Identity-Set-based Broadcast Encryption supporting ąřCut-or-Selectąś with Short Ciphertext

Yan Zhu, Xin Wang, RuiQi Guo
University of Science and Technology Beijing
zhuyan@ustb.edu.cn

Di Ma
University of Michigan-Dearborn
dmadma@umich.edu

ABSTRACT

In this paper we present an identity-set-based broadcast encryption scheme with three working modes: *set membership* (Include-mode), *all member* (All-mode), and *negative membership* (Exclude-mode) over the user identity set, simultaneously. The core of our scheme is the implementation of cryptographic representation of subset by using two aggregation functions: Zeros-based aggregation and Poles-based aggregation. These two aggregation functions are capable of compressing any subset into one element in a bilinear map group for determining the membership between an element and a subset. Our scheme achieves the optimal bound of $O(1)$-size for either ciphertext (consisting of just two elements) or decryption key (one element) for an identity set of large size. We prove that our scheme is secure under the General Diffie-Hellman Exponent (GDHE) assumption.

Keywords

Broadcast Encryption, Cryptographic Membership, Aggregation Function

1. INTRODUCTION

Broadcast encryption (BE) is a group-oriented cryptosystem in which a broadcaster encrypts messages and transmits them to a group of users $\mathcal{U}$ who are listening to a broadcast channel and use their private keys to decrypt received messages. [1, 2, 3] One remarkable feature of broadcast encryption is that each user has a unique private key. In the public key setting for $n = |\mathcal{U}|$ users, this feature means that broadcast encryption has a one-to-many (or $1 : n$) public/private key structure. In other words, many different private keys correspond to one public key. This kind of key structure is completely different from traditional $1 : 1$ public/private key structure. The benefits gained from this $1 : n$ key structure include additional functionalities such as traitor tracing [4], dynamic revocation, and so on.

Broadcast encryption can be widely used in many application scenarios ranging from TV subscription services, DVD copyright protection, and encrypted file systems, to secure communication of social networks, e.g., Email, Blog, Web communities. Existing broadcast encryption schemes can be roughly categorized into two groups (related work is discussed in Section 7):

- Category I: Broadcast for multiple designated receivers where a broadcast message is sent securely to a small subset of users $\mathcal{S}$ and $|\mathcal{S}| \ll n$ [2, 5];
- Category II: Broadcast without multiple revoked receivers where a broadcast message is sent to all BUT a small set of revoked users $\mathcal{R}$ and $|\mathcal{R}| \ll n$ [6].

Usually, these two categories of broadcast encryption do not replace each other when n is large, so they can be thought as two complementary approaches. There is little systematic work for integrating these two categories of broadcast encryption into a secure BE system [2, 7, 8] from literature.

The two mechanisms, one-time *designation or inclusion* (of Category I) and *revocation or exclusion* (of Category II) for a subset of users, are extremely important for large-scale IT systems with large-size users. For example, in email system or Blog sites, we often communicate with several specified friends, but also sometimes broadcast to almost users in community. To sum up, we list some desirable features for developing a practical and flexible BE scheme:

1. The scheme should be able to support large-size users and new user's joining at an arbitrary time;
2. Two broadcast mechanisms, designation and revocation, should be supported simultaneously, and the ciphertext size should be independent on the number of designated or revoked users;
3. For easy-to-use, the user's identity (similar as identity-based encryption) should be employed to memory and distinguish the different users;

Our Motivation and Roadmap. In this paper our goal is to develop a new group-oriented encryption which supports two broadcast mechanisms, designation and revocation, simultaneously. We believe that the foundation of constructing such an encryption system is to achieve *cryptographic decisional problem of set membership*, that is, given a subset $\mathcal{S}$ and an element e in $\mathcal{U}$, determine whether or not e belongs to $\mathcal{S}$ (i.e., $e \in \mathcal{S}$ or $e \notin \mathcal{S}$) in a secure (or non-deceptive) way. Obviously, our desired cryptosystem could be constructed by solving the above problem.

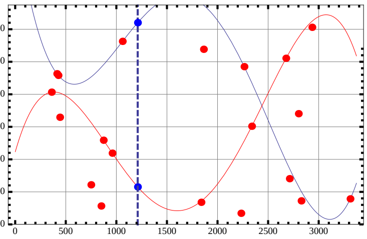

Figure 1: Example of cryptographic representation of subset and aggregation function.

