Hindawi
Security and Communication Networks
Volume 2021, Article ID 6639835, 17 pages
https://doi.org/10.1155/2021/6639835

Research Article

Anonymous Certificate-Based Inner Product Broadcast Encryption

Shuang Yao [1,2] **and Dawei Zhang** [1,2]

[1]*Department of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China*
[2]*Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing 100044, China*

Correspondence should be addressed to Dawei Zhang; dwzhang@bjtu.edu.cn

Received 8 October 2020; Revised 20 May 2021; Accepted 27 May 2021; Published 31 August 2021

Academic Editor: Leandros Maglaras

Broadcast encryption scheme enables a sender distribute the confidential content to a certain set of intended recipients. It has been applied in cloud computing, TV broadcasts, and many other scenarios. Inner product broadcast encryption takes merits of both broadcast encryption and inner product encryption. However, it is crucial to reduce the computation cost and to take the recipient's privacy into consideration in the inner product broadcast encryption scheme. In order to address these problems, we focus on constructing a secure and practical inner product broadcast encryption scheme in this paper. First, we build an anonymous certificate-based inner product broadcast encryption scheme. Especially, we give the concrete construction and security analysis. Second, compared with the existing inner product broadcast encryption schemes, the proposed scheme has an advantage of anonymity. Security proofs show that the proposed scheme achieves confidentiality and anonymity against adaptive chosen-ciphertext attacks. Finally, we implement the proposed anonymous inner product broadcast encryption scheme and evaluate its performance. Test results show that the proposed scheme supports faster decryption operations and has higher efficiency.

1. Introduction

Broadcast encryption is an efficient way to make secure group-oriented communication by distributing confidential information in an open channel to a certain set of intended recipients that are selected by the sender. In a broadcast encryption scheme, the sender sends a ciphertext containing secret messages, and the ciphertext is only readable by privileged users. Broadcast encryption has been applied to various scenarios such as GPS, TV broadcasts, and radio broadcasts and may be potentially applied to the blockchain to perform one-to-many information exchange in some scenarios.

There are two types of broadcast encryption schemes in the literature: one is symmetric key broadcast encryption [1] and the other is public key broadcast encryption [2]. In terms of symmetric key broadcast encryption, it generates private keys for all users through a trusted center which also broadcasts messages to the intended recipients. It is obvious that the symmetric key broadcast encryption is infeasible to most of broadcast scenarios due to its possibility of single-point failure. In contrast, any user can be a sender in the public key broadcast encryption scheme. It overcomes the shortcoming of single-point failure in the symmetric key scheme. However, there are certificate management problems in the public key broadcast encryption scheme.

Function encryption (FE) [3] is different from traditional encryption. Only owners of legitimated keys are able to learn the whole underlying data through the decryption of the ciphertext, while others obtain nothing in traditional encryption. Function encryption can control information amount in the ciphertext transmitted to recipients. Furthermore, the functional encryption for inner product (IPFE) enables the recipient to decrypt the ciphertext related to the vector $\vec{x}$ with the private key related to the vector $\vec{y}$. It will only obtain inner product $\langle \vec{x}, \vec{y} \rangle$ and nothing else.

Inner product encryption is simple, but it can provide powerful function. IPFE has been suggested to be applied in many scenarios such as delegation of sensitive computation and biometric authentication [4–6]. In some application scenarios, besides focusing on the privacy of encrypted messages, it is also significant to consider the privacy of the function being computed. Function hiding is an essential property of function encryption which means that the secret key can also hide the function f, and no one could learn any unnecessary information about f [7].

In recent years, the notion of inner product broadcast encryption has been proposed [8]. One might think a trivial solution which encrypts the message under the inner product encryption first and then encrypts the ciphertext with a broadcast encryption. However, this trivial solution has a security threat that if a recipient exposes its result obtained from the decryption of broadcast encryption no matter on purpose or not, all users in the inner product encryption system would be able to calculate their inner product values with their private keys. The broadcast encryption for inner product avoids this security threat. It takes merits of both broadcast encryption and inner product encryption. In the inner product broadcast encryption scheme, the recipient can only obtain the inner product associated with the encrypted message by providing their secret keys in the decryption period. The sender determines who can obtain the corresponding inner product value.

With the rapid development of information technology and the continuous upgrading of new techniques such as the Internet of Things (IoTs) and blockchain, broadcast encryption has been applied to these new scenarios to provide data security and to guarantee user privacy. In smart communities, it has been used for the information management center to send the encrypted information to some units and individuals that guarantees the secure transmission of information within the community [9]. In the blockchain, it has been applied to achieve group communication and protect the privacy of transaction data in the system [10]. As for the inner product broadcast encryption, it can determine who is able to obtain the plaintext and can give further protection to the plaintext. We pay attention to a personal skill evaluation system which was introduced and described in [8]. For instance, a student gets grades of mathematics 90, communication 80, and programming 60 that are represented by private vector $\overrightarrow{y} = (90, 80, 60)$. If a company wants to know whether a student is suitable for an occupation, it can evaluate the student by computing the weighted average of the scores $\langle \overrightarrow{y}, \overrightarrow{x} \rangle = 90^*50\% + 80^*30\% + 60^*20\% = 81$. $\overrightarrow{x} = (50\%, 30\%, 20\%)$ represents weights to each of the above scores.

1.1. Motivation. Broadcast encryption for the inner product has quite huge application potential. There are some research works that have been undertaken to provide inner product broadcast encryption schemes, and there also exist some shortcomings in the present schemes. First, to the best of our knowledge, the existing schemes do not take the recipient's identity privacy into consideration. Second, the existing scheme achieves selective CPA security. Third, the heavy decryption cost and large public parameters' size in the present schemes can bring down the efficiency for those applications in that recipients' computing ability is limited, and they do not implement their proposed schemes for performance evaluation. At last, the existing scheme constructed in the identity-based cryptosystem has key escrow problems that the key generation center has the ability to decrypt all the encrypted messages in the system compared to certificate-based schemes [11]. Certificate-based broadcast encryption has attracted more and more attention [12, 13]. It has the feature of decentralization which makes it more suitable to be applied in the blockchain, so we build our scheme in the certificate-based cryptosystem. The motivation of this paper is to build a more feasible inner product broadcast encryption scheme with anonymity property. This new construction is also more suitable to be applied to those scenarios whose broadcast plaintext needs further protection. The goals of our scheme can be summarized as the following:

In terms of security, we aim to provide adaptive CCA security in the random oracle model

In the aspect of recipient privacy, we aim to provide anonymity that an encrypted broadcast message should hide who can access its contents; even users in the intended recipient set are not able to recognize other users' identities

In terms of efficiency, we aim to have lower computational overhead in the proposed scheme

1.2. Contribution. To summarize, we make the following contributions in this paper:

We design an efficient certificate-based inner product broadcast encryption (CBBE-IP) with anonymity property. Compared with the existing construction, the proposed anonymous scheme takes the recipient's identity privacy into consideration. A user cannot obtain other recipients' identities, even from each other in the set of authorized recipients in the proposed scheme. It achieves stronger privacy protection.

We give the formal proofs under the random oracle model to claim that our construction is confidential and anonymous. It is secure under the adaptive chosen-ciphertext attack.

We give the theoretical analysis of our proposed scheme's efficiency. We also implement both our scheme and the IBBE-IP scheme in Python and evaluate their performance. Experimental and theoretical analysis results show that the proposed scheme has higher efficiency, which enables faster decryption. In addition, our scheme has no restriction that the recipient number has to be less than vector length ($n < d$).

1.3. Related Work. In recent years, great efforts have been devoted to construct inner product encryption and broadcast encryption.

As for inner product encryption, Boneh et al. [3] took the formal study of functional encryption and gave precise definitions of the concept and security about functional encryption. Abdalla et al. [4] showed how to efficiently construct function encryption for the inner product under the standard assumption. Chotard et al. [14] introduced a primitive decentralized multiclient functional encryption (DMCFE) which combined techniques from private stream aggregation (PSA) and functional encryption for the inner product. The scheme can be applied in situations where multiple parties noninteractively share and update data.

Considering the function privacy, the notion of predicate privacy was first proposed by Shen et al. [15]. Since then, function-hiding inner product encryption has been deeply researched in numerous proposed papers. Bishop et al. [7] gave us the construction of secret-key function-hiding inner product encryption under the symmetric external Diffie–Hellman (SXDH) assumption in a quite weak and unrealistic security model. Datta et al. [16] proposed a simple and efficient private key IPE that has the strongest indistinguishability-based notion based on the SXDH assumption. Benhamouda et al. [17] proposed a generic construction of IND-CCA inner-product functional encryption from projective hash functions with homomorphic properties. Zhang et al. [18] proposed a generic construction of functional encryption for inner products that is IND-CCA secure. Abdalla et al. [19] proposed a novel methodology which is surprisingly simple and efficient to convert single-input IPE schemes into multi-input functional encryption (MIFE) schemes with the same functionality. Datta et al. [20] developed two nongeneric and practically efficient private key inner product MIFE schemes that first simultaneously achieved message and function privacy. Wang et al. [21] proposed two adaptively CCA-secure functional encryptions in the PKE and SKE settings, respectively. Kim et al. [5] focused on the practical applications of the above schemes; they proposed a fully secure, function-hiding inner product encryption scheme which has obvious shorter secret key and ciphertext compared with the existing schemes.

As for broadcast encryption, Fiat and Naor [1] gave the primitive formal definition of broadcast encryption which was a kind of symmetric key broadcast encryption. Naor and Pinkas [2] proposed the first public key broadcast encryption. Gay et al. [22] constructed a new scheme which was the first public key broadcast encryption scheme with constant size of the ciphertext and secret keys.

There have been considerable efforts devoted to build broadcast encryption such as identity-based broadcast encryption (IBBE), attribute-based broadcast encryption (ABBE), and certificate-based broadcast encryption (CBBE) with various functions. Delerablee [23] gave the first constant size of private keys and ciphertext. It is an identity-based broadcast encryption scheme with selective CPA security. Jiang et al. [24] proposed a keyword search identity-based broadcast encryption against insider attacks for cloud database systems. Lubicz and Sirvent [25] put forward the concept of attribute-based broadcast encryption by describing the group of privileged users through attributes. It allows one to select or revoke users. Xiong et al. [26] proposed a ciphertext-policy attribute-based encryption (CP-ABE) that, for the first time, realized partial policy hiding, direct revocation, and secure delegation simultaneously in edge computing. There have also been many more recent studies considering the case of ABBE in many fields [27, 28]. Barth et al. [29] took the user anonymity into consideration in broadcast encryption and put forward the concept of privacy in the broadcast encryption scheme. There is no reveal of intended recipients' identities in this scheme. Sur et al. [30] constructed the first certificate-based multireceiver encryption without formal definitions and proofs to the security. Then, Fan et al. [12] proposed an anonymous CBBE which defined the security models and offered formal proofs to all properties including anonymity. However, it only achieves CPA security and has expensive decryption cost. Zhu et al. [31] proposed adaptive security in the multichallenge setting with constant-size ciphertext header which is a strong security notion for broadcast encryption. Li et al. [32] put forward an anonymous CBBE scheme with constant decryption cost and adaptive CCA security. The CBBE construction avoids key escrow problems of identity-based broadcast encryption. Deng [9] constructed an anonymous certificateless multi-receiver encryption scheme for smart community management systems.

Jin and Yu-pu [33] proposed the notion of broadcast encryption for inner product predicate encryption under the standard model in 2012. The intended recipients output the plaintext via decryption in the scheme. Then, Lai et al. [8] constructed the first broadcast encryption for inner product scheme (IBBE-IP) under the random oracle model in 2018. It combines the IBBE [23] scheme and the inner product encryption (IPE) [4] scheme which outputs the real value of the inner product via decryption to the user and is a special functional encryption that has potential practical applications. However, these existing inner product broadcast encryption scheme and the inner product predicate broadcast encryption scheme do not take users' identity privacy into consideration. In this paper, we explore how to construct a more efficient and secure scheme of inner product broadcast encryption in order to extend its application scenarios.

1.4. Organization. We first recall some necessary preliminaries in Section 2, and then in Section 3, we describe the formal definitions and security model of our broadcast encryption scheme. In Section 4, we give the concrete construction of our scheme. We give the detailed security proof in Section 5. We then implement our broadcast encryption scheme and analyze its performance in Section 6. Conclusions are drawn in Section 7 where we also suggest further work.

2. Preliminaries

2.1. Notations. Notations in this paper are presented in Table 1.

Table 1: Notations.

Symbol	Meaning
λ	System security parameter
N	Maximal amount of recipients
$\mathcal{N}$	User index space $\mathcal{N} = \{1, 2, \ldots, N\}$
ID_i	The user's identity
S	Intended recipient set $\{ID_1, ID_2, \ldots, ID_n\}$
pp	Public parameter
MSK	Master secret key
K_{1i}, K_{2i}	Public keys of ID_i
S_{1i}, S_{2i}	Secret keys of ID_i
d	Length of vectors $\overrightarrow{x}$ and $\overrightarrow{y}$
n	Number of intended recipients in S
P	Bilinear pairing computation
E	Exponentiation computation in G
E_T	Exponentiation computation in G_T
M_T	Multiplication operation in G_T

Suppose that the sender distributes secret messages to a certain set of recipients. Let n denote the number of intended recipients in set S, and let vector length d denote the length of vectors $\overrightarrow{x}$ and $\overrightarrow{y}$.

2.2. Bilinear Groups. Let G and G_T be two cyclic groups with prime order q. $g \in G$ is the generator of group G, and e: $G \times G \longrightarrow G_T$. The symmetric bilinear group (G, G_T, q, e) has the following properties:

(1) The map e is bilinear: for all $a, b \in Z_q$ and $u, v \in G$, we have that $e(u^a, v^b) = e(u, v)^{ab}$

(2) The map e is nondegenerate: $e(g, g) \neq 1$

(3) There exists an efficient algorithm to compute $e(u, v)$, for any $u, v \in G$

We also briefly review the definition of vectors of group elements [34]. Let G be a cyclic group of prime order q, $g \in G$ be an element of group G, and vector $\overrightarrow{x} = \{x_1, ..., x_d\} \in Z_q^d$, where d is a natural number. Let $g^{\overrightarrow{x}}$ denote the vector of group elements $(g^{x_1}, \ldots, g^{x_d})$. For any scalar $t \in N$ and $\overrightarrow{x}$, $\overrightarrow{y}$, let

$$\left(g^{\overrightarrow{x}}\right)^t = g^{(t\overrightarrow{x})}, g^{\overrightarrow{x}} \cdot g^{\overrightarrow{y}} = g^{\overrightarrow{x}+\overrightarrow{y}},$$
$$e\left(g_1^{\overrightarrow{x}}, g_2^{\overrightarrow{y}}\right) = \prod_{i\in[d]} e\left(g_1^{x_i}, g_2^{y_i}\right) = e\left(g_1, g_2^{\langle \overrightarrow{x}, t\overrightarrow{y}\rangle}\right). \quad (1)$$

2.3. Security Assumption

Definition 1 (discrete logarithm (DL) problem). Given $g, h \in G$, the DL problem in G is to find x (if it exists) such that $g^x = h$. The advantage of any probabilistic polynomial-time (PPT) algorithm $\mathcal{B}$ in solving the DL problem in G is defined as $\mathrm{Adv}_{\mathcal{B}}^{\mathrm{DL}}$. The DL assumption is that, for any PPT algorithm $\mathcal{B}$, $\mathrm{Adv}_{\mathcal{B}}^{\mathrm{DL}}$ is negligible.

Definition 2 (computational bilinear Diffie–Hellman (CBDH) problem). Given $g, g^a, g^b, g^c \in G$ for unknown $a, b, c \in Z_p^*$, the CBDH problem in (G, G_T) is to compute $e(g, g)^{abc} \in G_T$. The advantage of any probabilistic polynomial-time (PPT) algorithm $\mathcal{B}$ in solving the CBDH problem in (G, G_T) is defined as $\mathrm{Adv}_{\mathcal{B}}^{CBDH} = Pr[\mathcal{B}(g, g^a, g^b, g^c) = e(g, g)^{abc} | a, b, c \in Z_p^*]$. The CBDH assumption is that, for any PPT algorithm $\mathcal{B}$, $\mathrm{Adv}_{\mathcal{B}}^{CBDH}$ is negligible.

2.4. IND-CCA Security of Inner Product Encryption. We review the IND-CCA security of inner product encryption [18]. The security against chosen-ciphertext attacks is defined via a game played by an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. An inner product encryption scheme is indistinguishable under adaptive chosen-ciphertext attacks if $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{IND-CCA}}$ is negligible for all adversary $\mathcal{A}$ winning Game 1 in polynomial time. The advantage of $\mathcal{A}$ winning Game 1 is $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{IND-CCA}} = |Pr[b = b' | < \overrightarrow{x}, \overrightarrow{y_0} > = < \overrightarrow{x}, \overrightarrow{y_1} >]|$. Game 1 is described as the following:

(1) The challenger $\mathcal{C}$ runs the Setup $(1^\lambda, S)$ to generate public parameters pp and master secret key MSK. Then, it sends pp to the adversary $\mathcal{A}$.

