$.

Query Phase 2: The decryption queries are handled as the same as query phase 1.

Guess: $\mathcal{B}_2$ outputs $\perp$.

By combining the results of the above simulations, we obtain the following equation

$$\begin{aligned}
\mathbf{Adv}_{DBE}^{SS\text{-}CCA}(\lambda) &= \Pr[\mu = \mu'] - 1/2 \\
&= \Pr[\neg \mathsf{Bad}] \cdot \Pr[\mu = \mu' | \neg \mathsf{Bad}] + \Pr[\mathsf{Bad}] \cdot \Pr[\mu = \mu' | \mathsf{Bad}] - 1/2 \\
&\leq \Pr[\mu = \mu' | \neg \mathsf{Bad}] - 1/2 + \Pr[\mathsf{Bad}] \leq \mathbf{Adv}_{B_1}^{BDHE}(\lambda) + \mathbf{Adv}_{B_2}^{CRHF}(\lambda).
\end{aligned}$$

This completes our proof. □

5.2 Adaptive Security Analysis

The basic idea of the adaptive CCA proof is to change the challenge session key to a random value through hybrid games so that the attacker can never distinguish the coin thrown by the challenger. To do this, we first play hybrid games to change the secret keys used for CT_0^*, CT_1^* in the challenge ciphertext header CH^* to random values by using the semi-static CCA security of the underlying DBE_{SS} scheme. Then, we play hybrid games to change the messages of CT_0^*, CT_1^* to random values by using the OMI security of the underlying SKE scheme. In the last game, the challenge session key CK_μ^* is not related to the challenge ciphertext header CH^*. The detailed description of the adaptive CCA proof is given as follows:

Theorem 5.2 (Adaptive CCA Security)**.** *The above DBE_{AD} scheme is AD-CCA secure if the DBE_{SS} scheme is SS-CCA secure and the OTS scheme is strongly unforgeable.*

Proof. The security proof consists of a sequence of hybrid games $\mathbf{G}_0, \mathbf{G}_1, \ldots, \mathbf{G}_4$. The first game will be the original adaptive CCA security game and the last one will be a game in which an adversary has no advantage. We define the games as follows:

Game $\mathbf{G}_0$. This game is the original security game defined in Section 3.3. That is, the simulator of this game simply follows the honest algorithms.

Game $\mathbf{G}_1$. This game is the same as the game $\mathbf{G}_0$ except that the ciphertext CT_0 in CH^* is created as $CT_0 = \mathbf{SKE.Encrypt}(RK_{SS,0}, CK_0)$ by using a random $RK_{SS,0}$ instead of a valid $CK_{SS,0}$.

Game $\mathbf{G}_2$. This game is also similar to the game $\mathbf{G}_1$ except that the ciphertext CT_1 in CH^* is created as $CT_1 = \mathbf{SKE.Encrypt}(RK_{SS,1}, CK_0)$ by using random $RK_{SS,1}$ instead of a valid $CK_{SS,1}$.

Game $\mathbf{G}_3$. This game is the same as the game $\mathbf{G}_2$ except that the ciphertext CT_0 in CH^* is generated as $CT_0 = \mathbf{SKE.Encrypt}(RK_{SS,0}, RK_0)$ by selecting a random RK_0 instead of a valid CK.

Game $\mathbf{G}_4$. This final game $\mathbf{G}_4$ is also similar to the game $\mathbf{G}_3$ except that the ciphertext CT_1 in CH^* is generated as $CT_1 = \mathbf{SKE.Encrypt}(RK_{SS,1}, RK_1)$ by selecting a random RK_1 instead of a valid CK. In this final game, CH^* is not related to the session key CK_μ^*. Thus, the advantage of $\mathcal{A}$ is zero.