Our approach of solving the above problem is to realize the cryptographic representation of any given subset, and then design a secure aggregation function to compress the subset into a constant-size random element, which would be used to construct our desired encryption system. Further, we show an example of our approach in Figure 1, in which such an approach can be realized by four following steps:

1. Given a set $\mathcal{U} = \{e_1, \cdots, e_n\}$, we map each element e_i into a random point v_i in cryptographic space, where the partial information of these points is published as the public key *mpk* (see the red points in Figure 1);
2. Given a subset $\mathcal{S} \subseteq \mathcal{U}$, we construct a curve $c(x)$ through all random points $\{v_i\}$ included in $\mathcal{S}$ (see two curves in Figure 1, each of which was interpolated from points);
3. We introduce a random secret γ and then the aggregation function is defined as the corresponding point $c(\gamma)$ in this curve, that is, $Aggregate(mpk, \mathcal{S}) \rightarrow c(\gamma)$, where *mpk* is the public cryptographic parameters (see the dashed blue line and the intersections of this line and two curves);
4. We define the security feature of function that intend to ensure the security of aggregation under the malicious attacks (see Definition **??** and **??** in Section **??**).

Finally, we expect to use this kind of aggregation for developing our encryption scheme. The privacy and randomness of aggregation point $c(\gamma)$ will provide guarantee for the security of our scheme. Moreover, the compression property of aggregation function can ensure the $O(1)$-size ciphertext.

Our Contributions. In this paper we first present a new identity-Set-based Broadcast Encryption (SBE) based on cryptographic decisional problem of set membership. By supporting both designation and revocation simultaneously, our scheme allows a nice property of "**Cut-or-Select**". That is, a broadcaster is able to send a message to some selected users or to all but some revoked ones. In detail, our work is listed as follows:

- We propose a new approach to solve cryptographic decisional problem of set membership by designing two aggregation functions: Zeros-based aggregation and Poles-based aggregation. Two corresponding fast algorithms are developed to compress any subset quickly into a constant-size element for designation and revocation mechanisms.
- We present the construction of a SBE scheme with three operation modes over the set of user identities: *set-membership, all member, negative membership* for any subset of users, simultaneously. Moreover, our scheme achieve the optimal bound of $O(1)$-size for either ciphertexts or decryption keys.
- We prove the security of two aggregation functions and provide a complete security proof of our SBE scheme based on general Diffie-Hellman exponent (GDHE) assumption. Our scheme is secure for arbitrarily large collusion of corrupted users. Moreover, our experiments show that our SBE scheme is simple, easy-to-implement and high performance, as well as short (128-byte) and constant-size ciphertext for any size subset.

Organization. The preliminaries and definition of SBE are provided in Section 2. We propose an effective solution for cryptographic set membership in **??**. Our ISBE scheme in Section 3. We present the security and performance analysis in Section 4, 5, and 6. Related work is presented in Section 7 and Section 8 concludes the Lab.

2. OUR DEFINITION OF SBE

We now give a formal definition of identity-Set-based Broadcast Encryption (SBE) with key encapsulation mechanism [2], which is made up of four algorithms, shown as follows:

Setup($\mathbb{S}$): takes as input a bilinear map group system $\mathbb{S}$. It outputs a public key *mpk* and a master secret key *msk*, where *mpk* contains a list of user's profiles pp.

KeyGen(msk, ID_k): takes as input *msk* and a user's identity ID_k. It outputs the user's secret key sk_k and adds a user's profile pp_k to pp, i.e., $\mathrm{pp} = \mathrm{pp} \cup \{pp_k\}$.

Encrypt($mpk, \mathcal{S}, mode$): takes as input *mpk*, a set of user's identities $\mathcal{S}$, and a mode of operation *mode*, where *mode* belongs to one of three modes in $\{u \in \mathcal{S}, u \in \mathrm{ALL}, u \notin \mathcal{S}\}$. It outputs a ciphertext C and a random session key *ek*, where $(\mathcal{S}, mode)$ is included in C.

Decrypt(mpk, sk_k, C): takes as input *mpk*, a ciphertext C, and a user's secret key sk_k. If this user satisfies the access mode *mode* then the algorithm will decrypt the ciphertext C and return a session key *ek*.