(2) The adversary $\mathcal{A}$ adaptively queries the key generation oracle for the functional secret key $sk_{\overrightarrow{x}}$ with the restriction that $\mathcal{A}$ can only query the secret key in that $\langle \overrightarrow{x}, \overrightarrow{y}_0 \rangle = \langle \overrightarrow{x}, \overrightarrow{y}_1 \rangle$, where $\overrightarrow{y_0}$ and $\overrightarrow{y_1}$ are the target plaintexts. $\mathcal{A}$ can also ask $\mathcal{C}$ to decrypt a ciphertext $ct_{\overrightarrow{y}}'$ to obtain $\langle \overrightarrow{x}, \overrightarrow{y} \rangle$ via the decryption oracle.

(3) The adversary $\mathcal{A}$ outputs two target plaintexts $\overrightarrow{y}_0$ and $\overrightarrow{y}_1$.

(4) The challenger $\mathcal{C}$ randomly selects a bit $b \in \{0, 1\}$ and generates a target ciphertext ct. Then, $\mathcal{C}$ passes ct to the adversary $\mathcal{A}$.

(5) The adversary $\mathcal{A}$ can continue to query the key generation oracle with the same restriction as before. $\mathcal{A}$ can also query the decryption oracle with the restriction that $\mathcal{A}$ cannot query the target ciphertext ct.

(6) The adversary $\mathcal{A}$ outputs a bit b', and $\mathcal{A}$ wins if $b = b'$.

2.5. Certificate-Based Broadcast Encryption. The certificate-based broadcast encryption scheme [12, 32] contains the following algorithms:

(i) Setup (1^λ): it inputs the security parameter λ and outputs the public parameters params and the master secret key msk.

(ii) KeyGen $(\mathrm{params}, \mathrm{ID}_i)$: it inputs the public parameters params and identity information ID_i. This algorithm outputs a key pair (pk_i, sk_i).

(iii) Certify $(\mathrm{params}, \mathrm{msk}, ID_i, pk_i)$: it inputs the public parameters params, master secret key msk, identity information ID_i, and public key pk_i. The algorithm outputs a certificate Cert_i.

(iv) Encrypt $(\text{params}, S, pk_i, M)$: it inputs the public parameters params, an intended recipient set S, a public key pk_i, and a message M. The algorithm outputs a ciphertext ct.

(v) Decrypt $(pp, sk_i, ct, ID_i, \text{Cert}_i)$: it inputs the public parameters pp, a secret key sk_i, a ciphertext ct, identity information ID_i, and a certificate Cert_i. The user in the intended recipient set outputs the message M.

2.6. Inner Product Encryption. We briefly recall the definition of the secret-key inner product encryption scheme [5]. It is shown as follows:

(i) Setup $(1^\lambda, S)$: it inputs a security parameter λ and a set S. Setup $(1^\lambda, S)$ outputs the public parameters pp and the master secret key msk.

(ii) KeyGen $(msk, \overrightarrow{x})$: it inputs the master secret key msk and a vector $\overrightarrow{x} \in Z_q^d$. KeyGen $(msk, \overrightarrow{x})$ outputs the functional secret key $sk_{\overrightarrow{x}}$.

(iii) Encrypt $(\text{msk}, \overrightarrow{y})$: it inputs the secret key $sk_{\overrightarrow{x}}$, a vector $\overrightarrow{y} \in Z_q^d$, and $\beta \in Z_q$. Encrypt $(\text{msk}, \overrightarrow{y})$ outputs a ciphertext $ct_{\overrightarrow{y}}$.

(iv) Decrypt $(pp, sk_{\overrightarrow{x}}, ct)$: it inputs the public parameters pp, a secret key $sk_{\overrightarrow{x}}$, and a ciphertext ct. Decrypt $(pp, sk_{\overrightarrow{x}}, ct)$ outputs a message $z \in Z_q$ or $\perp$.

As we can see from the above definition of the inner product encryption scheme, secret keys are associated with the vector $\overrightarrow{x}$, and the encrypted message is associated with the vector $\overrightarrow{y}$. Given a secret key for $\overrightarrow{x}$ and the ciphertext for $\overrightarrow{y}$, the recipient obtains the inner product value $\langle \overrightarrow{x}, \overrightarrow{y} \rangle$ via decryption. Especially, the above inner product encryption used in our scheme is different from the inner product predicate encryption scheme proposed by Okamoto and Takashima [35]. In the inner product predicate encryption scheme, a message m is encrypted with a tag $\overrightarrow{y}$, and the decryption key is associated with vector $\overrightarrow{x}$. The recipient can recover the message m only if $\langle \overrightarrow{x}, \overrightarrow{y} \rangle = 0$.

3. Formal Definition and Security Model

3.1. Formal Definition. The system model of our proposed scheme is shown in Figure 1. The formal definition of our scheme is shown as follows:

(i) Setup $(1^\lambda, d)$: it inputs a security parameter λ and vector length d. This algorithm outputs public parameters pp and master secret key MSK. Certificate authority (CA) runs this algorithm. It publishes pp and keeps MSK.

(ii) KeyGen $(pp, ID_i, \overrightarrow{x}_i)$: it inputs public parameters pp, a vector $\overrightarrow{x}_i$, and an identity ID_i. $\overrightarrow{x}_i$ is kept secretly, and it is not allowed to be known by others. It outputs secret keys $SK_i = \{SK_{1i}, SK_{2i}\}$ in addition to public keys $K_i = \{K_{1i}, K_{2i}\}$. This algorithm is executed by users.

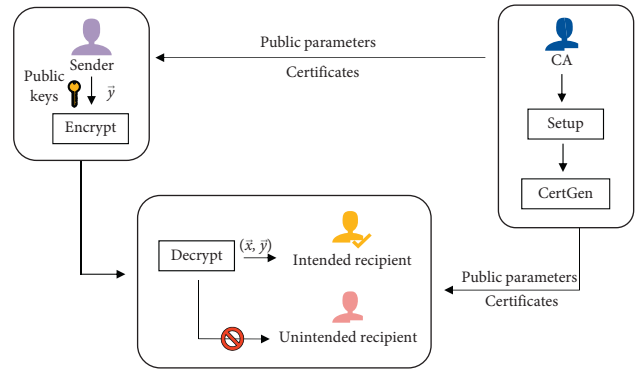

Figure 1: System model of the proposed scheme.

(iii) CertGen $(pp, \text{MSK}, ID_i, K_i)$: it inputs public parameters pp, a master secret key MSK, a user's identity ID_i, and public keys K_{1i} and K_{2i}. It outputs certificate Cert_i. This algorithm is executed by CA. Users obtain their certificates from the CA. The certificate is anonymous for the reason that no one is able to obtain the user identity by its certificate except the CA. The certificate plays a role as a portion of the user's keys. Though the CA generates the certificate for each user, it is not able to decrypt the ciphertext.

(iv) Encrypt $(pp, \overrightarrow{y}, S, K_i)$: it inputs public parameters pp, a vector $\overrightarrow{y} \in Z_q^d$ as the plaintext, the intended recipient set S, and public keys K_{1i} and K_{2i}. It outputs ciphertext CT. This algorithm is executed by the sender.

(v) Decrypt $(pp, CT, ID_i, SK_i, Cert_i)$: it inputs public parameters pp, a ciphertext CT, a user identity ID_i, a certificate $Cert_i$, and secret keys SK_{1i} and SK_{2i}. If ID_i is an intended recipient, it will obtain the corresponding inner product value of the related message. Otherwise, it outputs $\perp$.

3.2. Security Model. The security of our proposed scheme requires confidentiality and anonymity. As for the confidentiality, it means that, for an encrypted message which is associated with $\overrightarrow{y}$, only the intended recipients in S can obtain $\langle \overrightarrow{x}, \overrightarrow{y} \rangle$ through the decryption using their secret keys that are associated with $\overrightarrow{x}$. We give the definition for confidentiality of our proposed scheme via IND-CBIP-CCA Game 1 and IND-CBIP-CCA Game 2. As for the anonymity, all users, even users in S, are not able to recognize who is the intended recipient. In our scheme, the vector $\overrightarrow{x}_i$ is kept secretly by users, and it cannot be known by others though it may have implied relationship with user identity information; we do not consider $\overrightarrow{x}_i$ in anonymity games. On the contrary, the user identifier ID_i is public, so we considered the user identifier ID_i in anonymity games. We give the definition for anonymity of our proposed scheme via ANO-CBIP-CCA Game 1 and ANO-CBIP-CCA Game 2.

The security model of our proposed scheme contains two adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$. $\mathcal{A}_1$ is an uncertified user with no access to the master key. It can replace any user's public key

and query any user's secret key. $\mathcal{A}_1$ can also query any user's certificate except the target user's certificate. $\mathcal{A}_1$ can make the decryption query of any broadcast ciphertext except the target broadcast ciphertext. $\mathcal{A}_2$ is a malicious certifier that has a master key. It can generate any user's certificate. $\mathcal{A}_2$ is not able to replace any user's public key, but it can query any user's secret key except the target user. $\mathcal{A}_2$ can also perform the broadcast ciphertext's decryption query except the target broadcast ciphertext.

IND-CBIP-CCA Game 1 is played by a challenger $\mathcal{C}$ and an adversary $\mathcal{A}_1$.

Setup: $\mathcal{C}$ runs the Setup $(1^\lambda, d)$ algorithm, gives $\mathcal{A}_1$ the generated public parameters pp, and keeps the generated master secret key MSK with itself.

Phase 1: $\mathcal{A}_1$ adaptively launches the following queries to $\mathcal{C}$. $\mathcal{C}$ maintains a list $L_1 = (ID_i, K_i, SK_i, T_i)$ in order to answer queries. We denote public key $K_i = (K_{1i}, K_{2i})$ and secret key $SK_i = (SK_{1i}, SK_{2i})$. If $T_i = 0$, it represents that K_i has not been replaced by $\mathcal{A}_1$, while $T_i = 1$ means that $\mathcal{A}_1$ has made a replacement of K_i. L_1 was empty when it was initialized.

$\mathcal{O}^{\text{PublicKey}}(ID_i)$: public keys' query: on inputting ID_i, $\mathcal{C}$ retrieves L_1. If there is an item (ID_i, K_i, SK_i, T_i) related to ID_i in L_1, $\mathcal{C}$ returns corresponding K_i to $\mathcal{A}_1$. Otherwise, $\mathcal{C}$ runs KeyGen$(pp, ID_i, \overrightarrow{x}_i)$ and generates K_i and SK_i for ID_i. It adds the new item $(ID_i, K_i, SK_i, 0)$ to L_1 and returns K_i to $\mathcal{A}_1$.

$\mathcal{O}^{\text{PublicKeyReplace}}(ID_i, K_i')$: public keys' replacing query: on inputting ID_i and a public key K_i' randomly chosen by $\mathcal{A}_1$, $\mathcal{C}$ updates the item $(ID_i, K_i', \perp, 1)$ which is related to ID_i in L_1.

$\mathcal{O}^{\text{SecretKey}}(ID_i)$: secret keys' query: on inputting ID_i, $\mathcal{C}$ does the following things to answer the query. It searches the item (ID_i, K_i, SK_i, T_i) related to ID_i in L_1. If $T_i = 1$, $\mathcal{C}$ returns $\perp$. Otherwise, it returns SK_i to $\mathcal{A}_1$.

$\mathcal{O}^{\text{Certificate}}(ID_i)$: certificate query: on inputting ID_i, in order to make a response to the query, $\mathcal{C}$ searches the item (ID_i, K_i, SK_i, T_i) in L_1. Then, it executes CertGen$(pp, \text{MSK}, ID_i, K_i)$ and returns the generated $Cert_i$ to $\mathcal{A}_1$.

$\mathcal{O}^{\text{Decrypt}}(ID_i, CT)$: decrypt query: on inputting ID_i and ciphertext CT, $\mathcal{C}$ searches the item (ID_i, K_i, SK_i, T_i) in L_1. If $T_i = 1$, $\mathcal{A}_1$ has made replacement of K_i, and it should give corresponding SK_i of K_i. $\mathcal{C}$ executes CertGen$(pp, \text{MSK}, ID_i, K_i)$ and generates $Cert_i$ of ID_i. Then, $\mathcal{C}$ executes Decrypt$(pp, CT, ID_i, SK_i, \text{Cert}_i)$ to decrypt CT. It sends the decryption result to $\mathcal{A}_1$.

Challenge: $\mathcal{A}_1$ sends a challenge recipient set $S^* = \{ID_1, \ldots, ID_n\}$, two distinct messages (M_0, M_1) and $M_0 = \overrightarrow{y}_0$ and $M_1 = \overrightarrow{y}_1$, and then $\langle \overrightarrow{x}, \overrightarrow{y}_0 \rangle = \langle \overrightarrow{x}, \overrightarrow{y}_1 \rangle$. It sends (M_0, M_1) to $\mathcal{C}$ with the constraint that $\mathcal{A}_1$ neither queried $Cert_i$ of ID_i in S^* nor made replacement of K_i for ID_i in S^* in Phase 1. $\mathcal{C}$ selects a bit $\mu \in \{0, 1\}$ at random. Then, it executes Encrypt(pp, M_μ, S^*, K_i) and returns the generated challenge ciphertext CT^* to $\mathcal{A}_1$.

Phase 2: $\mathcal{A}_1$ issues a set of queries adaptively as in Phase 1. However, it is forbidden to query $Cert_i$ of ID_i in S^* or decryption of ID_i in S^*.

Guess: $\mathcal{A}_1$ outputs a guess $\mu \in \{0, 1\}$. It wins the game if $\mu' = \mu$. We define $\mathcal{A}_1$'s advantage in attacking the scheme to win IND-CBIP-CCA Game 1 as $\text{Adv}_{\mathcal{A}_1}^{\text{IND-CBIP-CCA}}(\lambda) = |Pr[\mu' = \mu] - 1/2|$.

Definition 3. We say that our proposed scheme is IND-CBIP-CCA secure if $\text{Adv}_{\mathcal{A}_1}^{\text{IND-CBIP-CCA}}(\lambda) < \varepsilon$ is satisfied for any PPT adversary $\mathcal{A}_1$.

IND-CBIP-CCA Game 2 is played by a challenger $\mathcal{C}$ and an adversary $\mathcal{A}_2$.

Setup: $\mathcal{C}$ runs the Setup $(1^\lambda, d)$ algorithm and gives $\mathcal{A}_2$ the generated public parameters pp and the generated master secret key MSK.

Phase 1: $\mathcal{A}_2$ adaptively launches the following queries to $\mathcal{C}$. $\mathcal{C}$ maintains list $L_2 = (ID_i, K_i, SK_i)$ for answering queries. We denote $K_i = (K_{1i}, K_{2i})$ and $SK_i = (SK_{1i}, SK_{2i})$. L_2 was empty when it was initialized.

$\mathcal{O}^{\text{PublicKey}}(ID_i)$:public keys' query: on inputting ID_i, $\mathcal{C}$ retrieves L_2. If there is an item (ID_i, K_i, SK_i) related to ID_i in L_2, $\mathcal{C}$ returns corresponding K_i to $\mathcal{A}_2$. Else, it runs KeyGen$(pp, ID_i, \overrightarrow{x_i})$ and generates K_i and SK_i for ID_i. Then, it adds the new item (ID_i, K_i, SK_i) to L_2 and returns K_i to $\mathcal{A}_2$.

$\mathcal{O}^{\text{SecretKey}}(ID_i)$: secret keys' query: on inputting ID_i, in order to make a response to the query, $\mathcal{C}$ searches item (ID_i, K_i, SK_i) related to ID_i in L_2 and returns SK_i to $\mathcal{A}_2$.

$\mathcal{O}^{\text{Decrypt}}(ID_i, CT)$: decrypt query: on inputting ID_i and ciphertext CT, $\mathcal{C}$ does the following things to answer the query. It searches the item (ID_i, K_i, SK_i) in L_2, executes CertGen$(pp, \text{MSK}, ID_i, K_i)$, and generates $Cert_i$ of ID_i. Then, $\mathcal{C}$ executes Decrypt$(pp, CT, ID_i, SK_i, \text{Cert}_i)$ to decrypt CT. It sends the decryption result to $\mathcal{A}_2$.

Challenge: $\mathcal{A}_2$ sends a challenge recipient set $S^* = \{ID_1, \ldots, ID_n\}$, two distinct messages (M_0, M_1) and $M_0 = \overrightarrow{y}_0$ and $M_1 = \overrightarrow{y}_1$, and then $\langle \overrightarrow{x}, \overrightarrow{y}_0 \rangle = \langle \overrightarrow{x}, \overrightarrow{y}_1 \rangle$. It sends (M_0, M_1) to $\mathcal{C}$ with the constraint that $\mathcal{A}_2$ has not queried SK_i of ID_i in S^* in Phase 1. $\mathcal{C}$ selects a bit $\mu \in \{0, 1\}$ at random. Then, it executes Encrypt(pp, M_μ, S^*, K_i) and returns the generated challenge ciphertext CT^* to $\mathcal{A}_2$.