Let $\mathbf{Adv}_{\mathcal{A}}^{G_j}$ be the advantage of $\mathcal{A}$ in the game $\mathbf{G}_j$. We have that $\mathbf{Adv}_{DBE,\mathcal{A}}^{AD\text{-}CCA}(\lambda) = \mathbf{Adv}_{\mathcal{A}}^{G_0}$, and $\mathbf{Adv}_{\mathcal{A}}^{G_4} = 0$. From the following Lemmas 5.3, 5.4, 5.5, and 5.6, we obtain the equation

$$\begin{aligned}\mathbf{Adv}_{DBE}^{AD\text{-}CCA}(\lambda) &\leq \sum_{j=1}^{4} \left|\mathbf{Adv}_{\mathcal{A}}^{G_{j-1}} - \mathbf{Adv}_{\mathcal{A}}^{G_j}\right| + \mathbf{Adv}_{\mathcal{A}}^{G_4} \\ &\leq 2\mathbf{Adv}_{DBE}^{SS\text{-}CCA}(\lambda) + 2\mathbf{Adv}_{OTS}^{SUF}(\lambda) + 2\mathbf{Adv}_{SKE}^{OMI}(\lambda).\end{aligned}$$

This completes the proof. □

Lemma 5.3. *If the DBE_{SS} scheme is SS-CCA secure and the OTS scheme is SUF secure, then no PPT adversary can distinguish $\mathbf{G}_0$ from $\mathbf{G}_1$ with a non-negligible advantage.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that distinguishes $\mathbf{G}_0$ from $\mathbf{G}_1$ with a non-negligible advantage. Let $\mathcal{C}$ be the challenger of the DBE_{SS} scheme with an input 1^{2L}. The simulator $\mathcal{B}_1$ that breaks the DBE_{SS} scheme by using $\mathcal{A}$ with an input 1^L is described as follows:

Setup: In this step, $\mathcal{B}_1$ proceeds as follows:

1. It chooses random bits $z^* = \{z_j^*\}_{j\in[L]}$ where $z_j \in \{0,1\}$ and defines an initial set $\tilde{S}_{SS} = \{2j - z_j^*\}_{j=1}^{L}$. It also generates (SK^*, VK^*) by running **OTS.GenKey**(1^λ).
2. It commits $\tilde{S}_{SS}$ to $\mathcal{C}$ and receives PP_{SS}. It sets $PP = PP_{SS}$ and gives PP to $\mathcal{A}$.
3. Next, it requests a key generation query to $\mathcal{C}$ and receives all public keys $\{(k, UPK_{SS,k})\}_{k\in\tilde{S}_{SS}}$.

Query Phase 1: For the key generation query of $\mathcal{A}$ on an index $i \in [L]$ such that $i \notin KQ$, $\mathcal{B}_1$ proceeds as follows:

1. It sets $k = 2i - (1 - z_i^*)$ and obtains $(USK_{SS,k}, UPK_{SS,k})$ by running $\mathbf{DBE}_{SS}$.**GenKey**(k, PP_{SS}). It stores $USK_{SS,k}$ to KL.
2. It sets $UPK_i = (UPK_{SS,2i}, UPK_{SS,2i-1})$, adds i to KQ, and gives UPK_i to $\mathcal{A}$.

For the key corruption query of $\mathcal{A}$ on an index $i \in [L]$ such that $i \in KQ \setminus DQ$, $\mathcal{B}_1$ proceeds as follows:

1. It fixes $u_i = 1 - z_i^*$ and retrieves $USK_{SS,2i-u_i}$ from KL.
2. It sets $USK_i = (USK_{SS,2i-u_i}, u_i)$, adds i to CQ, and gives USK_i to $\mathcal{A}$.

For the decryption query of $\mathcal{A}$ on $(S, CH = (CM, \sigma, VK), i, AU)$ such that $S \subseteq KQ \setminus CQ$ where $CM = (CH_{SS,0}, CH_{SS,1}, CT_0, CT_1, z)$, $\mathcal{B}_1$ proceeds as follows:

1. If $VK = VK^*$, then it sets $\mathsf{Bad} = 1$ and aborts the simulation.
2. It first checks $1 \stackrel{?}{=}$ **OTS.Verify**(CM, σ, VK). If this check fails, it gives $\perp$ to $\mathcal{A}$.
3. If $(i, u_i) \in UL$, then it retrieves u_i. Otherwise, It selects a random bit $u_i \in \{0,1\}$ and adds (i, u_i) to UL.
4. It defines $S_0 = \{2j - z_j\}_{j\in S}$ and $S_1 = \{2j - (1 - z_j)\}_{j\in S}$. If $z_i = u_i$, then it sets $S' = S_0, CH'_{SS} = CH_{SS,0}, CT' = CT_0$. Otherwise, it sets $S' = S_1, CH'_{SS} = CH_{SS,1}, CT' = CT_1$.
5. If $u_i = 1 - z_i^*$, then it retrieves $USK_{SS,2i-u_i}$ from KL and obtains CK'_{SS} by running $\mathbf{DBE}_{SS}$.**Decaps**$(S', CH'_{SS}, 2i - u_i, USK_{SS,2i-u_i}, \{(k, UPK_{SS,k})\}_{k\in S'}, PP_{SS}, VK)$. Otherwise ($u_i = z_i^*$), it requests a decryption query on $(S', CH'_{SS}, 2i - u_i, VK)$ to $\mathcal{C}$ and receives CK'_{SS}.
6. It derives CK by running **SKE.Decrypt**(CT', CK'_{SS}). It adds i to DQ and gives CK to $\mathcal{A}$.

Challenge: $\mathcal{A}$ submits a challenge set $S^* \subseteq [L]$ such that $S^* \subseteq KQ \setminus CQ$, $\mathcal{B}_1$ proceeds as follows:

1. It defines $S_0^* = \{2j - z_j^*\}_{j\in S^*}$ and $S_1^* = \{2j - (1 - z_j^*)\}_{j\in S^*}$. It submits a challenge set S_0^* to $\mathcal{C}$ and receives $(CH_{SS,0}^*, CK_{SS,0}^*)$. It obtains $(CH_{SS,1}^*, CK_{SS,1}^*)$ by running $\mathbf{DBE}_{SS}$.**Encaps**$(S_1^*, \{(k, UPK_{SS,k})\}_{k\in S_1^*}, PP_{SS}, VK^*)$.
2. It selects a session key CK^*. It obtains CT_0^* and CT_1^* by running **SKE.Encrypt**$(CK_{SS,0}^*, CK^*)$ and **SKE.Encrypt**$(CK_{SS,1}^*, CK^*)$ respectively.
3. It sets $CM^* = (CH_{SS,0}^*, CH_{SS,1}^*, CT_0^*, CT_1^*, z^*)$ and calculates σ^* by running **OTS.Sign**(SK^*, CM^*).
4. It sets $CH^* = (CM^*, \sigma^*, VK^*)$, $CK_0^* = CK^*$, and $CK_1^* = RK$ by selecting a random RK. Next, it flips a random coin $\mu \in \{0,1\}$ and gives (CH^*, CK_μ^*) to $\mathcal{A}$

Query Phase 2: For a decryption query on $(S, CH = (CM, \sigma, VK), i, AU)$ such that $S \subseteq S^* \wedge CH \neq CH^*$, $\mathcal{B}_1$ handles this decryption query as the same as query phase 1.

Guess: $\mathcal{A}$ outputs a guess μ'. If $\mu = \mu'$, then $\mathcal{B}_1$ outputs 1. Otherwise, it outputs 0.

Suppose there exists an adversary $\mathcal{A}$ that breaks the DBE_{SS} scheme with a non-negligible advantage. A simulator $\mathcal{B}_2$ that forges the OTS scheme using $\mathcal{A}$ is given VK^*. Then $\mathcal{B}_2$ that interacts with $\mathcal{A}$ is described as follows:

Setup: $\mathcal{B}_2$ creates public parameters by running the setup algorithm.

Query Phase 1: For a key generation query, $\mathcal{B}_2$ generates (USK_j, UPK_j) by simply running the key generation algorithm. It gives UPK_j to $\mathcal{A}$. For a key corruption query, it simply responds with USK_j to $\mathcal{A}$. For a decryption query on $(S, CH = (CM, \sigma, VK), i, AU)$, it first verify the validity of σ by running the verification algorithm of OTS. If $VK = VK^*$, then it sets $\mathsf{Bad} = 1$ and outputs a forgery (σ, CM). Otherwise, it responds with a session key by simply running the decryption algorithm.

Challenge: To create a ciphertext (CH^*, CK^*), $\mathcal{B}_2$ simply runs the encapsulation algorithm except that it uses the signing oracle of OTS to create σ^*. It sets $CK_0^* = CK^*$ and $CK_1^* = RK$ by selecting a random RK. It flips a coin $\mu \in \{0,1\}$ and gives (CH^*, CK_μ^*) to $\mathcal{A}$.