In SBE scheme, the *user's profile* includes the identity of this user and a public parameter generated in registry. Our scheme makes use of these profiles to realize encryption and decryption for a subset of users. As a group-oriented cryptosystem, we employ a list of profiles to realize the management of memberships. According to the different of operations, the set of users $\mathcal{S}$ will be used in three cases:

Include-mode ($\mathbf{u} \in \mathcal{S}$): used to specify multiple receivers, where $\mathcal{S}$ denotes a set of specified users, such that the user $u \in \mathcal{S}$ will be authorized to decrypt the message.

All-mode ($\mathbf{u} \in \mathrm{ALL}$): used to specify all receivers, and all users is specified to decrypt the message.

Exclude-mode ($\mathbf{u} \notin \mathcal{S}$): used to revoke multiple receivers, where $\mathcal{S}$ denotes a set of revoked users, such that the user $u \notin \mathcal{S}$ will be authorized to decrypt the message.

Consider all possible *mpk* from $Setup(\mathbb{S}) \rightarrow (mpk, msk)$, a valid ciphertext C from $Encrypt(mpk, \mathcal{S}, mode) \rightarrow (\mathrm{C}, ek)$ and $KeyGen(msk, \mathrm{ID}_k) \rightarrow sk_k$. If the user's identity ID_k satisfies the operation mode $(\mathcal{S}, mode)$ in C, then the decryption algorithm will retrieve the session key *ek*, i.e.,

$$\Pr\left[\begin{array}{c} Decrypt(mpk, sk_k, \mathrm{C}) = ek : \\ mode(\mathrm{ID}_k, \mathcal{S}) = 1 \end{array}\right] = 1,$$

where $mode(\mathrm{ID}_k, \mathcal{S}) = 1$ denotes the boolean judgment over $mode := \{u \in \mathcal{S}, u \in \mathrm{ALL}, u \notin \mathcal{S}\}$ for a certain ID_k and a set of user's identities $\mathcal{S}$.

We now describe a game-based security definition of our SBE scheme. We define a selective-set model for proving the security of SBE under chosen plaintext attack (CPA). Given a challenge ciphertext C^* with $(\mathcal{S}^*, mode)$, the attacker can repeatedly ask for secret keys $\mathcal{R} = \{(\mathrm{ID}_i, sk_i)\}$ corresponding to a given *mode* used in C^*, but we have $mode(\mathrm{ID}_i, \mathcal{S}) = 0$ for all possible ID_i in the corrupted keys $\mathcal{R}$, where i is a user counter. The security game follows.

- **Setup.** Given a *mode* and a set $\mathcal{S}^*$, the challenger runs the Setup algorithm and gives *mpk* to the adversary.

- **Learning.** The adversary makes n times repeated private keys queries for a user's identity ID_i. The challenger returns $\mathrm{KeyGen}(\mathrm{MK}, \phi) \rightarrow (sk_i, pp_i)$ if $mode(\mathrm{ID}_i, \mathcal{S}^*) = 0$. Otherwise, it merely returns a public profile pp_i, where t is the number of all corrupted secret keys and $|\mathcal{S}^*| = n - t$.

- **Challenge.** The challenger completes $\mathrm{Encrypt}(mpk, \mathcal{S}^*, mode) = (\mathrm{C}, ek)$, and then flips a random coin $b = \{0, 1\}$ and sets $ek_b = ek$ and ek_{1-b} to a random element of $\mathbb{G}_T$. The ciphertext (C, ek_0, ek_1) is given to the adversary.

- **Guess.** The adversary outputs a guess b' of b.

The advantage of an adversary $\mathcal{A}$ in this game is defined as $Adv^{\mathrm{IND},mode}_{\mathrm{SBE},\mathcal{A}}(n, t) = |\Pr[b' = b] - 1/2|$ for three modes. A SBE scheme is (n, t)-secure against colluders [3, 9] if all polynomial time adversaries have at most a negligible advantage in the above game.