Phase 2: $\mathcal{A}_2$ issues a set of queries adaptively as in Phase 1. However, it is forbidden to query SK_i of ID_i in S^* or decryption of ID_i in S^*.

Guess: $\mathcal{A}_2$ outputs a guess $\mu \in \{0, 1\}$. It wins the game if $\mu' = \mu$. We define $\mathcal{A}_2$'s advantage in attacking the scheme to win IND-CBIP-CCA Game 2 as $\text{Adv}_{\mathcal{A}_2}^{\text{IND-CBIP-CCA}}(\lambda) = |Pr[\mu' = \mu] - 1/2|$.

Definition 4. We say that our proposed scheme is IND-CBIP-CCA secure if $\mathrm{Adv}_{\mathscr{A}_2}^{\mathrm{IND-CBIP-CCA}}(\lambda) < \varepsilon$ is satisfied for any PPT adversary $\mathscr{A}_2$.

ANO-CBIP-CCA Game 1 is played by a challenger $\mathscr{C}$ and an adversary $\mathscr{A}_1$.

Setup: $\mathscr{C}$ runs the Setup $(1^\lambda, d)$ algorithm, gives $\mathscr{A}_1$ the generated public parameters pp, and keeps the generated master secret key MSK with itself.

Phase 1: $\mathscr{A}_1$ adaptively launches the following queries to $\mathscr{C}$. $\mathscr{C}$ maintains a list $L_1 = (ID_i, K_i, SK_i, T_i)$ in order to answer queries. We denote public key $K_i = (K_{1i}, K_{2i})$ and secret key $SK_i = (SK_{1i}, SK_{2i})$. If $T_i = 0$, it represents that K_i has not been replaced by $\mathscr{A}_1$, while $T_i = 1$ means that $\mathscr{A}_1$ has made replacement of K_i. L_1 was empty when it was initialized.

$\mathcal{O}^{\mathrm{PublicKey}}(ID_i)$: public keys' query: on inputting ID_i, $\mathscr{C}$ retrieves L_1. If there is an item (ID_i, K_i, SK_i, T_i) related to ID_i in L_1, $\mathscr{C}$ returns the corresponding public key K_i to $\mathscr{A}_1$. Otherwise, $\mathscr{C}$ runs $\mathrm{KeyGen}(pp, ID_i, \vec{x}_i)$ and generates K_i and SK_i for ID_i. It adds the new item $(ID_i, K_i, SK_i, 0)$ to L_1 and returns K_i to $\mathscr{A}_1$.

$\mathcal{O}^{\mathrm{PublicKeyReplace}}(ID_i, K_i')$: public keys' replacing query: on inputting ID_i and a public key K_i' randomly chosen by $\mathscr{A}_1$, $\mathscr{C}$ updates the item $(ID_i, K_i', \perp, 1)$ which is related to ID_i in L_1.

$\mathcal{O}^{\mathrm{SecretKey}}(ID_i)$: secret keys' query: on inputting ID_i, $\mathscr{C}$ does the following things to answer the query. It searches the item (ID_i, K_i, SK_i, T_i) related to ID_i in L_1. If $T_i = 1$, $\mathscr{C}$ returns $\perp$. Otherwise, it returns SK_i to $\mathscr{A}_1$.

$\mathcal{O}^{\mathrm{Certificate}}(ID_i)$: certificate query: on inputting ID_i, $\mathscr{C}$ searches the item (ID_i, K_i, SK_i, T_i) in L_1. Then, it executes $\mathrm{CertGen}(pp, \mathrm{MSK}, ID_i, K_i)$ and returns the generated certificate $Cert_i$ to $\mathscr{A}_1$.

$\mathcal{O}^{\mathrm{Decrypt}}(ID_i, CT)$: decrypt query: on inputting ID_i and ciphertext CT, $\mathscr{C}$ searches the item (ID_i, K_i, SK_i, T_i) in L_1. If $T_i = 1$, $\mathscr{A}_1$ has made replacement of K_i, and it should give corresponding SK_i of K_i to $\mathscr{C}$. $\mathscr{C}$ executes $\mathrm{CertGen}(pp, \mathrm{MSK}, ID_i, K_i)$ and generates $Cert_i$ of ID_i. Then, it runs $\mathrm{Decrypt}(pp, CT, ID_i, SK_i, \mathrm{Cert}_i)$ to decrypt CT and sends the decryption result to $\mathscr{A}_1$.

Challenge: $\mathscr{A}_1$ sends a challenge recipient set $S = \{ID_1, \ldots, ID_n\}$, two user identities (ID_0^*, ID_1^*), and a message $M = \vec{y}$ to $\mathscr{C}$ with the constraint that $\mathscr{A}_1$ neither queried $Cert_i$ of ID_i in $S \cup (ID_0^*, ID_1^*)$ nor replaced K_i of ID_i in $S \cup (ID_0^*, ID_1^*)$ in Phase 1. $\mathscr{C}$ selects a bit $\mu \in \{0, 1\}$ and set $S_\mu^* = ID_\mu^* \cup S$ at random. Then, it executes $\mathrm{Encrypt}(pp, M, S_\mu^*, K_i)$ and returns the generated challenge ciphertext CT^* to $\mathscr{A}_1$.

Phase 2: $\mathscr{A}_1$ issues a set of queries adaptively as in Phase 1 with the constraint that it is not able to query $Cert_i$ of ID_i in $S \cup (ID_0^*, ID_1^*)$ or decryption of ID_i in $S \cup (ID_0^*, ID_1^*)$.

Guess: $\mathscr{A}_1$ outputs a guess $\mu' \in \{0, 1\}$. It wins the game if $\mu' = \mu$. We define $\mathscr{A}_1$'s advantage in attacking the scheme to win ANO-CBIP-CCA Game 1 as $\mathrm{Adv}_{\mathscr{A}_1}^{\mathrm{ANO-CBIP-CCA}}(\lambda) = |Pr[\mu' = \mu] - 1/2|$.

Definition 5. We say that our proposed scheme is ANO-CBIP-CCA secure if $\mathrm{Adv}_{\mathscr{A}_1}^{\mathrm{ANO-CBIP-CCA}}(\lambda) < \varepsilon$ is satisfied for any PPT adversary $\mathscr{A}_1$.

ANO-CBIP-CCA Game 2 is played by a challenger $\mathscr{C}$ and an adversary $\mathscr{A}_2$.

Setup: $\mathscr{C}$ runs the Setup $(1^\lambda, d)$ algorithm and gives $\mathscr{A}_2$ the generated public parameters pp and the generated master secret key MSK.

Phase 1: $\mathscr{A}_2$ adaptively launches the following queries to $\mathscr{C}$. $\mathscr{C}$ maintains list $L_2 = (ID_i, K_i, SK_i)$ for answering queries. We denote $K_i = (K_{1i}, K_{2i})$ and $SK_i = (SK_{1i}, SK_{2i})$. L_2 was empty when it was initialized.

$\mathcal{O}^{\mathrm{PublicKey}}(ID_i)$: public keys' query: on inputting ID_i, $\mathscr{C}$ retrieves L_2. If there is an item (ID_i, K_i, SK_i) related to ID_i in L_2, $\mathscr{C}$ returns corresponding K_i to $\mathscr{A}_2$. Otherwise, $\mathscr{C}$ runs $\mathrm{KeyGen}(pp, ID_i, \vec{x}_i)$ and generates K_i and SK_i for ID_i. It adds the new item (ID_i, K_i, SK_i) to L_2 and returns K_i to $\mathscr{A}_2$.

$\mathcal{O}^{\mathrm{SecretKey}}(ID_i)$: secret keys' query: on inputting ID_i, $\mathscr{C}$ searches the item (ID_i, K_i, SK_i) related to ID_i in L_2 and returns SK_i to $\mathscr{A}_2$.

$\mathcal{O}^{\mathrm{Decrypt}}(ID_i, CT)$: decrypt query: on inputting ID_i and ciphertext CT, $\mathscr{C}$ searches the item (ID_i, K_i, SK_i) in L_2. It executes $\mathrm{CertGen}(pp, \mathrm{MSK}, ID_i, K_i)$ and generates $Cert_i$ for ID_i. Then, $\mathscr{C}$ executes $\mathrm{Decrypt}(pp, CT, ID_i, SK_i, Cert_i)$ to decrypt CT. It sends the decryption result to $\mathscr{A}_2$.

Challenge: $\mathscr{A}_2$ sends a challenge recipient set $S = \{ID_1, \ldots, ID_n\}$, two user identities (ID_0^*, ID_1^*), and a message $M = \vec{y}$ to $\mathscr{C}$ with the constraint that $\mathscr{A}_2$ has not queried SK_i of ID_i in $S \cup (ID_0^*, ID_1^*)$ in Phase 1. $\mathscr{C}$ selects a bit $\mu \in \{0, 1\}$ at random and set $S_\mu^* = ID_\mu^* \cup S$. Then, it executes $\mathrm{Encrypt}(pp, M, S_\mu^*, K_i)$ and returns the generated challenge ciphertext CT^* to $\mathscr{A}_2$.

Phase 2: $\mathscr{A}_2$ issues a set of queries adaptively as in Phase 1 with the constraint that it is not able to query SK_i of ID_i in $S \cup (ID_0^*, ID_1^*)$ or decryption of ID_i in $S \cup (ID_0^*, ID_1^*)$.

Guess: $\mathscr{A}_2$ outputs a guess $\mu' \in \{0, 1\}$. It wins the game if $\mu' = \mu$. We define $\mathscr{A}_2$'s advantage in attacking the scheme to win ANO-CBIP-CCA Game 2 as $\mathrm{Adv}_{\mathscr{A}_2}^{\mathrm{ANO-CBIP-CCA}}(\lambda) = |Pr[\mu' = \mu] - 1/2|$.

Definition 6. We say that our proposed scheme is ANO-CBIP-CCA secure if $\mathrm{Adv}_{\mathscr{A}_2}^{\mathrm{ANO-CBIP-CCA}}(\lambda) < \varepsilon$ is satisfied for any PPT adversary $\mathscr{A}_2$.

4. Our Certificate-Based Inner Product Broadcast Encryption Scheme

In this section, we present the concrete construction of our proposed scheme as follows.

Setup $(1^\lambda, d)$: taking the security parameter λ and vector length d as the input, the CA performs the following tasks:

(1) Generate symmetric cyclic bilinear groups G and G_T with order q. The large prime q is λ bits. g is a generator of group G e: $G \times G \longrightarrow G_T$ is a bilinear map, and $g_T = e(g, g)$.

(2) Choose $\gamma \in Z_q^*$ randomly. Calculate $g_1 = g^\gamma$.

(3) Select four cryptographic hash functions with forms as $H_1: = \{0,1\}^* \times G \times G \longrightarrow G$, $H_2: = \{0,1\}^* \times G \times G \times G \longrightarrow G$, $H_3: = G_T \times G_T \longrightarrow Z_p^*$, and $H_4: = G_T \longrightarrow Z_p^*$.

(4) Keep the MSK = γ secretly and publish the public parameters $pp = \{G, G_T, d, q, g, e, g_T, g_1, H_1, H_2, H_3, H_4\}$.

KeyGen $(pp, ID_i, \overrightarrow{x}_i)$: taking the public parameters pp, a vector $\overrightarrow{x}_i$, and an identity ID_i as the input, the user ID_i randomly chooses $\alpha_i \in Z_q$. It has $\overrightarrow{x}_i \in Z_q^d$. It generates secret keys $SK_i = \{SK_{1i}, SK_{2i}\}$ and public keys $K_i = \{K_{1i}, K_{2i}\}$ by the following steps:

(1) Calculate $SK_{1i} = \alpha_i$ and $SK_{2i} = \alpha_i \overrightarrow{x}_i$ as secret keys

(2) Compute $K_{1i} = g^{SK_{1i}}$ and $K_{2i} = g^{SK_{2i}}$ as public keys

CertGen $(pp, \text{MSK}, ID_i, K_i)$: taking public parameters pp, a master secret key MSK, a user's identity ID_i, and public keys K_{1i} and K_{2i} as the input, the CA computes $Q_i = H_1(ID_i, K_{1i}, K_{2i})$ and $\text{Cert}_i = Q_i^\gamma$. The user ID_i checks whether its Cert_i is valid. If eg,$\text{Cert}_i = \text{eg}_1$, Q_i, Cert_i is valid.

Encrypt $(pp, \overrightarrow{y}, S, K)$: taking the public parameters pp, a vector $\overrightarrow{y} \in Z_q^d$, the intended recipient set S, and public keys K_{1i} and K_{2i} as the input, the sender executes the algorithm to output a ciphertext CT. We suppose $S = \{\text{ID}_1, \text{ID}_2, ..., \text{ID}_n\}$.

First, the sender computes $Q_i = H_1(\text{ID}_i, K_{1i}, K_{2i})$ and $R_i = H_2(\text{ID}_i, K_{1i}, K_{2i}, g_1)$ for every intended recipient ID_i.

Next, the sender chooses $k \in G_T$ and $\beta \in Z_q^*$ at random. It selects $r_i \in Z_q^*$ at random and computes $\chi_i = eg, Q_i^{-r_i} \cdot e(K_{1i}, R_i)^{-r_i}$.

Then, the sender computes ciphertext CT as shown in equations:

$$C_{i\_0} = g^{r_i}, \tag{2}$$

$$C_{i\_1} = D_{i\_1} \cdot e(g_1, Q_i)^{-r_i} = e\left(K_{1i}, g^\beta\right) \cdot e(g_1, Q_i)^{-r_i}, \tag{3}$$

$$C_{i\_2} = D_{i\_2} \cdot e(K_{1i}, R_i)^{-r_i} = e\left(K_{2i}, g^{\beta \overrightarrow{y}}\right) \cdot e(K_{1i}, R_i)^{-r_i}, \tag{4}$$

$$C_{i\_3} = r_i \oplus H_3(D_{i\_1}, D_{i\_2}), \tag{5}$$

$$C_i = \left(H_4(\chi_i), C_{i\_0}, C_{i\_1}, C_{i\_2}, C_{i\_3}\right), \tag{6}$$

$$CT = (C_1, C_2, \ldots, C_n). \tag{7}$$

Decrypt $(pp, CT, ID_i, SK_i \text{Cert}_i)$: taking the public parameters pp, a ciphertext CT, a user's identity ID_i, a certificate $Cert_i$, and secret keys SK_{1i} and SK_{2i} as the input, the user performs the following steps.

First, the user calculates $\chi_i' = e(C_{i\_0}, Cert_i)^{-1} \cdot e(C_{i\_0}, R_i)^{-SK_{1i}}$.

Next, the user computes $H_4(\chi_i')$. If the user is not an intended recipient, it is not able to find the same value $H_4(\chi_i)$ in CT and is not able to determine the corresponding C_{i_1} and C_{i_2} of $H_4(\chi_i')$. Then, it outputs $\perp$. Otherwise, the user utilizes $H_4(\chi_i')$ to locate its associated C_i by relationships among $H_4(\chi_i')$, C_{i_1}, and C_{i_2} in C_i.

Then, the user computes $D_{i\_1} = C_{i\_1} \cdot e(C_{i\_0}, Cert_i)$ and $D_{i\_2} = C_{i\_2} \cdot e(C_{i\_0}, R_i)^{SK_{1i}}$.

Then, the user calculates C_{i_0}' as the following:

$$\begin{aligned} r_i' &= C_{i\_3}' \oplus H_3(D_{i\_1}, D_{i\_2}), \\ C_{i\_0}' &= g^{r_i'}. \end{aligned} \tag{8}$$

Finally, if $C_{i\_0}' = C_{i\_0}$, the user calculates $z \in T$ which satisfies $(D_{i\_1})^z = D_{i\_2}$. Let T be a polynomial-sized subset of Z_q. If there exists $z \in T$, the algorithm outputs z. Otherwise, it outputs $\perp$.

Correctness: our proposed scheme is said to satisfy the correct condition if the following equation holds:

$$\begin{aligned} \chi_i' &= e(C_{i\_0}, Cert_i)^{-1} \cdot e(C_{i\_0}, R_i)^{-SK_{1i}} \\ &= e(g^{r_i}, Q_i^\gamma)^{-1} \cdot e(g^{r_i}, R_i)^{-SK_{1i}} \\ &= e(g_1, Q_i)^{-r_i} \cdot e(K_{1i}, R_i)^{-r_i} \\ &= \chi_i. \end{aligned} \tag{9}$$

Meanwhile, it requires the plaintext vectors to satisfy $<\overrightarrow{x}_i, \overrightarrow{y}> \in T$, for polynomially sized T.