Query Phase 2: The decryption queries are handled as the same as query phase 1.

Guess: $\mathcal{B}_2$ outputs $\perp$.

By combining the analysis of two simulators, we obtain the following equation

$$\mathbf{Adv}_{\mathcal{A}}^{G_0} - \mathbf{Adv}_{\mathcal{A}}^{G_1} \leq \mathbf{Adv}_{DBE}^{SS-CCA}(\lambda) + \mathbf{Adv}_{OTS}^{SUF}(\lambda).$$

This completes our proof. □

Lemma 5.4. *If the DBE_{SS} scheme is SS-CCA secure and the OTS scheme is SUF secure, then no PPT adversary can distinguish $\boldsymbol{G}_1$ from $\boldsymbol{G}_2$ with a non-negligible advantage.*

Proof. The proof of this lemma is almost the same as that of Lemma 5.3 except that a simulator $\mathcal{B}_1$ that breaks the DBE_{SS} scheme defines the initial set $\tilde{S}_{SS} = \{2j - (1 - z_j^*)\}_{j=1}^{L}$ in the setup phase. Because of this change, the key generation, key corruption, decryption queries are slightly changed. In the challenge step, the set S_1^* become the challenge set. The detailed description of the challenge setup is given as follows:

Challenge: $\mathcal{A}$ submits a challenge set $S^* \subseteq [L]$ such that $S^* \subseteq KQ \setminus CQ$, $\mathcal{B}$ proceeds as follows:

1. It defines $S_0^* = \{2j - z_j^*\}_{j \in S^*}$ and $S_1^* = \{2j - (1 - z_j^*)\}_{j \in S^*}$. It obtains $(CH_{SS,0}^*, CK_{SS,0}^*)$ by running $\mathbf{DBE}_{SS}.\mathbf{Encaps}(S_0^*, \{(k, UPK_{SS,k})\}_{k \in S_0^*}, PP_{SS}, VK^*)$. It submits a challenge set S_1^* to $\mathcal{C}$ and receives $(CH_{SS,1}^*, CK_{SS,1}^*)$. It selects a random $RK_{SS,0}$.

2. It selects a session key CK^*. It obtains CT_0^* and CT_1^* by running $\mathbf{SKE.Encrypt}(RK_{SS,0}, CK^*)$ and $\mathbf{SKE.Encrypt}(CK_{SS,1}^*, CK^*)$ respectively.

3. It sets $CM^* = (CH_{SS,0}^*, CH_{SS,1}^*, CT_0^*, CT_1^*, z^*)$ and calculates σ^* by running $\mathbf{OTS.Sign}(SK^*, CM^*)$.

4. It sets $CH^* = (CM^*, \sigma^*, VK^*)$, $CK_0^* = CK^*$, and $CK_1^* = RK$ by selecting a random RK. Next, it flips a random coin $\mu \in \{0,1\}$ and gives (CH^*, CK_μ^*) to $\mathcal{A}$

The description of a simulator $\mathcal{B}_2$ that breaks the OTS scheme is the same as that of Lemma 5.3. We omit the description of this simulator.

Similar to the analysis of Lemma 5.3, we obtain the following equation

$$\mathbf{Adv}_{\mathcal{A}}^{G_1} - \mathbf{Adv}_{\mathcal{A}}^{G_2} \leq \mathbf{Adv}_{DBE}^{SS-CCA}(\lambda) + \mathbf{Adv}_{OTS}^{SUF}(\lambda).$$

This completes our proof. □

Lemma 5.5. *If the SKE scheme is OMI secure, then no PPT adversary can distinguish $\mathbf{G}_2$ from $\mathbf{G}_3$ with a non-negligible advantage.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that distinguishes $\mathbf{G}_2$ from $\mathbf{G}_3$ with a non-negligible advantage. Let $\mathcal{C}$ be a challenger of the SKE scheme. A simulator $\mathcal{B}$ that breaks the SKE scheme by using $\mathcal{A}$ with an input 1^L is described as follows:

Setup: In this step, $\mathcal{B}$ simply runs the normal setup algorithm.

Query Phase 1: For the key generation, key corruption, and decryption queries of $\mathcal{A}$, $\mathcal{B}$ simply handles these queries by running the normal algorithms of DBE.