3. OUR CONSTRUCTION

3.1 Aggregation Functions of Subsets

In this section, we illustrate our basic idea to design cryptographic construction for set membership. In our idea, the core notion is cryptographic representation of subset based on aggregation functions. Given a set $\mathcal{U}$, an **aggregation function** is a cryptographic function to compress the information of any subset $\mathcal{S} \subseteq \mathcal{U}$ into a constant-size value. The output of aggregation function is called the *cryptographic representation of subset*. The definition of this function is stated as follows:

Definition 1 (Aggregate function). *Let$\mathcal{PK}$ denote the public key space over a group $\mathbb{G}$ and$\mathcal{U} = \{e_1, \cdots, e_n\}$ be a set of elements, the function Aggregate : $\mathcal{PK} \times 2^U \rightarrow \mathbb{G}$ is a deterministic polynomial time algorithm satisfying:*

$$Aggregate(mpk, \mathcal{S}) = R_{\mathcal{S}}, \tag{1}$$

where mpk is the public key in $\mathcal{PK}$, a subset $\mathcal{S} \subseteq \mathcal{U}$, and $R_{\mathcal{S}}$ is a random enough element in $\mathbb{G}$ to avoid guessing.

This kind of aggregation function is our core in our SBE scheme and foundation of cryptographic decisional problem of set membership (see Section 1). In this paper, we use this kind of function to build membership and negative membership. More importantly, the constant size ciphertext can be implemented in our SBE scheme only if the compression property can be efficiently realized. Based on the presented approach in Section 1, we present two aggregate functions, ZerosAggr and PolesAggr, that realize the decision over set membership and negative membership, respectively.

Before our aggregation functions are introduced, we first give the definition of zeros and poles in a function as follows:

Definition 2 (Zeros and Poles). *A rational polynomial function has the form $H(x) = \frac{P(x)}{Q(x)}$ that is the quotient of two polynomial $P(x)$ and $Q(x)$. We say the value z is a zero of $H(x)$ if $P(z) = 0$, and z is a pole of $H(x)$ if $Q(z) = 0$.*

Next, we propose two aggregation functions, ZerosAggr and PolesAggr, for $u \in \{v_i\}$ and $u \notin \{v_i\}$ as below.

given a secret γ and a subset $\mathcal{S}$, we propose the definition of aggregation function that can aggregate the information of $\mathcal{S}$ into the constant-size value $g^{f_{\mathcal{S}}(\gamma)}$ based on the above polynomial $f_{\mathcal{S}}(x)$. We call it the Zeros-based Aggregation (in short, **ZerosAggr**) function since the hash values of all elements in $\mathcal{S}$ are used for the (negative) zeros in the polynomial $f_{\mathcal{S}}(x)$. The algorithm is defined as follows:

Definition 3 (Zeros-based Aggregation). *Given a subset $\mathcal{S} = \{e_1, \cdots, e_m\} \in \mathcal{U}$ and a cyclic group $\mathbb{G}$, an algorithm is called Zeros-based Aggregation function if there exists a polynomial-time algorithm ZerosAggr that outputs*

$$G_{\mathcal{S}} = \mathrm{ZerosAggr}(mpk, \mathcal{S}) = g^{\gamma \cdot \prod_{e_i \in \mathcal{S}} (\gamma + x_i)}, \tag{2}$$

where, $mpk = \{g_i = g^{\gamma^i}\}_{i \in [1, |\mathcal{U}|]}$ is the public parameter, g is a generator in $\mathbb{G}$, $x_i = hash(e_i)$ and γ is a secret.

Definition 4 (Poles-based Aggregation). *Given a subset $\mathcal{R} = \{e_1, \cdots, e_m\} \in \mathcal{U}$ and a cyclic group $\mathbb{G}$, an algorithm is called Poles-based Aggregation function if there exists a polynomial-time algorithm PolesAggr that outputs*

$$H_{\mathcal{R}} = \mathrm{PolesAggr}(mpk, \mathcal{R}) = h^{\frac{1}{\prod_{e_i \in \mathcal{R}} (\gamma + x_i)}}, \tag{3}$$

where, $mpk = \{h_i = h^{\frac{1}{\gamma + x_i}}\}_{e_i \in \mathcal{U}}$ is the public parameter, h is a generator in $\mathbb{G}$, $x_i = hash(e_i)$ and γ is a secret.

3.2 Our Construction

We now present our SBE scheme with three modes, such as Include, ALL, and Exclude. In this scheme, we assume that each user has a unique identity ID (e.g., email address) and all users in cryptosystem make up a full set $\mathcal{U} = \{\mathrm{ID}_1, \cdots, \mathrm{ID}_n\}$, where the size of $\mathcal{U}$ is not restricted. Given a subset of users $\mathcal{S}$ in $\mathcal{U}$, we now present the SBE providing two basic access control mechanisms: set membership decision over $u \in \mathcal{S}$ and negative membership decision over $u \notin \mathcal{S}$, simultaneously.