For any SK_{1i}, SK_{2i} and K_{1i}, K_{2i}, we have

$$\begin{aligned} D_{i\_1} &= C_{i\_1} \cdot e(C_{i\_0}, Cert_i) \\ &= e\left(K_{1i}, g^\beta\right) \cdot e(g_1, Q_i)^{-r_i} \cdot e(C_{i\_0}, Cert_i) \\ &= e\left(g^{\alpha_i}, g^\beta\right) \\ &= e(g, g)^{\alpha_i \beta}, \\ D_{i\_2} &= C_{i\_2} \cdot e(C_{i\_0}, R_i)^{SK_{1i}} \\ &= e\left(K_{2i}, g^{\beta \cdot \overrightarrow{y}}\right) \cdot e(K_{1i}, R_i)^{-r_i} \cdot e(C_{i\_0}, R_i)^{SK_{1i}} \\ &= e\left(g^{\alpha_i \overrightarrow{x_i}}, g^{\beta \overrightarrow{y}}\right) \\ &= e(g, g)^{\alpha_i \beta \cdot \overrightarrow{x}_i \overrightarrow{y}^T} \\ &= e(g, g)^{\alpha_i \beta \cdot \langle \overrightarrow{x}_i, \overrightarrow{y} \rangle}. \end{aligned} \tag{10}$$

If $\langle \overrightarrow{x}_i, \overrightarrow{y} \rangle \in T$, the decryption algorithm outputs inner product value $\langle \overrightarrow{x}_i, \overrightarrow{y} \rangle$ by the baby-step giant-step algorithm. It is efficient since $|T| = \text{poly}(\lambda)$.

5. Security Analysis

Now, we prove the confidentiality and anonymity of our scheme through the security models defined in Section 3. Our proof strategy draws inspiration from the CBBE scheme [32]. First, the confidentiality of our scheme will be proved through IND-CBIP-CCA Game 1 and IND-CBIP-CCA Game 2 defined in Section 3.

Theorem 1. *Suppose that hash functions $H_i(i = 1, 2, 3, 4)$ are random oracles and $\mathcal{A}_1$ is able to launch q_{cert} queries to $\mathcal{O}^{Certificate}$, q_{dec} queries to $\mathcal{O}^{Decrypt}$, and $q_{H_i}(i = 1, 2, 3, 4)$ queries to $H_i(i = 1, 2, 3, 4)$, respectively. It has advantage ε over our proposed scheme in IND-CBIP-CCA Game 1. Then, there exists an PPT algorithm $\mathcal{B}$ to solve the CBDH problem with the advantage at least $(n\varepsilon/q_{H_3}N)(1 - q_{dec}/2^{\lambda})(q_{H_1} - 1/q_{H_1})^{q_{cert}}$.*

Proof. Suppose that there exists an adversary $\mathcal{A}_1$ that can break the proposed scheme in IND-CBIP-CCA Game 1 with advantage ε. We build an algorithm $\mathcal{B}$ to solve the CBDH problem by running $\mathcal{A}_1$. Given as the input a problem instance (g, g^a, g^b, g^c), $\mathcal{B}$ needs to simulate a challenger $\mathcal{C}$ and all oracles. It works as follows.

Setup: $\mathcal{B}$ executes the Setup$(1^{\lambda}, d)$ algorithm and outputs (q, G, G_T, e). We note that g is the generator of G, and $g_1 = g^{\gamma}$. Then, $\mathcal{B}$ computes $g_T = e(g, g)$ and picks index $\ell \in \{1, 2, \ldots, q_{H_1}\}$ at random. $\mathcal{B}$ controls random oracles $H_i(i = 1, 2, 3, 4)$. It also publishes system public parameters $pp = \{G, G_T, d, q, g, e, g_T, g_1, H_1, H_2, H_3, H_4\}$ and keeps master secret key MSK $= \gamma$.

H_1 query: $\mathcal{A}_1$ makes the H_1 query adaptively. $\mathcal{B}$ responds to $\mathcal{A}_1$'s query on (ID_i, K_i) as shown below. $\mathcal{B}$ maintains the list $L_{H_1} = (ID_i, K_i, Q_i, Cert_i)$. If the query ID_i appears on L_{H_1} in an item $(ID_i, K_i, Q_i, Cert_i)$, it returns corresponding Q_i to $\mathcal{A}_1$. Otherwise, if the query is on ID_ℓ, $\mathcal{B}$ sets $Q_\ell = g^b$, returns Q_ℓ to $\mathcal{A}_1$, and adds $(ID_\ell, K_\ell, Q_\ell, \perp)$ to L_{H_1}. Note that $K_\ell = \{K_{1\ell}, K_{2\ell}\}$. Otherwise, $\mathcal{B}$ does the following things:

(1) Select $t_i \in Z_p^*$ at random. Let $Q_i = g^{t_i}$ and $\text{Cert}_i = (g^a)^{t_i}$.

(2) Add the tuple $(ID_i, K_i, Q_i, \text{Cert}_i)$ to L_{H_1} and respond with Q_i to $\mathcal{A}_1$.

H_2 query: $\mathcal{A}_1$ makes H_2 query adaptively. $\mathcal{B}$ makes a response to $\mathcal{A}_1$'s query on (ID_i, K_i, g_1) as shown below. $\mathcal{B}$ maintains the list $L_{H_2} = (ID_i, K_i, \nu_i, R_i)$. If the query (ID_i, K_i) appears on L_{H_2} in an item (ID_i, K_i, ν_i, R_i), it returns corresponding R_i to $\mathcal{A}_1$. Else, $\mathcal{B}$ picks $\nu_i \in Z_p^*$ at random and calculates $R_i = g^{\nu_i}$, and then $\mathcal{B}$ adds (ID_i, K_i, ν_i, R_i) to L_{H_2} and responds to $\mathcal{A}_1$ with R_i.

H_3 query: $\mathcal{A}_1$ makes the H_3 query adaptively. $\mathcal{B}$ responds to $\mathcal{A}_1$'s query on (D_{i_1}, D_{i_2}) as shown below. $\mathcal{B}$ maintains the list $L_{H_3} = \{(D_{i\_1}, D_{i\_2}, h_3)\}$. If the query (D_{i_1}, D_{i_2}) appears on L_{H_3} in an item (D_{i_1}, D_{i_2}, h_3), $\mathcal{B}$ returns corresponding h_3 to $\mathcal{A}_1$. Otherwise, $\mathcal{B}$ picks $h_3 \in Z_p^*$ at random, adds (D_{i_1}, D_{i_2}, h_3) to L_{H_3}, and responds to $\mathcal{A}_1$ with h_3.

H_4 query: $\mathcal{A}_1$ makes the H_4 query adaptively. $\mathcal{B}$ responds to $\mathcal{A}_1$'s query on ID_i as shown below. $\mathcal{B}$ maintains the list $L_{H_4} = \{(ID_i, h_4)\}$. If the query ID_i appears on L_{H_4} in an item (ID_i, h_4), $\mathcal{B}$ returns corresponding h_4 to $\mathcal{A}_1$. Otherwise, $\mathcal{B}$ picks $h_4 \in Z_p^*$ at random, adds (ID_i, h_4) to L_{H_4}, and returns h_4 to $\mathcal{A}_1$.

$\mathcal{O}^{\text{PublicKey}}$ query: $\mathcal{A}_1$ makes the $\mathcal{O}^{\text{PublicKey}}$ query adaptively. $\mathcal{B}$ responds to $\mathcal{A}_1$'s query on ID_i as shown below. If the query ID_i is already on L_1 in an item (ID_i, K_i, SK_i, T_i), $\mathcal{B}$ returns corresponding K_i to $\mathcal{A}_1$. Otherwise, $\mathcal{B}$ randomly picks $\alpha_i' \in Z_p$ and $\overrightarrow{x}_i' \in Z_q^d$, and then it computes $SK_{1i} = \alpha_i'$ and $SK_{2i} = \alpha_i \overrightarrow{x}_i'$ as secret keys. It computes $K_{1i} = g^{SK_{1i}}$ and $K_{2i} = g^{SK_{2i}}$ as public keys, and then it adds (ID_i, K_i, SK_i, T_i) to L_1 and responds to $\mathcal{A}_1$ with K_i.

$\mathcal{O}^{\text{PublicKeyReplace}}$ query: $\mathcal{A}_1$ makes the $\mathcal{O}^{\text{PublicKeyReplace}}$ query adaptively. On receiving the query on (ID_i, K_i, SK_i, T_i), $\mathcal{B}$ retrieves items related to ID_i in L_1 and updates the item (ID_i, K_i, SK_i, T_i) to $(ID_i, K_i', \perp, 1)$.

$\mathcal{O}^{\text{SecretKey}}$ query: $\mathcal{A}_1$ makes the $\mathcal{O}^{\text{SecretKey}}$ query adaptively. On receiving the query on ID_i, $\mathcal{B}$ searches the entry (ID_i, K_i, SK_i, T_i) of ID_i. If $T_i = 0$, $\mathcal{B}$ returns SK_i to $\mathcal{A}_1$. Otherwise, $\mathcal{B}$ aborts.

$\mathcal{O}^{\text{Certificate}}$ query: $\mathcal{A}_1$ makes the $\mathcal{O}^{\text{Certificate}}$ query adaptively. On receiving the query on ID_i, if $ID_i = ID_\ell$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ searches the item $(ID_i, K_i, Q_i, \text{Cert}_i)$ of ID_i and responds to $\mathcal{A}_1$ with $Cert_i$.

$\mathcal{O}^{\text{Decrypt}}$ query: $\mathcal{A}_1$ makes the $\mathcal{O}^{\text{Decrypt}}$ query adaptively by submitting CT and ID_i to $\mathcal{B}$. We note that $CT = (C_1, C_2, \ldots, C_n)$ and $n \le N$. $\mathcal{B}$ responds to the query from $\mathcal{A}_1$ on (ID_i, CT) as shown in the following:

(1) $\mathcal{B}$ searches item (ID_i, K_i, SK_i, T_i) of list L_1. If the user of ID_i is not an intended recipient, it rejects the query.

(2) If ID_ℓ is in the intended recipient set and $T_i = 0$, $\mathcal{B}$ searches the list L_{H_3} to find the entry (D_{i_1}, D_{i_2}, h_3) that satisfies $C_{i\_0} = g^{r_i}$ and $C_{i\_3} = r_i \oplus H_3(D_{i\_1}, D_{i\_2})$. If there is no item that satisfies the condition, $\mathcal{B}$ discards CT and aborts. Else, $\mathcal{B}$ responds to $\mathcal{A}_1$ with D_{i_1}, D_{i_2}.

(3) If ID_ℓ is not in the intended recipient set and $T_i = 1$, $\mathcal{A}_1$ should give SK_i corresponding to K_i of ID_i. Then, $\mathcal{B}$ checks whether $K_{1i} = g^{SK_{1i}}$ and $K_{2i} = g^{SK_{2i}}$ hold. If not so, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ searches L_{H_3} to find the entry (D_{i_1}, D_{i_2}, h_3) that satisfies $C_{i\_0} = g^{r_i}$ and $C_{i\_3} = r_i \oplus H_3(D_{i\_1}, D_{i\_2})$. If there is no item that satisfies the condition, $\mathcal{B}$ rejects the query. Otherwise, it returns D_{i_1}, D_{i_2} to the adversary $\mathcal{A}_1$.

(4) Otherwise, $\mathcal{B}$ obtains SK_i and $Cert_i$ which are related to ID_i, and then $\mathcal{B}$ executes Decrypt$(pp, CT, ID_i, SK_i, Cert_i)$ and responds to $\mathcal{A}_1$ with the result.

Phase 1: during this phase, $\mathcal{B}$ issues the above queries launched by $\mathcal{A}_1$ adaptively. For responding to the queries, $\mathcal{B}$ maintains a list $L_1 = \{ID_i, K_i, SK_i, T_i\}$. This list was initially empty. $T_i = 0$ represents that K_i has not been replaced by $\mathcal{A}_1$. Otherwise, $T_i = 1$ means that $\mathcal{A}_1$ has made replacement of K_i.

Challenge: $\mathcal{A}_1$ submits the intended recipient set $S^* = \{ID_1, ID_2, ID_3, \ldots, ID_n\}$, $(n \le N)$, two distinct messages (M_0, M_1) and $M_0 = \overrightarrow{y}_0$ and $M_1 = \overrightarrow{y}_1$, and then $\langle \overrightarrow{x}, \overrightarrow{y}_0 \rangle = \langle \overrightarrow{x}, \overrightarrow{y}_1 \rangle$. It sends (M_0, M_1) to the challenger $\mathcal{C}$, with the requirement that, in Phase 1, it neither obtained certificates of users in S^* nor made replacement of K_i for ID_i in S^*. Then, $\mathcal{B}$ randomly selects a value M_μ, $\mu \in \{0, 1\}$. If ID_ℓ is not in S^*, $\mathcal{B}$ aborts. Else, $\mathcal{B}$ sets $C_{i\_0}^* = g^c$ and chooses $H_4(\chi_i)^* \in Z_p^*$, $C_{i\_1}^* \in G_T$, $C_{i\_2}^* \in G_T$, and $C_{i\_3}^* \in G_T$ at

random. Then, the challenge broadcast ciphertext $CT^* = (C_1^*, C_2^*, \ldots, C_n^*)$ is returned to $\mathcal{A}_1$.

Phase 2: $\mathcal{A}_1$ issues a serious of queries adaptively. However, it cannot issue queries for certificates or decryption of ID_i in S^*.

Guess: $\mathcal{A}_1$ outputs a guess $\mu' \in \{0,1\}$ for μ. It wins the game if $\mu' = \mu$. For ID_ℓ, the description of CT^* is shown as follows. To produce the result, $\mathcal{B}$ should calculate D_{i_1} and D_{i_2} correctly. $\mathcal{B}$ chooses an item from L_{H_3} at random and searches v_ℓ from the item $(ID_\ell, K_\ell, v_\ell, R_\ell)$. To solve the CBDH problem, $\mathcal{B}$ computes $T_s = D_{i\_1} \cdot D_{i\_2} / \{C_{\ell\_1}^* \cdot C_{\ell\_2}^* \cdot e(K_\ell, C_{\ell\_0}^*)^{v_\ell}\}$ as the solution.

If $D_{i\_1} = C_{1\ell\_1}^* \cdot e(C_{\ell\_0}^*, Cert_\ell)$ and $D_{i\_2} = C_{\ell\_2}^* \cdot e(C_{\ell\_0}^*, R_\ell)^{SK_{1\ell}}$, then $C_{\ell\_1}^* = D_{i\_1} / e(C_{\ell\_0}^*, Cert_\ell)$ and $C_{\ell\_2}^* = D_{i\_2} / e(C_{\ell\_0}^*, R_\ell)^{SK_{1\ell}}$. $\mathcal{B}$ can extract the solution $T_s = D_{i\_1} \cdot D_{i\_2} / C_{\ell\_1}^* \cdot C_{\ell\_2}^* \cdot e(K_\ell, C_{\ell\_0}^*)^{v_\ell} = e(g,g)^{abc}$.

Analysis: then, we analyze the probability that the given CBDH problem can be solved by the challenger $\mathcal{C}$.

If $\mathcal{B}$ does not abort during the game, then $\mathcal{A}_1$'s view is identical to its view in the real scheme. Furthermore, we have $|Pr[\mu' = \mu] - 1/2| \geq \epsilon$. The game may be aborted before it finishes. Let Abort denote the game is aborted before it finishes. Then, event Abort occurs under any of the following occasions. (1)Ab_1: ID_ℓ is not in S^* during the Challenge phase. We have $Pr[Ab_1] = N - n/N$. (2)Ab_2: $\mathcal{B}$ aborts in the period that CT is given to $\mathcal{O}^{\mathrm{Decrypt}}$. We have $Pr[Ab_2] = q_{\mathrm{dec}}/2^\lambda$. (3)$\mathrm{Ab}_3$: $\mathcal{A}_1$ issues $\mathcal{O}^{\mathrm{Certificate}}$ query on $Pr[Ab_2] = q_{\mathrm{dec}}/2^\lambda$. We have $Pr[Ab_3] = (q_{H_1} - 1/q_{H_1})^{q_{\mathrm{cert}}}$. (4)$\mathrm{Ab}_4$: $\mathcal{A}_1$ issues $\mathcal{O}^{\mathrm{Secret}}$ query on ID_i, and $\mathcal{A}_1$ has made replacement of K_i for ID_i. If Ab_2 occurs, then Ab_4 also occurs. So, $Pr[\mathrm{Abort}] = Pr[Ab_1 \wedge Ab_2 \wedge Ab_3 \wedge Ab_4] \geq n/N 1 - (q_{\mathrm{dec}}/2^\lambda)(q_{H_1} - 1/q_{H_1})^{q_{\mathrm{cert}}}$.