Challenge: $\mathcal{A}$ submits a challenge set $S^* \subseteq [L]$, $\mathcal{B}$ proceeds as follows:

1. It generates (SK^*, VK^*) by running **OTS.GenKey**(1^λ).
2. It defines $S_0^* = \{2j - z_j^*\}_{j \in S^*}$ and $S_1^* = \{2j - (1 - z_j^*)\}_{j \in S^*}$. It obtains $(CH_{SS,0}^*, CK_{SS,0}^*)$ by running $\mathbf{DBE}_{SS}$.**Encaps**$(S_0^*, \{(k, UPK_{SS,k})\}_{k \in S_0^*}, PP_{SS}, VK^*)$. It also obtains $(CH_{SS,1}^*, CK_{SS,1}^*)$ by running $\mathbf{DBE}_{SS}$. **Encaps**$(S_1^*, \{(k, UPK_{SS,k})\}_{k \in S_1^*}, PP_{SS}, VK^*)$. It selects a random $RK_{SS,1}$.
3. It selects a session key CK^*. It also selects a random RK_0. It submits (CK^*, RK_0) to $\mathcal{C}$ and receives CT_0^*. It obtains CT_1^* by running **SKE.Encrypt**$(RK_{SS,1}, CK^*)$.
4. It sets $CM^* = (CH_{SS,0}^*, CH_{SS,1}^*, CT_0^*, CT_1^*, z^*)$ and calculates σ^* by running **OTS.Sign**(SK^*, CM^*).
5. It sets $CH^* = (CM^*, \sigma^*, VK^*)$, $CK_0^* = CK^*$, and $CK_1^* = RK$ by selecting a random RK. Next, it flips a random coin $\mu \in \{0,1\}$ and gives (CH^*, CK_μ^*) to $\mathcal{A}$

Query Phase 2: For a decryption query, $\mathcal{B}$ handles this decryption query as the same as query phase 1.

Guess: $\mathcal{A}$ outputs a guess μ'. If $\mu = \mu'$, then $\mathcal{B}$ outputs 1. Otherwise, it outputs 0. □

Lemma 5.6. *If the SKE scheme is OMI secure, then no PPT adversary can distinguish $\mathbf{G}_3$ from $\mathbf{G}_4$ with a non-negligible advantage.*

Proof. The proof of this lemma is almost the same as that of Lemma 5.5 except the generation of the challenge ciphertext. The detailed description of the challenge setup is given as follows:

Challenge: $\mathcal{A}$ submits a challenge set $S^* \subseteq [L]$, $\mathcal{B}$ proceeds as follows:

1. It generates (SK^*, VK^*) by running **OTS.GenKey**(1^λ).
2. It defines $S_0^* = \{2j - z_j^*\}_{j \in S^*}$ and $S_1^* = \{2j - (1 - z_j^*)\}_{j \in S^*}$. It obtains $(CH_{SS,0}^*, CK_{SS,0}^*)$ by running $\mathbf{DBE}_{SS}$.**Encaps**$(S_0^*, \{(k, UPK_{SS,k})\}_{k \in S_0^*}, PP_{SS}, VK^*)$. It also obtains $(CH_{SS,1}^*, CK_{SS,1}^*)$ by running $\mathbf{DBE}_{SS}$. **Encaps**$(S_1^*, \{(k, UPK_{SS,k})\}_{k \in S_1^*}, PP_{SS}, VK^*)$. It selects a random $RK_{SS,0}$.
3. It selects a session key CK^*. It also selects random RK_0 and RK_1. It obtains CT_0^* by running **SKE.Encrypt**$(RK_{SS,0}, RK_0)$. It submits (CK^*, RK_1) to $\mathcal{C}$ and receives CT_1^*.

4. It sets $CM^* = (CH^*_{SS,0}, CH^*_{SS,1}, CT^*_0, CT^*_1, z^*)$ and calculates σ^* by running **OTS.Sign**(SK^*, CM^*).