Our SBE scheme is illustrated in Figure 4. In this scheme, we choose the bilinear map system $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot, \cdot))$ of prime order p and two generators g and h in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively [10, 11], where p is the order of groups. Additionally, the algorithm will employ a hash function $hash : \{0, 1\}^* \rightarrow \mathbb{Z}_p^*$, mapping any identity ID described as a binary string to a random element $x_i \in \mathbb{Z}_p^*$, that is, $x_i = hash(\mathrm{ID}_i)$.

- Setup: we redefine two random elements G and H as the generators in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We require $G \neq g$ and G is a secret. Let m be the maximum number of aggregated users in the *ZerosAggr* algorithm. Usually, we set $m \leq n/2$ because the sender can use Include-mode when the size of $\mathcal{S}$ is greater than $n/2$. In addition, the public profile *pp* is used to list all users in system and their public tags, i.e., $\{(\mathrm{ID}_i, H_i)\}$. The initial status is defined as $\varnothing$. As a group-oriented cryptosystem, the parameter *pp* is usually shared through public media, e.g., web, facebook, where each user can search his friends profiles in a convenient way. Note that, we must keep two values, G and G^{ϵ}, secret because of the following reasons:

Setup($\mathbb{S}$): chooses two elements $G \xleftarrow{R} \mathbb{G}_1$, $H \xleftarrow{R} \mathbb{G}_2$, and two exponents $\gamma, \epsilon \xleftarrow{R} \mathbb{Z}_p^*$. And then set $R = e(G,H)^\epsilon$ and sets $G_k = G^{\gamma^k}$ for $k \in [1,m]$. The master key is outputted as $msk = (\gamma, \epsilon, G, G^\epsilon)$ and the public key is $mpk = \left\{\mathbb{S}, H, R, \{G_k\}_{k\in[1,m]}, pp = \phi\right\}$.

KeyGen(msk, ID_k): Given an user's identity ID_k, defines $x_k = hash(\mathrm{ID}_k)$ and computers the k-th user's secret key

$$sk_k = G^{\frac{x_k \epsilon}{\gamma + x_k}}, \qquad and \qquad H_k = H^{\frac{\epsilon}{\gamma+x_k}},$$

where, $pp_k = (\mathrm{ID}_k, H_k)$ is appended to pp, i.e., $\mathrm{pp} = \mathrm{pp} \cup \{pp_k\}$.

Encrypt($mpk, \mathcal{S}, mode$): picks a random $s \xleftarrow{R} \mathbb{Z}_p^*$ and executes the following process:

- Case $mode := (u \in \mathcal{S})$: invokes **PolesAggr**($mpk, \mathcal{S}$) $\rightarrow H_{\mathcal{S}} = H^{\epsilon \prod_{e_i \in \mathcal{S}} \frac{1}{\gamma + x_i}}$, then computes
$$C_1 = H^s, \qquad \text{and} \qquad C_2 = (H_{\mathcal{S}})^s.$$
- Case $mode := (u \notin \mathcal{S})$: invokes **ZerosAggr**($mpk, \mathcal{S}$) $\rightarrow G_{\mathcal{S}} = G^{\gamma \cdot \prod_{e_i \in \mathcal{S}}(\gamma + x_i)}$, then computes
$$C_1 = H^s, \qquad \text{and} \qquad C_2 = (G_{\mathcal{S}})^s$$

Finally, the ciphertext is published as $C = (\mathcal{S}, mode, C_1, C_2)$. The corresponded session key is $ek = R^s$.

Decrypt(mpk, sk_k, C): chooses one action from two following cases according to the $mode$ in C:

- Case $mode := (u \in \mathcal{S})$: checks whether ID_k is a member of $\mathcal{S}$, that is, $\mathrm{ID}_k \in \mathcal{S}$. If true, it sets $\mathcal{S}_- = \mathcal{S} \setminus \{\mathrm{ID}_k\}$ and invokes **ZerosAggr**($mpk, \mathcal{S}_-$) $\rightarrow G_{\mathcal{S}_-} = G^{\gamma \cdot \prod_{e_i \in \mathcal{S}_-}(\gamma + x_i)}$. Next it retrieves the session key
$$ek' = e(sk_k, C_1) \cdot e(G_{\mathcal{S}_-}, C_2). \tag{4}$$
- Case $mode := (u \notin \mathcal{S})$: checks whether ID_k satisfies the relation $\mathrm{ID}_k \notin \mathcal{S}$. If true, it sets $\mathcal{S}_+ = \mathcal{S} \cup \{\mathrm{ID}_k\}$ and invokes **PolesAggr**($mpk, \mathcal{S}_+$) $\rightarrow H_{\mathcal{S}_+} = H^{\epsilon \prod_{e_i \in \mathcal{S}_+} \frac{1}{(\gamma + x_i)}}$. Next, it also retrieves the session key
$$ek' = e(sk_k, C_1) \cdot e(C_2, H_{\mathcal{S}_+}). \tag{5}$$

Figure 2: The full construction of set-based broadcast encryption (SBE).

- When the adversary knows the value G^ϵ, the decryption is executed by using
$$e(G^\epsilon, C_1) = e(G^\epsilon, H^s) = e(G,H)^{s\epsilon} = ek.$$
- When the adversary knows G and a sub-ciphertext $C_2 = H^{\frac{s\epsilon}{\gamma + x_i}}$, the decryption is implemented by
$$e(G_1 \cdot G^{x_i}, C_2) \quad = \quad e(G^{\gamma + x_i}, H^{\frac{s\epsilon}{\gamma+x_i}}) = e(G,H)^{s\epsilon} = ek.$$

- KeyGen: given a unique identity ID_k, the user's secret key sk_k is only an element in $\mathbb{G}_1$. In addition, the new user's profile pp_k is also appended into pp. The scheme allows adding new members into the system anytime, and the total number of users is unlimited in the system.
- Encrypt: the sender may select one of Include and Exclude mechanisms to implement secure broadcast, where there is no limit for the number of "cut-or-section" users. These two mechanisms employ two *Aggregate* functions, ZerosAggr for Exclude-mode and PolesAggr for Include-mode, which can aggregate all information of user's identities in $\mathcal{S}$ into two group elements $G_{\mathcal{S}}$ and $H_{\mathcal{S}}$, respectively. We will introduce this algorithm in detail in Section 3.4. And then, the pair $(H, G_{\mathcal{S}})$ or $(H, H_{\mathcal{S}})$ is used to generate the ciphertext
$$(C_1, C_2) = \begin{cases} (H^s, (G_{\mathcal{S}})^s) & \text{for } (u \in \mathcal{S}) \\ (H^s, (H_{\mathcal{S}})^s) & \text{for } (u \notin \mathcal{S}) \end{cases}$$
Note that, a significant feature of our scheme is short and constant-size ciphertext.
- Decrypt: this process only needs two steps for a successful decryption: firstly, the receiver invokes two *Aggregate* functions, ZerosAggr for Include-mode and PolesAggr for Exclude-mode, taken as input $\mathcal{S}_-$ or $\mathcal{S}_+$, respectively. Secondly, the above result will be used to decrypt the ciphertext by using two bilinear maps.

We verify that the decryption works correctly as follows:

Case $mode := (u \in \mathcal{S})$**:** when $\mathrm{ID}_k \in \mathcal{S}$, we have $\mathcal{S}_- = \mathcal{S} \setminus \{e_k\}$ and $G_{\mathcal{S}_-} = G^{\prod_{e_i \in \mathcal{S}, e_i \neq e_k}(\gamma + x_i)}$ can be computed from the *ZerosAggr* algorithm. Based on this value, we check whether the triple (C_1, C_2) in ciphertext matches the private key sk_k by using

$$\begin{aligned} & e(sk_k, C_1) \cdot e(G_{\mathcal{S}_-}, C_2) \\ = \;& e\left(G^{\frac{x_k \epsilon}{\gamma + x_k}}, H^s\right) \cdot e\left(G^{\frac{\gamma}{\gamma+x_k} \prod\limits_{e_i \in \mathcal{S}} (\gamma + x_i)}, H^{s \cdot \epsilon \prod\limits_{e_i \in \mathcal{S}} \frac{1}{\gamma + x_i}}\right) \\ = \;& e(G,H)^{\frac{x_k s\epsilon}{\gamma + x_k}} \cdot e(G,H)^{\frac{\gamma s \epsilon}{\gamma + x_k}} = e(G,H)^{s\epsilon} = R^{s\epsilon} = ek. \end{aligned}$$