Let Oca denote H_3Query|Abort. So, we have $Pr[\mu' = \mu|\mathrm{Oca}] = 1/2$ and $Pr[\mu' = \mu] \leq 1/2(Pr[Oca] + 1)$. By the definition of the probability for $\mathcal{A}_1$ in IND-CBIP-CCA Game 1, we have $\varepsilon \leq 2|Pr[\mu' = \mu] - 1/2| \leq Pr[\mathrm{Oca}] \leq Pr[H_3\mathrm{Query}]/Pr[\mathrm{Abort}]$. So, we have $Pr[H_3\mathrm{Query}] \geq \varepsilon Pr[\mathrm{Abort}] \geq n\varepsilon/N 1 - (q_{\mathrm{dec}}/2^\lambda)(q_{H_1} - 1/q_{H_1})^{q_{\mathrm{cert}}}$. Finally, $\mathcal{B}$ selects the correct item from L_{H_3} with probability $1/q_{H_3}$. Consequently, $\mathcal{B}$'s advantage is at least $n\varepsilon/q_{H_3} N(1 - q_{\mathrm{dec}}/2^\lambda)(q_{H_1} - 1/q_{H_1})^{q_{\mathrm{cert}}}$ as required. □

Theorem 2. *Suppose that hash functions $H_i (i = 1,2,3,4)$ are random oracles and $\mathcal{A}_1$ is able to launch q_{pubkey} queries to $\mathcal{O}^{PublicKey}$, q_{seckey} queries to $\mathcal{O}^{SecretKey}$, q_{dec} queries to $\mathcal{O}^{Decrypt}$, and $q_{H_i} (i = 1,2,3,4)$ to functions $H_i (i = 1,2,3,4)$, respectively. It has advantage ε over our proposed scheme in IND-CBIP-CCA Game 2. Then, there exists a PPT algorithm $\mathcal{B}$ to solve the CBDH problem with the advantage at least $n\varepsilon/q_{H_3} N(1 - q_{dec}/2^\lambda)(q_{pubkey} - 1/q_{pubkey})^{q_{seckey}}$.*

Proof. Suppose that there exists an adversary $\mathcal{A}_2$ that can break the proposed scheme in IND-CBIP-CCA Game 2 with advantage ε. We build an algorithm $\mathcal{B}$ to solve the CBDH problem by running $\mathcal{A}_2$. Given as the input a problem instance (g, g^a, g^b, g^c), $\mathcal{B}$ needs to simulate a challenger $\mathcal{C}$ and all oracles. It works as follows.

Setup: $\mathcal{B}$ executes the Setup$(1^\lambda, d)$ algorithm and outputs (q, G, G_T, e). We note that g is the generator of G and $g_1 = g^\gamma$. Then, $\mathcal{B}$ computes $g_T = e(g,g)$ and picks index $\ell \in \{1,2,\ldots,q\}$ at random. $\mathcal{B}$ controls random oracles $H_i (i = 1,2,3,4)$. It also publishes system public parameters $pp = \{G, G_T, d, q, g, e, g_T, g_1, H_1, H_2, H_3, H_4\}$ and gives master secret key MSK $= \gamma$ to $\mathcal{A}_2$. Random oracles $H_i (i = 1,2,3,4)$ are controlled by $\mathcal{B}$.

H_1 query: $\mathcal{A}_2$ makes the H_1 query adaptively. $\mathcal{B}$ makes a response to $\mathcal{A}_2$'s query on (ID_i, K_i) as shown below. It maintains the list $L_{H_1} = \{(ID_i, K_i, Q_i)\}$. If the query ID_i appears on L_{H_1} in an item (ID_i, K_i, Q_i), it returns corresponding Q_i to $\mathcal{A}_2$. Else, $\mathcal{B}$ chooses $Q_i \in G$ at random. Then, it adds (ID_i, K_i, Q_i) to L_{H_1} and responds with Q_i to $\mathcal{A}_2$.

H_2 query: $\mathcal{A}_2$ makes the H_2 query adaptively. $\mathcal{B}$ makes a response to $\mathcal{A}_2$'s query on (ID_i, K_i, g_1) as shown below. $\mathcal{B}$ maintains the list $L_{H_2} = (ID_i, K_i, R_i)$. If the query (ID_i, K_i) appears on L_{H_2} in an item (ID_i, K_i, R_i), it returns corresponding R_i to $\mathcal{A}_2$. Else, if $ID_i = ID_\ell$, it sets $R_\ell = g^a$, returns R_ℓ to $\mathcal{A}_2$, and adds (ID_i, K_i, R_i) to L_{H_2}. Note that $K_\ell = \{K_{1\ell}, K_{2\ell}\}$. Otherwise, $\mathcal{B}$ randomly selects $R_i \in G$. It adds (ID_i, K_i, R_i) to the list L_{H_2} and responds with R_i to $\mathcal{A}_2$.

H_3 query: $\mathcal{A}_2$ makes the H_3 query adaptively. $\mathcal{B}$ responds to $\mathcal{A}_2$'s query on (D_{i_1}, D_{i_2}) as shown below. $\mathcal{B}$ maintains the list $L_{H_3} = \{(D_{i\_1}, D_{i\_2}, h_3)\}$. If the query (D_{i_1}, D_{i_2}) appears on L_{H_3} in an item (D_{i_1}, D_{i_2}, h_3), $\mathcal{B}$ returns corresponding h_3 to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ picks $h_3 \in Z_p^*$ at random, adds (D_{i_1}, D_{i_2}, h_3) to L_{H_3}, and responds to $\mathcal{A}_2$ with h_3.

H_4 query: $\mathcal{A}_2$ makes the H_4 query adaptively. $\mathcal{B}$ makes a response to $\mathcal{A}_2$'s query on ID_i as shown below. $\mathcal{B}$ has the list $L_{H_4} = \{(ID_i, h_4)\}$. If the query ID_i appears on L_{H_4} in an item (ID_i, h_4), $\mathcal{B}$ returns corresponding h_4 to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ picks $h_4 \in Z_p^*$ at random, adds (ID_i, h_4) to L_{H_4}, and returns h_4 to $\mathcal{A}_2$.

$\mathcal{O}^{PublicKey}$ query: $\mathcal{A}_2$ makes the $\mathcal{O}^{PublicKey}$ query adaptively. $\mathcal{B}$ makes a response to $\mathcal{A}_2$'s query on ID_i as shown below. If the query ID_i appears on L_{H_2} in an item (ID_i, K_i, R_i), $\mathcal{B}$ returns related K_i to $\mathcal{A}_2$. Otherwise, if the query is on ID_ℓ, $\mathcal{B}$ randomly selects $\vec{x}_i' \in Z_q^d$. Then, it returns $K_{1\ell} = g^b$ and $K_{2\ell} = g^{b \cdot \vec{x}_i'}$ to $\mathcal{A}_2$ and adds $(ID_\ell, K_\ell, \perp)$ to L_2, while $K_\ell = (K_{1\ell}, K_{2\ell})$. The secret key $SK_{1\ell} = b$ is unknown to $\mathcal{B}$. Else, $\mathcal{B}$ randomly picks $\alpha_i' \in Z_p$ and $\vec{x}_i' \in Z_q^d$, and then $\mathcal{B}$ computes $SK_{1i} = \alpha_i'$ and $SK_{2i} = \alpha_i \vec{x}_i'$ as secret keys. It computes $K_{1i} = g^{SK_{1i}}$ and $K_{2i} = g^{SK_{2i}}$ as public keys, and then it adds (ID_i, K_i, SK_i) to L_2 and returns K_i to $\mathcal{A}_2$.

$\mathcal{O}^{\mathrm{SecretKey}}$ query: $\mathcal{A}_2$ makes the $\mathcal{O}^{\mathrm{SecretKey}}$ query adaptively. On receiving the query on ID_i, if $ID_i = ID_\ell$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ searches the entry (ID_i, K_i, SK_i) of ID_i in L_2 and returns SK_i to $\mathcal{A}_2$.

$\mathcal{O}^{\mathrm{Decrypt}}$ query: $\mathcal{A}_2$ makes the $\mathcal{O}^{\mathrm{Decrypt}}$ query adaptively by submitting CT and ID_i to $\mathcal{B}$. We note that $CT = (C_1, C_2, \ldots, C_n)$ and $n \leq N$. $\mathcal{B}$ makes a response to the query from $\mathcal{A}_2$ on (ID_i, CT) as shown in the following:

(1) $\mathcal{B}$ searches item (ID_i, K_i, SK_i) of list L_2. If the user of ID_i is not an intended recipient, it rejects the query.

(2) If ID_ℓ is in the intended recipient set, $\mathcal{B}$ searches L_{H_3} to find the entry (D_{i_1}, D_{i_2}, h_3) that satisfies $C_{i\_0} = g^{r_i}$ and $C_{i\_3} = r_i \oplus H_3(D_{i\_1}, D_{i\_2})$. If there is no item

that satisfies the condition, $\mathcal{B}$ rejects the query. Otherwise, it returns D_{i_1} and D_{i_2} to the adversary $\mathcal{A}_2$.

(3) Otherwise, $\mathcal{B}$ obtains SK_i and $Cert_i$ which are related to ID_i, and then $\mathcal{B}$ executes Decrypt$(pp, CT, ID_i, SK_i, Cert_i)$ and responds to $\mathcal{A}_2$ with the result.

Phase 1: during this phase, $\mathcal{B}$ issues the above queries launched by $\mathcal{A}_2$ adaptively. For responding to the queries, $\mathcal{B}$ maintains a list $L_2 = \{ID_i, K_i, SK_i\}$. This list was initially empty.

Challenge: when $\mathcal{A}_2$ decides that Phase 1 is over, it submits the intended recipient set $S^* = \{ID_1, ID_2, ID_3, \ldots, ID_n\}$, $(n \le N)$, two distinct messages (M_0, M_1) and $M_0 = \overrightarrow{y}_0$ and $M_1 = \overrightarrow{y}_1$, and then $\langle \overrightarrow{x}, \overrightarrow{y}_0 \rangle = \langle \overrightarrow{x}, \overrightarrow{y}_1 \rangle$. It sends (M_0, M_1) to the challenger $\mathcal{C}$, with the requirement that, in Phase 1, it has not obtained SK_i of ID_i in S^*. Then, $\mathcal{B}$ selects a random value $M_\mu, \mu \in \{0, 1\}$. If ID_ℓ is not in S^*, $\mathcal{B}$ aborts. Else, it sets $C^*_{i\_0} = g^c$ and chooses $H_4(\chi_i)^* \in Z^*_p$, $C^*_{i\_1} \in G_T$, $C^*_{i\_2} \in G_T$, and $C^*_{i\_3} \in G_T$ at random. Then, the challenge broadcast ciphertext $CT^* = (C^*_1, C^*_2, \ldots, C^*_n)$ is returned to the adversary $\mathcal{A}_2$.

Phase 2: $\mathcal{A}_2$ issues a set of queries adaptively. However, it cannot issue queries for SK_i of ID_i in S^* or decryption of ID_i in S^*.

Guess: $\mathcal{A}_2$ outputs a guess $\mu' \in \{0, 1\}$ for μ. It wins the game if $\mu' = \mu$. For ID_ℓ, the description of CT^* is shown as follows. To produce the result, $\mathcal{B}$ should calculate D_{i_1} and D_{i_2} correctly. $\mathcal{B}$ chooses an item from L_{H_3} at random and searches ν_ℓ from the item $(ID_\ell, K_\ell, \nu_\ell, R_\ell)$. To solve the CBDH problem, $\mathcal{B}$ computes $T = D_{i\_1} \cdot D_{i\_2} / \{C^*_{\ell\_1} \cdot C^*_{\ell\_2} \cdot e(K_\ell, C^*_{\ell\_0})^{\nu_\ell}\}$ as the solution.

If $D_{i\_1} = C^*_{1\ell\_1} \cdot e(C^*_{\ell\_0}, Cert_\ell)$ and $D_{i\_2} = C^*_{\ell\_2} \cdot e(C^*_{\ell\_0}, R_\ell)^{SK_{1\ell}}$, then $C^*_{\ell\_1} = D_{i\_1} / e(C^*_{\ell\_0}, Cert_\ell)$ and $C^*_{\ell\_2} = D_{i\_2} / e(C^*_{\ell\_0}, R_\ell)^{SK_{1\ell}}$. $\mathcal{B}$ can extract the solution $T = D_{i\_1} \cdot D_{i\_2} / C^*_{\ell\_1} \cdot C^*_{\ell\_2} \cdot e((C^*_{\ell\_0})^\alpha, Q_\ell) = e(g, g)^{abc}$.

Analysis: then, we analyze the probability that the given CBDH problem can be solved by the challenger $\mathcal{C}$.

If $\mathcal{B}$ does not abort during the game, then $\mathcal{A}_2$'s view is identical to its view in the real scheme. Furthermore, we have $|Pr[\mu' = \mu] - 1/2| \ge \epsilon$. The game may be aborted before it finishes. Let Abort denote the game is aborted before it finishes. Then, event Abort occurs under any of the following occasions. (1)Ab_1: the adversary $\mathcal{A}_2$ queries the oracle $\mathcal{O}^{\text{SecretKey}}$ on the user ID_ℓ. We have $Pr[Ab_1] = (q_{\text{pubkey}} - 1/q_{\text{pubkey}})^{q_{\text{seckey}}}$. (2)$Ab_2$: $\mathcal{B}$ aborts in the period that CT is given to $\mathcal{O}^{\text{Decrypt}}$. We have $Pr[Ab_2] = q_{\text{dec}}/2^\lambda$. (3)$Ab_3$: ID_ℓ is not in S^* during the Challenge phase. We have $Pr[Ab_3] = N - n/N$. So, we have that $Pr[\text{Abort}] = Pr[Ab_1 \wedge Ab_2 \wedge Ab_3] \ge n/N(1 - q_{\text{dec}}/2^\lambda)(q_{\text{pubkey}} - 1/q_{\text{pubkey}})^{q_{\text{seckey}}}$.

Let Oca denote H_3Query|Abort. So, we have $Pr[\mu' = \mu|\text{Oca}] = 1/2$ and $Pr[\mu' = \mu] = 1/2(Pr[\text{Oca}] + 1)$. By the definition of the advantage for $\mathcal{A}_2$ in IND-CBIP-CCA Game 2, we have $\varepsilon \le 2|Pr[\mu' = \mu] - 1/2| \le Pr[\text{Oca}] \le Pr[H_3 Query]/Pr[\text{Abort}]$. So, we have $Pr[H_3\text{Query}] \ge \varepsilon Pr[\text{Abort}] \ge n\varepsilon/N(1 - q_{\text{dec}}/2^\lambda)(q_{\text{pubkey}} - 1/q_{\text{pubkey}})^{q_{\text{seckey}}}$. Finally, $\mathcal{B}$ selects the correct item from L_{H_3} with probability $1/q_{H_3}$. Consequently, $\mathcal{B}$'s advantage is at least $n\varepsilon/q_{H_3}N(1 - q_{\text{dec}}/2^\lambda)(q_{\text{pubkey}} - 1/q_{\text{pubkey}})^{q_{\text{seckey}}}$ as required. □

Next, the anonymity of our scheme will be proved through ANO-CBIP-CCA Game 1 and ANO-CBIP-CCA Game 2 defined in Section 3.

Theorem 3. *Suppose that hash functions $H_i(i = 1, 2, 3, 4)$ are random oracles and $\mathcal{A}_1$ is able to launch q_{cert} queries to $\mathcal{O}^{Certificate}$ and $q_{H_i}(i = 1, 2, 3, 4)$ queries to functions $H_i(i = 1, 2, 3, 4)$, respectively. It has advantage ε over our proposed scheme in ANO-CBIP-CCA Game 1. Then, there exists aPPT algorithm $\mathcal{B}$ to solve the CBDH problem with the advantage at least $n\varepsilon/q_{H_3}N(1 - q_{dec}/2^\lambda)(q_{H_1} - 1/q_{H_1})^{q_{cert}}$.*

Proof. Suppose that there exists an adversary $\mathcal{A}_1$ that can break the proposed scheme in ANO-CBIP-CCA Game 1 with advantage ε. We build an algorithm $\mathcal{B}$ to solve the CBDH problem by running $\mathcal{A}_1$. Given as the input a problem instance (g, g^a, g^b, g^c), $\mathcal{B}$ needs to simulate a challenger $\mathcal{C}$ and all oracles. It works as follows.

Setup: $\mathcal{B}$ executes the Setup$(1^\lambda, d)$ algorithm and outputs (q, G, G_T, e). We note that g is the generator of G and $g_1 = g^\gamma$. Then, $\mathcal{B}$ computes $g_T = e(g, g)$ and picks index $\ell \in \{1, 2, \ldots, q_{H_1}\}$ at random. $\mathcal{B}$ controls random oracles $H_i(i = 1, 2, 3, 4)$. It also publishes system public parameters $pp = \{G, G_T, d, q, g, e, g_T, g_1, H_1, H_2, H_3, H_4\}$ and keeps master secret key $MSK = \gamma$.