5. It sets $CH^* = (CM^*, \sigma^*, VK^*)$, $CK^*_0 = CK^*$, and $CK^*_1 = RK$ by selecting a random RK. Next, it flips a random coin $\mu \in \{0,1\}$ and gives (CH^*, CK^*_μ) to $\mathcal{A}$

This completes our proof. □

6 Conclusion

In this paper, we proposed an efficient DBE scheme in bilinear groups and proved its adaptive CCA security under the q-Type assumption. Our adaptive CCA secure DBE scheme has constant size ciphertexts, constant size private keys, and linear size public keys. The public key verification of our DBE scheme requires only a constant number of pairing operations and the linear number of efficient group membership operations. An interesting problem is to design an efficient and adaptive CCA secure DBE scheme under standard assumptions weaker than the q-Type assumption.

References

[1] Michel Abdalla, Eike Kiltz, and Gregory Neven. Generalized key delegation for hierarchical identity-based encryption. In Joachim Biskup and Javier Lopez, editors, *Computer Security - ESORICS 2007*, volume 4734 of *Lecture Notes in Computer Science*, pages 139–154. Springer, 2007.

[2] Shweta Agrawal, Daniel Wichs, and Shota Yamada. Optimal broadcast encryption from LWE and pairings in the standard model. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - TCC 2020*, volume 12550 of *Lecture Notes in Computer Science*, pages 149–178. Springer, 2020.

[3] Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and LWE. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020*, volume 12105 of *Lecture Notes in Computer Science*, pages 13–43. Springer, 2020.

[4] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004.

[5] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.

[6] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[7] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005.

[8] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptol.*, 17(4):297–319, 2004.

[9] Dan Boneh, Brent Waters, and Mark Zhandry. Low overhead broadcast encryption from multilinear maps. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014*, volume 8616 of *Lecture Notes in Computer Science*, pages 206–223. Springer, 2014.

[10] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014*, volume 8616 of *Lecture Notes in Computer Science*, pages 480–499. Springer, 2014.

[11] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security, CCS 2005*, pages 320–329. ACM, 2005.

[12] Jan Camenisch, Susan Hohenberger, and Michael Østergaard Pedersen. Batch verification of short signatures. *J. Cryptology*, 25(4):723–747, 2012.

[13] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2004.

[14] Jeffrey Champion and David J. Wu. Distributed broadcast encryption from lattices. In Elette Boyle and Mohammad Mahmoody, editors, *Theory of Cryptography - TCC 2024*, volume 15366 of *Lecture Notes in Computer Science*, pages 156–189. Springer, 2024.

[15] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2002.

[16] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2003.

[17] Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In Kaoru Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 200–215. Springer, 2007.

[18] Yevgeniy Dodis and Jonathan Katz. Chosen-ciphertext security of multiple encryption. In Joe Kilian, editor, *Theory of Cryptography - TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 188–209. Springer, 2005.

[19] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, 1993.

[20] Rachit Garg, George Lu, Brent Waters, and David J. Wu. Realizing flexible broadcast encryption: How to broadcast to a public-key directory. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM Conference on Computer and Communications Security, CCS 2023*, pages 1093–1107. ACM, 2023.

[21] Romain Gay, Lucas Kowalczyk, and Hoeteck Wee. Tight adaptively secure broadcast encryption with short ciphertexts and keys. In Dario Catalano and Roberto De Prisco, editors, *Security and Cryptography for Networks - SCN 2018*, volume 11035 of *Lecture Notes in Computer Science*, pages 123–139. Springer, 2018.

[22] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 171–188. Springer, 2009.

[23] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography - TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 581–600. Springer, 2006.

[24] Dimitris Kolonelos, Giulio Malavolta, and Hoeteck Wee. Distributed broadcast encryption from bilinear groups. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023*, volume 14442 of *Lecture Notes in Computer Science*, pages 407–441. Springer, 2023.

[25] Kwangsu Lee. Adaptively secure distributed broadcast encryption with linear-size public parameters. arXiv:2505.17527, 2025. `https://arxiv.org/abs/2505.17527`.

[26] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, 2001.

[27] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 1991.

[28] Victor Shoup. A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001. `http://eprint.iacr.org/2001/112`.

[29] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer, 2009.

[30] Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022*, volume 13276 of *Lecture Notes in Computer Science*, pages 217–241. Springer, 2022.

[31] Qianhong Wu, Bo Qin, Lei Zhang, and Josep Domingo-Ferrer. Ad hoc broadcast encryption. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM Conference on Computer and Communications Security, CCS 2010*, pages 741–743. ACM, 2010.