H_1 query: $\mathcal{A}_1$ makes the H_1 query adaptively. $\mathcal{B}$ responds to $\mathcal{A}_1$'s query on $\mathcal{B}$ as shown below. $\mathcal{B}$ maintains the list $L_{H_1} = (ID_i, K_i, Q_i, Cert_i)$. If the query ID_i appears on L_{H_1} in an item $(ID_i, K_i, Q_i, Cert_i)$, it returns corresponding Q_i to $\mathcal{A}_1$. Otherwise, if the query is on ID_ℓ, $\mathcal{B}$ sets $Q_\ell = g^b$, returns Q_ℓ to $\mathcal{A}_1$, and adds $(ID_\ell, K_\ell, Q_\ell, \perp)$ to L_{H_1}. Note that $K_\ell = \{K_{1\ell}, K_{2\ell}\}$. Otherwise, $\mathcal{B}$ does the following things:

(1) Select $t_i \in Z^*_p$ at random. Let $Q_i = g^{t_i}$ and $\text{Cert}_i = (g^a)^{t_i}$.

(2) Add the tuple $(ID_i, K_i, Q_i, Cert_i)$ to L_{H_1} and respond with Q_i to $\mathcal{A}_1$.

H_2 query: $\mathcal{A}_1$ makes the H_2 query adaptively. $\mathcal{B}$ makes a response to $\mathcal{A}_1$'s query on (ID_i, K_i, g_1) as shown below. $\mathcal{B}$ maintains the list $L_{H_2} = (ID_i, K_i, \nu_i, R_i)$. If the query (ID_i, K_i) appears on L_{H_2} in an item (ID_i, K_i, ν_i, R_i), it returns corresponding R_i to $\mathcal{A}_1$. Else, $\mathcal{B}$ picks $\nu_i \in Z^*_p$ at random and calculates $R_i = g^{\nu_i}$, and then $\mathcal{B}$ adds (ID_i, K_i, ν_i, R_i) to L_{H_2} and responds to $\mathcal{A}_1$ with R_i.

H_3 query: $\mathcal{A}_1$ makes the H_3 query adaptively. $\mathcal{B}$ responds to $\mathcal{A}_1$'s query on (D_{i_1}, D_{i_2}) as shown below. $\mathcal{B}$ maintains the list $L_{H_3} = \{(D_{i\_1}, D_{i\_2}, h_3)\}$. If the query (D_{i_1}, D_{i_2}) appears on L_{H_3} in an item (D_{i_1}, D_{i_2}, h_3), $\mathcal{B}$ returns corresponding h_3 to $\mathcal{A}_1$. Otherwise, $\mathcal{B}$ picks $h_3 \in Z^*_p$ at random, adds (D_{i_1}, D_{i_2}, h_3) to L_{H_3}, and responds to $\mathcal{A}_1$ with h_3.

H_4 query: $\mathcal{A}_1$ makes the H_4 query adaptively. $\mathcal{B}$ responds to $\mathcal{A}_1$'s query on ID_i as shown below. $\mathcal{B}$ maintains the list $L_{H_4} = \{(ID_i, h_4)\}$. If the query ID_i appears on L_{H_4} in an item (ID_i, h_4), $\mathcal{B}$ returns corresponding h_4 to $\mathcal{A}_1$.

Otherwise, $\mathscr{B}$ picks $h_4 \in Z_p^*$ at random, adds (ID_i, h_4) to L_{H_4}, and returns h_4 to $\mathscr{A}_1$.

$\mathcal{O}^{PublicKey}$ query: $\mathscr{A}_1$ makes the $\mathcal{O}^{PublicKey}$ query adaptively. $\mathscr{B}$ responds to $\mathscr{A}_1$'s query on ID_i as shown below. If the query ID_i is already on L_1 in an item (ID_i, K_i, SK_i, T_i), $\mathscr{B}$ returns corresponding K_i to $\mathscr{A}_1$. Otherwise, $\mathscr{B}$ randomly picks $\alpha_i' \in Z_p$ and $\overrightarrow{x}_i' \in Z_q^d$, and then it computes $SK_{1i} = \alpha_i'$ and $SK_{2i} = \alpha_i \mathbf{x}_i'$ as secret keys. It computes $K_{1i} = g^{SK_{1i}}$ and $K_{2i} = g^{SK_{2i}}$ as public keys, and then it adds (ID_i, K_i, SK_i, T_i) to L_1 and responds to $\mathscr{A}_1$ with K_i.

$\mathcal{O}^{\text{PublicKeyReplace}}$ query: $\mathscr{A}_1$ makes the $\mathcal{O}^{\text{PublicKeyReplace}}$ query adaptively. On receiving the query on (ID_i, K_i', SK_i, T_i), $\mathscr{B}$ retrieves items related to ID_i in L_1 and updates the item (ID_i, K_i, SK_i, T_i) to $(ID_i, K_i', \perp, 1)$.

$\mathcal{O}^{\text{SecretKey}}$ query: $\mathscr{A}_1$ makes the $\mathcal{O}^{\text{SecretKey}}$ query adaptively. On receiving the query on ID_i, $\mathscr{B}$ searches the entry (ID_i, K_i, SK_i, T_i) of ID_i. If $T_i = 0$, $\mathscr{B}$ returns SK_i to $\mathscr{A}_1$. Otherwise, $\mathscr{B}$ aborts.

$\mathcal{O}^{\text{Certificate}}$ query: $\mathscr{A}_1$ makes the $\mathcal{O}^{\text{Certificate}}$ query adaptively. On receiving the query on ID_i, if $ID_i = ID_l$, $\mathscr{B}$ aborts. Otherwise, $\mathscr{B}$ searches the item $(ID_i, K_i, Q_i, Cert_i)$ of ID_i and responds to $\mathscr{A}_1$ with $Cert_i$.

$\mathcal{O}^{\text{Decrypt}}$ ***query:*** *$\mathscr{A}_1$ makes the $\mathcal{O}^{\text{Decrypt}}$ query adaptively by submitting CT and ID_i to $\mathscr{B}$. We note that $CT = (C_1, C_2, \ldots, C_n)$ and $n \le N$. $\mathscr{B}$ responds to the query from $\mathscr{A}_1$ on (ID_i, CT) as shown in the following:*

(1) $\mathscr{B}$ searches item (ID_i, K_i, SK_i, T_i) of list L_1. If the user of ID_i is not an intended recipient, it rejects the query.

(2) If ID_ℓ is in the intended recipient set and $T_i = 0$, $\mathscr{B}$ searches list L_{H_3} to find the entry (D_{i_1}, D_{i_2}, h_3) that satisfies $C_{i\_0} = g^{r_i}$ and $C_{i\_3} = r_i \oplus H_3(D_{i\_1}, D_{i\_2})$. If there is no item that satisfies the condition, $\mathscr{B}$ discards CT and aborts. Else, $\mathscr{B}$ responds to $\mathscr{A}_1$ with D_{i_1}, D_{i_2}.

(3) If ID_ℓ is not in the intended recipient set and $T_i = 1$, $\mathscr{A}_1$ should give SK_i corresponding to K_i of ID_i. Then, $\mathscr{B}$ checks whether $K_{1i} = g^{SK_{1i}}$ and $K_{2i} = g^{SK_{2i}}$ hold. If not so, $\mathscr{B}$ aborts. Otherwise, $\mathscr{B}$ searches L_{H_3} to find the entry (D_{i_1}, D_{i_2}, h_3) that satisfies $C_{i\_0} = g^{r_i}$ and $C_{i\_3} = r_i \oplus H_3(D_{i\_1}, D_{i\_2})$. If there is no item that satisfies the condition, $\mathscr{B}$ rejects the query. Otherwise, it returns D_{i_1}, D_{i_2} to the adversary $\mathscr{A}_1$.

(4) Otherwise, $\mathscr{B}$ obtains SK_i and $Cert_i$ which are related to ID_i, and then $\mathscr{B}$ executes $\text{Decrypt}(pp, CT, ID_i, SK_i, Cert_i)$ and responds to $\mathscr{A}_1$ with the result.

Phase 1: during this phase, $\mathscr{B}$ issues the above queries launched by $\mathscr{A}_1$ adaptively. For responding to the queries, $\mathscr{B}$ maintains a list $L_1 = \{ID_i, K_i, SK_i, T_i\}$. This list was initially empty. $T_i = 0$ represents that K_i has not been replaced by $\mathscr{A}_1$. Otherwise, $T_i = 1$ means that $\mathscr{A}_1$ has made replacement of K_i.

Challenge: $\mathscr{A}_1$ submits the intended recipient set $S = \{ID_1, ID_2, ID_3, \ldots, ID_n\}$, $(n \le N)$, message $M = \overrightarrow{y}$, and two user identities (ID_0^*, ID_1^*) to the challenger $\mathscr{C}$, with the requirement that, in Phase 1, it neither obtained certificates of users in $S \cup (ID_0^*, ID_1^*)$ nor made replacement of K_i for ID_i in $S \cup (ID_0^*, ID_1^*)$. Then, $\mathscr{B}$ randomly selects a value $\mu \in \{0, 1\}$ and sets $S_\mu^* = S \cup ID_\mu^*$. If ID_ℓ is not in S^*, $\mathscr{B}$ aborts. Else, $\mathscr{B}$ sets $C_{i\_0}^* = g^c$ and chooses $H_4(\chi_i)^* \in Z_p^*$, $C_{i\_1}^* \in G_T$, $C_{i\_2}^* \in G_T$, and $C_{i\_3}^* \in G_T$ at random. Then, the challenge broadcast ciphertext $CT^* = (C_1^*, C_2^*, \ldots, C_n^*)$ is returned to $\mathscr{A}_1$.

Phase 2: $\mathscr{A}_1$ issues a serious of queries adaptively. However, it cannot issue queries for certificates or decryption of ID_i in $S^* \cup (ID_0^*, ID_1^*)$.

Guess: $\mathscr{A}_1$ outputs a guess $\mu' \in \{0, 1\}$ for μ. It wins the game if $\mu' = \mu$. For ID_ℓ, the description of CT^* is shown as follows. To produce the result, $\mathscr{B}$ should calculate D_{i_1} and D_{i_2} correctly.$\mathscr{B}$ chooses an item from L_{H_3} at random and searches v_ℓ from the item $(ID_\ell, K_\ell, v_\ell, R_\ell)$. To solve the CBDH problem, $\mathscr{B}$ computes $T_s = D_{i\_1} \cdot D_{i\_2} / \{C_{\ell\_1}^* \cdot C_{\ell\_2}^* \cdot e(K_\ell, C_{\ell\_0}^*)^{v_\ell}\}$ as the solution.

If $D_{i\_1} = C_{1\ell\_1}^* \cdot e(C_{\ell\_0}^*, Cert_\ell)$ and $D_{i\_2} = C_{\ell\_2}^* \cdot e(C_{\ell\_0}^*, R_\ell)^{SK_{1\ell}}$, then $C_{\ell\_1}^* = D_{i\_1} / e(C_{\ell\_0}^*, Cert_\ell)$ and $C_{\ell\_2}^* = D_{i\_2} / e(C_{\ell\_0}^*, R_\ell)^{SK_{1\ell}}$. $\mathscr{B}$ can extract the solution $T_s = D_{i\_1} \cdot D_{i\_2} / C_{\ell\_1}^* \cdot C_{\ell\_2}^* \cdot e(K_\ell, C_{\ell\_0}^*)^{v_\ell} = e(g, g)^{abc}$.

Analysis: then, we analyze the probability that the given CBDH problem can be solved by the challenger $\mathscr{C}$.

If $\mathscr{B}$ does not abort during the game, then $\mathscr{A}_1$'s view is identical to its view in the real scheme. Furthermore, we have $|Pr[\mu' = \mu] - 1/2| \ge \epsilon$. The game may be aborted before it finishes. Let Abort denote the game is aborted before it finishes. Then, event Abort occurs under any of the following occasions. (1)$\mathbf{Ab}_1$: ID_ℓ is not in S^* during the Challenge phase. We have $Pr[Ab_1] = N - n/N$. (2)$\mathbf{Ab}_2$: $\mathscr{B}$ aborts in the period that CT is given to $\mathcal{O}^{\text{Decrypt}}$. We have $Pr[Ab_2] = q_{\text{dec}}/2^\lambda$. (3)$\mathbf{Ab}_3$: $\mathscr{A}_1$ issues $\mathcal{O}^{\text{Certificate}}$ query on ID_ℓ. We have $Pr[Ab_3] = (q_{H_1} - 1/q_{H_1})^{q_{\text{cert}}}$. (4)$\mathbf{Ab}_4$: $\mathscr{A}_1$ issues $\mathcal{O}^{\text{Secret}}$ query on ID_i, and $\mathscr{A}_1$ has made replacement of K_i for ID_i. If Ab_2 occurs, then Ab_4 also occurs. So, $Pr[\text{Abort}] = Pr[Ab_1 \wedge Ab_2 \wedge Ab_3 \wedge Ab_4] \ge n/N(1 - q_{\text{dec}}/2^\lambda)(q_{H_1} - 1/q_{H_1})^{q_{\text{cert}}}$.

Let Oca denote $H_3 Query | Abort$. So, we have $Pr[\mu' = \mu | \text{Oca}] = 1/2$ and $Pr[\mu' = \mu] \le 1/2(Pr[\text{Oca}] + 1)$. By the definition of the probability for $\mathscr{A}_1$ in ANO-CBIP-CCA Game 1, we have $\varepsilon \le 2|Pr[\mu' = \mu] - 1/2| \le Pr[\text{Oca}] \le Pr[H_3 Query]/Pr[\text{Abort}]$. So, we have $Pr[H_3 Query] \ge \varepsilon Pr[\text{Abort}] \ge n\varepsilon/N(1 - q_{\text{dec}}/2^\lambda)(q_{H_1} - 1/q_{H_1})^{q_{\text{cert}}}$. Finally, $\mathscr{B}$ selects the correct item from L_{H_3} with probability $1/q_{H_3}$. Consequently, $\mathscr{B}$'s advantage is at least $n\varepsilon/q_{H_3} N(1 - q_{\text{dec}}/2^\lambda)(q_{H_1} - 1/q_{H_1})^{q_{\text{cert}}}$ as required. □

Theorem 4. *Suppose that hash functions $H_i (i = 1, 2, 3, 4)$ are random oracles and $\mathscr{A}_1$ is able to launch q_{pubkey} queries to $\mathcal{O}^{PublicKey}$, q_{seckey} queries to $\mathcal{O}^{SecretKey}$, q_{dec} queries to $\mathcal{O}^{Decrypt}$, and $q_{H_i} (i = 1, 2, 3, 4)$ to functions $H_i (i = 1, 2, 3, 4)$, respectively. It has advantage ε over our proposed scheme in ANO-CBIP-CCA Game 2. Then, there exists a PPT algorithm $\mathscr{B}$ to solve the CBDH problem with the advantage at least $n\varepsilon/q_{H_3} N(1 - q_{dec}/2^\lambda)(q_{pubkey} - 1/q_{pubkey})^{q_{seckey}}$.*

Proof. Suppose that there exists an adversary $\mathscr{A}_2$ that can break the proposed scheme in ANO-CBIP-CCA Game 2 with advantage ε. We build an algorithm $\mathscr{B}$ to solve the

CBDH problem by running $\mathcal{A}_2$. Given as the input a problem instance (g, g^a, g^b, g^c), $\mathcal{A}_2$ needs to simulate a challenger $\mathcal{C}$ and all oracles. It works as follows.

Setup: $\mathcal{B}$ executes the Setup$(1^\lambda, d)$ algorithm and outputs (q, G, G_T, e). We note that g is the generator of G and $g_1 = g^\gamma$. Then, $\mathcal{B}$ computes $g_T = e(g, g)$ and picks index $\ell \in \{1, 2, \ldots, q\}$ at random. $\mathcal{B}$ controls random oracles $H_i (i = 1, 2, 3, 4)$. It also publishes system public parameters $pp = \{G, G_T, d, q, g, e, g_T, g_1, H_1, H_2, H_3, H_4\}$ and gives master secret key MSK $= \gamma$ to $\mathcal{A}_2$. Random oracles $H_i (i = 1, 2, 3, 4)$ are controlled by $\mathcal{B}$.

H_1 query: $\mathcal{A}_2$ makes the H_1 query adaptively. $\mathcal{B}$ makes a response to $\mathcal{A}_2$'s query on (ID_i, K_i) as shown below. It maintains the list $L_{H_1} = \{(ID_i, K_i, Q_i)\}$. If the query ID_i appears on L_{H_1} in an item (ID_i, K_i, Q_i), it returns corresponding Q_i to $\mathcal{A}_2$. Else, $\mathcal{B}$ chooses $Q_i \in G$ at random. Then, it adds (ID_i, K_i, Q_i) to L_{H_1} and responds with Q_i to $\mathcal{A}_2$.

H_2 query: $\mathcal{A}_2$ makes the H_2 query adaptively. $\mathcal{B}$ makes a response to $\mathcal{A}_2$'s query on (ID_i, K_i, g_1) as shown below. $\mathcal{B}$ maintains the list $L_{H_2} = (ID_i, K_i, R_i)$. If the query (ID_i, K_i) appears on L_{H_2} in an item (ID_i, K_i, R_i), it returns corresponding R_i to $\mathcal{A}_2$. Else, if $ID_i = ID_\ell$, it sets $R_\ell = g^a$, returns R_ℓ to $\mathcal{A}_2$, and adds $(ID_\ell, K_\ell, R_\ell)$ to L_{H_2}. Note that $K_\ell = \{K_{1\ell}, K_{2\ell}\}$. Otherwise, $\mathcal{B}$ randomly selects $R_i \in G$. It adds (ID_i, K_i, R_i) to the list L_{H_2} and responds with R_i to $\mathcal{A}_2$.

H_3 query: $\mathcal{A}_2$ makes the H_3 query adaptively. H_3 responds to $\mathcal{A}_2$'s query on D_{i_1}, D_{i_2} as shown below. H_3 maintains the list $L_{H_3} = \{(D_{i\_1}, D_{i\_2}, h_3)\}$. If the query D_{i_1}, D_{i_2} appears on L_{H_3} in an item (D_{i_1}, D_{i_2}, h_3), $\mathcal{B}$ returns corresponding h_3 to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ picks $h_3 \in Z_p^*$ at random, adds (D_{i_1}, D_{i_2}, h_3) to LH3 and responds to $\mathcal{A}_2$ with h_3.

H_4 query: *$\mathcal{A}_2$ makes the H_4 query adaptively. $\mathcal{B}$ makes a response to $\mathcal{A}_2$'s query on ID_i as shown below. $\mathcal{B}$ has the list* $L_{H_4} = \{(ID_i, h_4)\}$. If the query ID_i appears on L_{H_4} in an item (ID_i, h_4), $\mathcal{B}$ returns corresponding h_4 to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ picks $h_4 \in Z_p^*$ at random, adds (ID_i, h_4) to L_{H_4}, and returns h_4 to $\mathcal{A}_2$.

$\mathcal{O}^{\text{PublicKey}}$ query: $\mathcal{A}_2$ makes the $\mathcal{O}^{\text{PublicKey}}$ query adaptively. $\mathcal{B}$ makes a response to $\mathcal{A}_2$'s query on ID_i as shown below. If the query ID_i appears on L_{H_2} in an item (ID_i, K_i, R_i), $\mathcal{B}$ returns related K_i to $\mathcal{A}_2$. Otherwise, if the query is on ID_ℓ, $\mathcal{B}$ randomly selects $\overrightarrow{x}'_i \in Z_q^d$. Then, it returns $K_{1\ell} = g^b$ and $K_{2\ell} = g^{b \cdot \overrightarrow{x}'_i}$ to $\mathcal{A}_2$ and adds $(ID_\ell, K_\ell, \perp)$ to L_2 while $K_\ell = (K_{1\ell}, K_{2\ell})$. The secret key $SK_{1\ell} = b$ is unknown to $\mathcal{B}$. Else, $\mathcal{B}$ randomly picks $\alpha'_i \in Z_p$ and $\overrightarrow{x}'_i \in Z_q^d$, and then $\mathcal{B}$ computes $SK_{1i} = \alpha'_i$ and $SK_{2i} = \alpha_i \overrightarrow{x}'_i$ as secret keys. It computes $K_{1i} = g^{SK_{1i}}$ and $K_{2i} = g^{SK_{2i}}$ as public keys, and then it adds (ID_i, K_i, SK_i) to L_2 and returns K_i to $\mathcal{A}_2$.

$\mathcal{O}^{\text{SecretKey}}$ query: $\mathcal{A}_2$ makes the $\mathcal{O}^{\text{SecretKey}}$ query adaptively. On receiving the query on ID_i, if $ID_i = ID_\ell$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ searches the entry (ID_i, K_i, SK_i) of ID_i in L_2 and returns SK_i to $\mathcal{A}_2$.

$\mathcal{O}^{\text{Decrypt}}$ query: $\mathcal{A}_2$ makes the $\mathcal{O}^{\text{Decrypt}}$ query adaptively by submitting CT and ID_i to $\mathcal{B}$. We note that $CT = (C_1, C_2, \ldots, C_n)$ and $n \le N$. $\mathcal{B}$ makes a response to the query from $\mathcal{A}_2$ on (ID_i, CT) as shown in the following:

(1) $\mathcal{B}$ searches item (ID_i, K_i, SK_i) of list L_2. If the user of ID_i is not an intended recipient, it rejects the query.

(2) If ID_ℓ is in the intended recipient set, $\mathcal{B}$ searches L_{H_3} to find the entry (D_{i_1}, D_{i_2}, h_3) that satisfies $C_{i\_0} = g^{r_i}$ and $C_{i\_3} = r_i \oplus H_3(D_{i\_1}, D_{i\_2})$. If there is no item that satisfies the condition, $\mathcal{B}$ rejects the query. Otherwise, it returns D_{i_1}, D_{i_2} to the adversary $\mathcal{A}_2$.

(3) Otherwise, $\mathcal{B}$ obtains SK_i and $Cert_i$ which are related to ID_i, and then $\mathcal{B}$ executes Decrypt$(pp, CT, ID_i, SK_i, Cert_i)$ and responds to $\mathcal{A}_2$ with the result.

Phase 1: during this phase, $\mathcal{B}$ issues the above queries launched by $\mathcal{A}_2$ adaptively. For responding to the queries, $\mathcal{B}$ maintains a list $L_2 = \{ID_i, K_i, SK_i\}$. This list was initially empty.

Challenge: when $\mathcal{A}_2$ decides that Phase 1 is over, it submits the intended recipient set $S^* = \{ID_1, ID_2, ID_3, \ldots, ID_n\}$, $(n \le N)$, two distinct messages (M_0, M_1) and $M_0 = \overrightarrow{y}_0, M_1 = \overrightarrow{y}_1$, and then $\langle \overrightarrow{x}, \overrightarrow{y}_0 \rangle = \langle \overrightarrow{x}, \overrightarrow{y}_1 \rangle$. It sends (M_0, M_1) to the challenger $\mathcal{C}$, with the requirement that, in Phase 1, it has not obtained SK_i of ID_i in S^*. Then, $\mathcal{B}$ selects a random value $M_\mu, \mu \in \{0, 1\}$. If ID_ℓ is not in S^*, $\mathcal{B}$ aborts. Else, it sets $C^*_{i\_0} = g^c$ and chooses $H_4(\chi_i)^* \in Z_p^*$, $C^*_{i\_1} \in G_T$, $C^*_{i\_2} \in G_T$ and $C^*_{i\_3} \in G_T$ at random. Then, the challenge broadcast ciphertext $CT^* = (C_1^*, C_2^*, \ldots, C_n^*)$ is returned to the adversary $\mathcal{A}_2$.

Phase 2: $\mathcal{A}_2$ issues a set of queries adaptively. However, it cannot issue queries for SK_i of ID_i in S^* or decryption of ID_i in S^*.

Guess: $\mathcal{A}_2$ outputs a guess $\mu' \in \{0, 1\}$ for μ. It wins the game if $\mu' = \mu$. For ID_ℓ, the description of CT^* is shown as follows. To produce the result, $\mathcal{B}$ should calculate D_{i_1} and D_{i_2} correctly. $\mathcal{B}$ chooses an item from L_{H_3} at random and searches v_ℓ from the item $(ID_\ell, K_\ell, v_\ell, R_\ell)$. To solve the CBDH problem, $\mathcal{B}$ computes $T = D_{i\_1} \cdot D_{i\_2} / \{C^*_{\ell\_1} \cdot C^*_{\ell\_2} \cdot e(K_\ell, C^*_{\ell\_0})^{v_\ell}\}$ as the solution.

If $D_{i\_1} = C^*_{1\ell\_1} \cdot e(C^*_{\ell\_0}, Cert_\ell)$ and $D_{i\_2} = C^*_{\ell\_2} \cdot e(C^*_{\ell\_0}, R_\ell)^{SK_{1\ell}}$, then $C^*_{\ell\_1} = D_{i\_1} / e(C^*_{\ell\_0}, Cert_\ell)$ and $C^*_{\ell\_2} = D_{i\_2} / e(C^*_{\ell\_0}, R_\ell)^{SK_{1\ell}}$. $\mathcal{B}$ can extract the solution $T = D_{i\_1} \cdot D_{i\_2} / C^*_{\ell\_1} \cdot C^*_{\ell\_2} \cdot e((C^*_{\ell\_0})^\alpha, Q_\ell) = e(g, g)^{abc}$.

Analysis: then, we analyze the probability that the given CBDH problem can be solved by the challenger $\mathcal{C}$.

If $\mathcal{B}$ does not abort during the game, then $\mathcal{A}_2$'s view is identical to its view in the real scheme. Furthermore, we have $|Pr[\mu' = \mu] - 1/2| \ge \epsilon$. The game may be aborted before it finishes. Let Abort denote the game is aborted before it finishes. Then, event Abort occurs under any of the following occasions. (1)$\mathbf{Ab}_1$: the adversary $\mathcal{A}_2$ queries the oracle $\mathcal{O}^{\text{SecretKey}}$ on the user ID_ℓ. We have $Pr[Ab_1] = (q_{\text{pubkey}} - 1/q_{\text{pubkey}})^{q_{\text{seckey}}}$. (2)$\mathbf{Ab}_2$: $\mathcal{B}$ aborts in the period that CT is given to $\mathcal{O}^{Decrypt}$. We have $Pr[Ab_2] = q_{\text{dec}}/2^\lambda$. (3)$\mathbf{Ab}_3$: ID_ℓ is not in S^* during the Challenge phase. We have $Pr[Ab_3] = N - n/N$. So, we have that $Pr[\text{Abort}] = Pr[Ab_1 \wedge Ab_2 \wedge Ab_3] \ge n/N(1 - q_{\text{dec}}/2^\lambda)\ (q_{\text{pubkey}} - 1/q_{\text{pubkey}})^{q_{\text{seckey}}}$.

Let Oca denote $H_3 Query|Abort$. So, we have $Pr[\mu' = \mu|\text{Oca}] = 1/2$ and $Pr[\mu' = \mu] = 1/2(Pr[\text{Oca}] + 1)$. By the

definition of the advantage for $\mathcal{A}_2$ in ANO-CBIP-CCA Game 2, we have $\varepsilon \leq 2|Pr[\mu' = \mu] - 1/2| \leq Pr[\text{Oca}] \leq Pr[H_3 \text{ Query}]/Pr[\text{Abort}]$. So, we have $Pr[H_3\text{Query}] \geq \varepsilon Pr[\text{Abort}] \geq n\varepsilon/N(1 - q_{\text{dec}}/2^{\lambda})(q_{\text{pubkey}} - 1/q_{\text{pubkey}})^{q_{\text{seckey}}}$. Finally, $\mathcal{B}$ selects the correct item from L_{H_3} with probability $1/q_{H_3}$. Consequently, $\mathcal{B}$'s advantage is at least $n\varepsilon/q_{H_3}N(1 - q_{\text{dec}}/2^{\lambda})(q_{\text{pubkey}} - 1/q_{\text{pubkey}})^{q_{\text{seckey}}}$ as required. □

6. Implementation and Evaluation

6.1. Theoretical Analysis. In Table 2, we give analytical measurements for public parameters' size, user secret keys' size, ciphertext size, encryption cost, and decryption cost of the IBBE-IP scheme [8] and the proposed scheme.

Table 2 shows that our scheme has a significant advantage over the IBBE-IP scheme on decryption cost. The decryption cost of our scheme is $3P + 3E + 3E_T + 5M_T$ which is constant, while the decryption cost of the IBBE-IP scheme is $3P + (2n + d + 3)E + (2n + d)E_T + 4M_T$ which grows multiplicatively in n and d. Our scheme also optimizes the public parameters' size for the reason that our public parameters' size is constant, while the IBBE-IP scheme is linear with n.

As for the ciphertext size, the ciphertext size of our scheme is linear with the number of recipients n, while the ciphertext size of the IBBE-IP scheme is linear with the vector length d. However, there is a restriction in IBBE-IP that the recipient number has to be less than vector length ($n < d$). So, the increasing recipient number will lead to the growth of vector length, and the ciphertext size is also increasing as a result.

It is obvious that our scheme achieves better performance than the existing scheme in the aspects of public parameters' size and decryption time according to the analytical measurements.

6.2. Experimental Implementation. To evaluate the performance of the proposed scheme in practice, we give a reference implementation of our scheme and IBBE-IP scheme in Python language. We use the Charm library [36] to implement the pairing group operations and Flint library [37] for the finite field arithmetic in Z_q. Our experiments are performed on a Linux desktop with 8 GB of RAM and an 8-core Intel Core i7-8550U 2.00 GHz processor to evaluate the above theoretical analysis illustration. In our implementation, we use the SS512 curve in the Charm library. We get the average result over ten runs.

Figure 2(a) shows that the encryption and key generation time of our scheme increase with the growing vector length given the certain number of recipients, while the decryption time remains constant at the same time. Figure 2(b) shows that encryption time is linear with the number of recipients in our scheme. Decryption time remains constant regardless of the number of recipients. Figure 3(a) shows that the ciphertext size of our scheme remains constant with the growing vector length given the certain number of recipients. Figure 3(b) shows that the ciphertext size is linear with the number of recipients given a certain vector length in our scheme.

In Table 3, we give a more detailed computation time and ciphertext size of our scheme with the change of vector length and the intended recipient number. In order to achieve higher efficiency, we have precomputed $e(g_1, Q_i), e(K_{1i}, R_i), e(K_{2i}, R_i)$ and have stored them in lists. We see that the ciphertext size rises from 1.0 KB to 5.8 KB when the recipient number grows from 3 to 19. Key generation time and encryption time grow from 3.3 ms to 48.2 ms and from 50.9 ms to 696.8 ms, respectively, as the recipient number and vector length grow. Decryption time is approximately 3.9 ms.

Figure 4(a) shows the ciphertext size difference between our scheme and IBBE-IP scheme. The ciphertext size of the IBBE-IP scheme is linear with the vector length with the restriction that the number of recipients is less than the vector length ($n < d$), while our scheme has no restriction. Especially, as we can see from Figure 4(a), with the growing of recipient number n, the vector length d has to grow, and the ciphertext size is also increasing in the IBBE-IP scheme. As it is also shown in Table 3, the ciphertext size of our scheme is independent of the vector length. CT is linear with the number of recipients in our scheme because our scheme enables that different intended users in S obtain their corresponding inner product via the decryption of CT, and it achieves stronger plaintext protection. It avoids a security threat existing in a trivial solution that the sender encrypts a message under an inner product encryption first and then encrypts the ciphertext with a broadcast encryption. The threat is that once the decryption result of broadcast encryption is made public, all users in the inner product encryption system obtain the inner product ciphertext and are able to calculate their own inner product value. In our scheme, we avoid this threat. If there are users that maliciously expose the decryption result of broadcast encryption, others will not be able to obtain their corresponding inner product by the result. This leads to further protection to the plaintext.

Figure 4(b) shows our scheme's significant advantage in decryption cost. In our implementation, this decryption time of IBBE-IP does not include the Pollard kangaroo algorithm runtime, while our scheme's decryption time includes the baby-step giant-step algorithm runtime. Besides, the number of recipients needs to be less than the vector length ($n < d$) in IBBE-IP, so we let $d = n + 1$ in the measurement of decryption time. As we can see from Figure 4(b), the decryption time of IBBE-IP is linear with the recipient number and the vector length, while the decryption time of our scheme is constant. In our scheme, it is about 4.0 ms, and it is independent on the vector length and the recipient number.

Obviously, our scheme is efficient according to the above analytical measurements and experimental evaluation because of its constant decryption cost. In addition, differing from our scheme, IBBE-IP scheme has the restriction that the number of recipients is less than the vector length [8]. Therefore, our scheme is applicable to those scenarios in that a number of recipients with limited computation capacity need to obtain the inner product values through decryption regularly.

Table 2: Efficiency comparison.

	IBBE-IP [8]	Our scheme
Public parameters' size	$O(n)$	$O(1)$
User secret keys' size	$O(1)$	$O(d)$
Ciphertext size	$O(d)$	$O(n)$
Encryption cost	$(n + 5d + 7)E + 2P + dM_T + (1 + 3d)E_T$	$4nP + (2 + d)E + 3nM_T + 2nE_T$
Decryption cost	$3P + (2n + d + 3)E + (2n + d)E_T + 4M_T$	$3P + 3E + 3E_T + 5M_T$

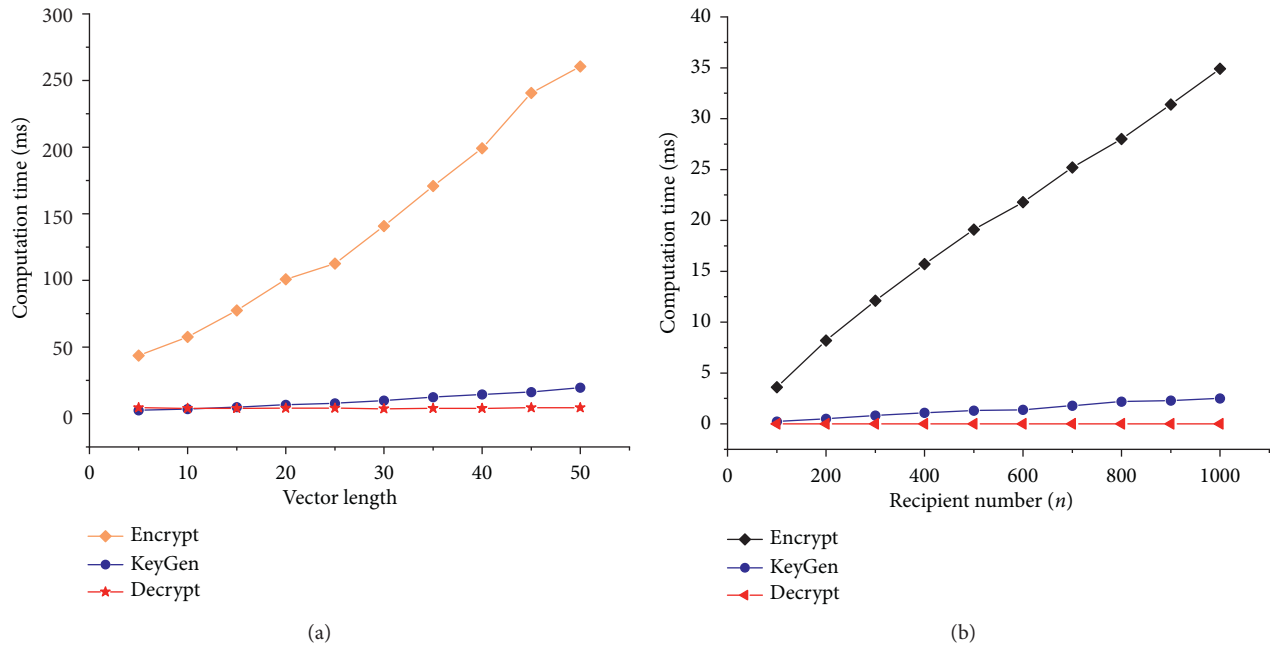

Figure 2: Our scheme's computation time.

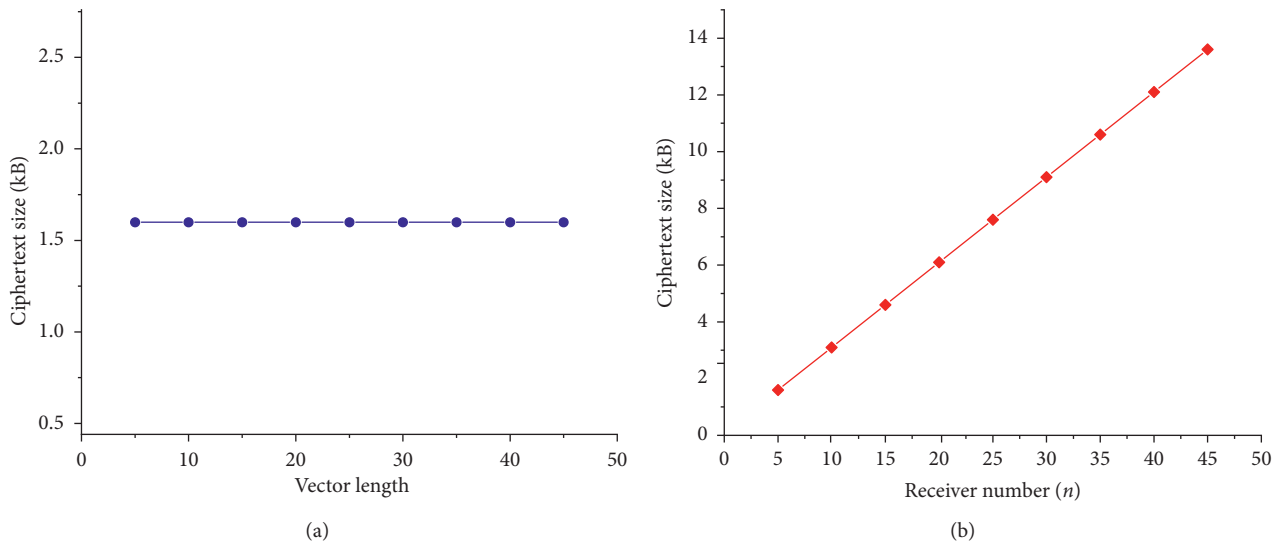

Figure 3: Our scheme's ciphertext size.

Table 3: Performance of our scheme.

Vector length	Recipient number	KeyGen time (ms)	Enc time (ms)	Dec time (ms)	Ciphertext size (KB)
$d = 10$	$n = 3$	3.3	50.9	3.9	1.0
	$n = 6$	7.2	105.7	4.0	1.9
	$n = 9$	10.4	159.3	3.8	2.8
$d = 20$	$n = 13$	27.7	414.3	3.8	4.0
	$n = 16$	32.5	475.6	3.9	4.9
	$n = 19$	48.2	696.8	3.9	5.8

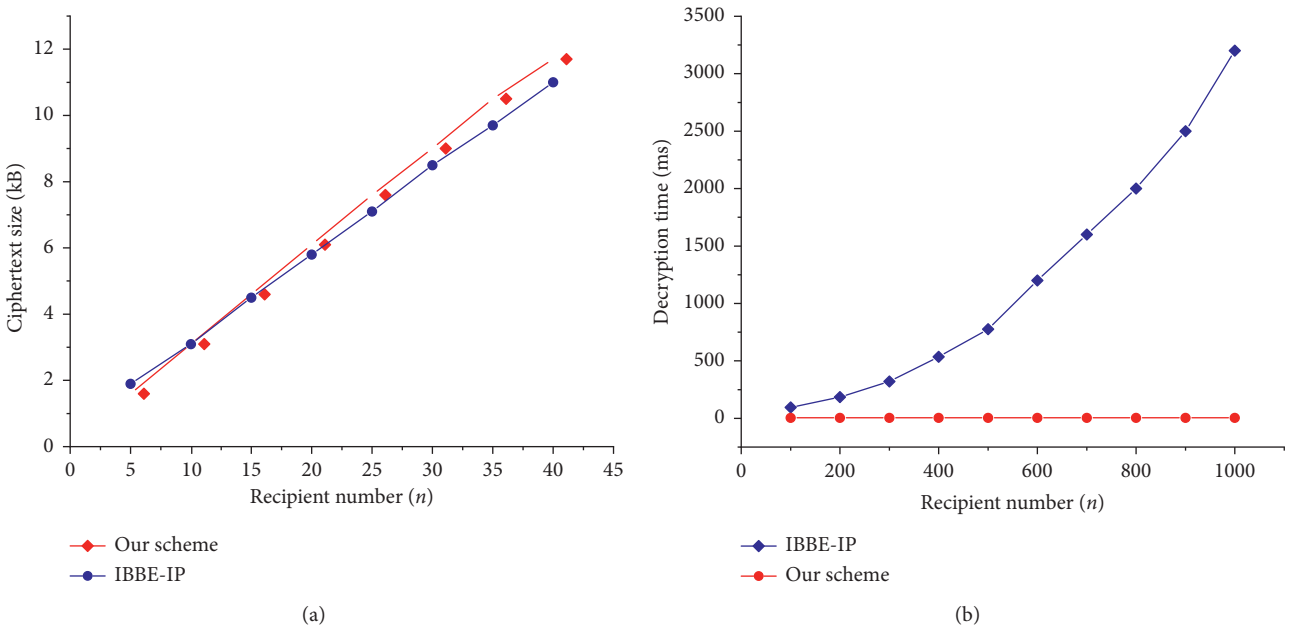

Figure 4: Comparison between the existing scheme and our scheme.

7. Conclusion and Future Work

In this paper, we propose a certificate-based inner product broadcast encryption with anonymity due to the limitation in efficiency and recipient privacy in the present broadcast encryption for inner product scheme. Concrete construction and formal security definitions are given in this paper. We show that our scheme is adaptively secure under the IND-CCA security model which is different from the previous inner product broadcast encryption under the IND-CPA security model. In addition, the identity of a user is anonymous to others in our scheme. Furthermore, analytical and experimental results show that our scheme enables faster decryption. Because of these good properties, our scheme may have some significant value in some practical applications such as enabling secure group communication in the consortium blockchain. However, the size of the ciphertext is linear with the number of recipients, and how to further reduce ciphertext size is still a challenging problem.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper was supported by the National Natural Science Foundation of China (Grant no. U1736114) and National Key R&D Program of China (Grant no. 2020YFB2103802).

References

[1] A. Fiat and M. Naor, "Broadcast encryption," in *Proceedings of the Annual International Cryptology Conference*, pp. 480–491, Springer, Santa Barbara, CA, USA, August 1993.

[2] M. Naor and B. Pinkas, "Efficient trace and revoke schemes," in *Proceedings of the International Conference on Financial Cryptography*, pp. 1–20, Springer, Kota Kinabalu, Malaysia, February 2000.

[3] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Proceedings of the Theory of Cryptography Conference*, pp. 253–273, Springer, Providence, RI, USA, March 2011.

[4] M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval, "Simple functional encryption schemes for inner products," in *Lecture Notes in Computer Science*, pp. 733–751, Springer, Berlin, Germany, 2015.

[5] S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, and D. J. Wu, "Function-hiding inner product encryption is practical," in *Lecture Notes in Computer Science*, pp. 544–562, Springer, Berlin, Germany, 2018.

[6] Z. Brakerski and G. Segev, "Function-private functional encryption in the private-key setting," *Journal of Cryptology*, vol. 31, no. 1, pp. 202–225, 2018.

[7] A. Bishop, A. Jain, and L. Kowalczyk, "Function-hiding inner product encryption," *Advances in Cryptology -- ASIACRYPT 2015*.Springer, Berlin, Germany, pp. 470–491, 2015.

[8] J. Lai, Y. Mu, F. Guo, P. Jiang, and S. Ma, "Identity-based broadcast encryption for inner products," *The Computer Journal*, vol. 61, no. 8, pp. 1240–1251, 2018.

[9] L. Deng, "Anonymous certificateless multi-receiver encryption scheme for smart community management systems," *Soft Computing*, vol. 24, no. 1, pp. 281–292, 2020.

[10] C. Lin, D. He, X. Huang, X. Xie, and K.-K. R. Choo, "Ppchain: A privacy-preserving permissioned blockchain architecture for cryptocurrency and other regulated applications," *IEEE Systems Journal*, pp. 1–12, 2020.

[11] C. Gentry, "Certificate-based encryption and the certificate revocation problem," *Lecture Notes in Computer Science*, vol. 2656, pp. 272–293, 2003.

[12] C.-I. Fan, P.-J. Tsai, J.-J. Huang, and W.-T. Chen, "Anonymous multi-receiver certificate-based encryption," in *Proceedings of the 2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pp. 19–26, IEEE, Beijing, China, October 2013.

[13] L. Chen, J. Li, and Y. Zhang, "Anonymous certificate-based broadcast encryption with personalized messages," *IEEE Transactions on Broadcasting*, pp. 1–15, 2020.

[14] J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval, "Decentralized multi-client functional encryption for inner product," *Lecture Notes in Computer Science*, Springer, Berlin, Germany, pp. 703–732, 2018.

[15] E. Shen, E. Shi, and B. Waters, "Predicate Privacy in Encryption Systems," *Theory of Cryptography*, , pp. 457–473, Springer, 2009.

[16] P. Datta, R. Dutta, and S. Mukhopadhyay, "Functional encryption for inner product with full function privacy," *Public-Key Cryptography - PKC 2016*, Springer, Berlin, Germany, pp. 164–195, 2016.

[17] F. Benhamouda, F. Bourse, and H. Lipmaa, "Cca-secure inner-product functional encryption from projective hash functions," *Lecture Notes in Computer Science*, Springer, Berlin, Germany, pp. 36–66, 2017.

[18] S. Zhang, Y. Mu, and G. Yang, "Achieving IND-CCA Security for Functional Encryption for Inner Products," *Information Security and Cryptology*, Springer, Berlin, Germany, pp. 119–139, 2017.

[19] M. Abdalla, D. Catalano, D. Fiore, R. Gay, and B. Ursu, "Multi-input functional encryption for inner products: function-hiding realizations and constructions without pairings," *Lecture Notes in Computer Science*, Springer, Berlin, Germany, pp. 597–627, 2018.

[20] P. Datta, T. Okamoto, and J. Tomida, "Full-hiding (unbounded) multi-input inner product functional encryption from the k-linear assumption," in *Proceedings of the IACR International Workshop on Public Key Cryptography*, pp. 245–277, Springer, Edinburgh, UK, May 2018.

[21] H. Wang, K. Chen, T. Pan, and Y. Zhao, "Practical cca-secure functional encryptions for deterministic functions," *Security and Communication Networks*, vol. 2020, Article ID 8823788, 14 pages, 2020.

[22] R. Gay, L. Kowalczyk, and H. Wee, "Tight adaptively secure broadcast encryption with short ciphertexts and keys," *Lecture Notes in Computer Science*, Springer, Berlin, Germany, pp. 123–139, 2018.

[23] C. Delerablée, "Identity-based broadcast encryption with constant size ciphertexts and private keys," *Advances in Cryptology - ASIACRYPT 2007*, Springer, Berlin, Germany, pp. 200–215, 2007.

[24] P. Jiang, F. Guo, and Y. Mu, "Efficient identity-based broadcast encryption with keyword search against insider attacks for database systems," *Theoretical Computer Science*, vol. 767, pp. 51–72, 2019.

[25] D. Lubicz and T. Sirvent, "Attribute-based broadcast encryption scheme made efficient," in *Proceedings of the International Conference on Cryptology in Africa*, pp. 325–342, Springer, Cairo, Egypt, July 2008.

[26] H. Xiong, Y. Zhao, L. Peng, H. Zhang, and K.-H. Yeh, "Partially policy-hidden attribute-based broadcast encryption with secure delegation in edge computing," *Future Generation Computer Systems*, vol. 97, pp. 453–461, 2019.

[27] S. Canard, D.-H. Phan, and V. C. Trinh, "Attribute-based broadcast encryption scheme for lightweight devices," *IET Information Security*, vol. 12, pp. 52–59, 2017.

[28] G. Sravan Kumar and A. Sri Krishna, "Privacy sustaining constant length ciphertext-policy attribute-based broadcast encryption," *Advances in Intelligent Systems and Computing*, Springer, Berlin, Germany, pp. 313–324, 2019.

[29] A. Barth, D. Boneh, and B. Waters, "Privacy in encrypted content distribution using private broadcast encryption," in *Proceedings of the International Conference on Financial Cryptography and Data Security*, pp. 52–64, Springer, Kota Kinabalu, Malaysia, February 2006.

[30] C. Sur, C. D. Jung, and K. H. Rhee, "Multi-receiver certificate-based encryption and application to public key broadcast encryption," in *Proceedings of the 2007 ECSIS Symposium on Bio-inspired, Learning, and Intelligent Systems for Security (BLISS 2007)*, pp. 35–40, IEEE, Edinburgh, UK, August 2007.

[31] B. Zhu, P. Wei, and M. Wang, "Adaptive security of broadcast encryption, revisited," *Security and Communication Networks*, vol. 2017, Article ID 1404279, 16 pages, 2017.

[32] J. Li, L. Chen, Y. Lu, and Y. Zhang, "Anonymous certificate-based broadcast encryption with constant decryption cost," *Information Sciences*, vol. 454-455, pp. 110–127, 2018.

[33] S. Jin and H. Yu-pu, "Fully secure broadcast encryption for inner-product predicates," *Procedia Engineering*, vol. 29, pp. 316–320, 2012.

[34] V. S. Miller, "The weil pairing, and its efficient calculation," *Journal of cryptology*, vol. 17, no. 4, pp. 235–261, 2004.

[35] T. Okamoto and K. Takashima, "Fully secure unbounded inner-product and attribute-based encryption," in *Procedings of the International Conference on the Theory and Application of Cryptology and Information Security*, Singapore, December 2012.

[36] J. A. Akinyele, C. Garman, I. Miers et al., "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, no. 2, pp. 111–128, 2013.

[37] W. B. Hart, "Fast library for number theory: an introduction," *Mathematical Software - ICMS 2010*, Springer, Berlin, Germany, pp. 88–91, 2010.