Case $mode := (u \notin \mathcal{S})$**:** when $\mathrm{ID}_k \notin \mathcal{S}$, we have $\mathcal{S}_+ = \mathcal{S} \cup \{e_k\}$ and $H_{\mathcal{S}_+} = H^{\frac{1}{\gamma + x_k} \prod_{e_i \in \mathcal{S}} \frac{1}{(\gamma + x_i)}}$ can be computed from the *PolesAggr* algorithm. Based on this value, we check whether the triple (C_1, C_2) in ciphertext matches the private key sk_k by using

$$\begin{aligned} & e(sk_k, C_1) \cdot e(C_2, H_{\mathcal{S}_+}) \\ = \;& e\left(G^{\frac{x_k \epsilon}{\gamma+x_k}}, H^s\right) \cdot e\left(G^{s\gamma \prod_{e_i \in \mathcal{S}} (\gamma+x_i)}, H^{\frac{\epsilon}{\gamma+x_k} \prod_{e_k \in \mathcal{S}} \frac{1}{\gamma+x_i}}\right) \\ = \;& e(G,H)^{\frac{x_k s\epsilon}{\gamma+x_k}} \cdot e(G,H)^{\frac{\gamma s \epsilon}{\gamma+x_k}} = e(G,H)^{s\epsilon} = R^{s\epsilon} = ek. \end{aligned}$$

In summary, our scheme is easy-to-understand and the ciphertexts and decryption keys are constant size.

3.3 Construction for ALL-mode

We provide a solution for mode $u \in$ ALL, that means that all users can be authorized only if everyone of them holds a valid key. This mode is usually realized based on Include-mode with $\mathcal{S} = \mathcal{U}$, but it need to provide a complete list of all users. A more effective method is to consider ALL-mode as a special case of Exclude-mode with $\mathcal{S} = \varnothing$, where $\varnothing$ denotes the empty set. The reason is that $u \in$ ALL is logically equivalent to $u \notin \varnothing$. In this case, the work mode is defined as $u \notin \varnothing$. We provide this process as follows:

- Encrypt($mpk, \varnothing, u \in$ ALL): picks a random integer $s \in \mathbb{Z}_p$ and computes the following cipertext $C = (\varnothing, u \in \text{ALL}, C_1, C_2)$, where
$$C_1 = H^s, \qquad C_2 = G_1^s.$$
The corresponded session key is $ek = R^s$.
- Decrypt(mpk, sk_k, C): When the mode $u \in$ ALL in C is found, it makes use of sk_k and the corresponding public profile H_k to recover ek
$$\begin{aligned} ek &= e(sk_k, C_2) \cdot e(C_1, H_k) \\ &= e\left(G^{\frac{x_k \epsilon}{\gamma+x_k}}, H^s\right) \cdot e\left(G^{s\gamma}, H^{\frac{1}{\gamma+x_k}}\right) \\ &= e(G,H)^{\frac{x_k s\epsilon}{\gamma+x_k}} \cdot e(G,H)^{\frac{\gamma s\epsilon}{\gamma+x_k}} = e(G,H)^{s\epsilon} = R^{s\epsilon}. \end{aligned}$$

In contrast with Include-mode with $\mathcal{S} = \mathcal{U}$, our construction does not require to invoke the aggregation function. Moreover, the broadcaster does not need to provide (or know) a list of all users' identities, such that the total number of receivers is unlimited in one time broadcast.

3.4 Construction of Aggregation Functions

Given a subset $\mathcal{S}$, the aggregate functions defined in Equation (2) and (3) are used repeatedly in our constructions, such that it is crucial to compute the output values $(G_\mathcal{S}, H_\mathcal{S})$ from the public key mpk in an efficient way. We provide a fast recursive method to realize them as follows:

3.4.1 Implement of ZerosAggr function

To implement fast ZerosAggr, we first extract the related information $\{G_i = G^{\gamma^i}\}_{i\in[1,m]}$ from mpk, where γ is an unknown secret. Let $|\mathcal{S}| = t$ and we require $t < m$. We provide a fast recursive way to realize the ZerosAggr function: given all $\{x_i = hash(e_i)\}_{e_i\in\mathcal{S}}$, we define the polynomial of X as

$$f_\mathcal{S}(X) = \prod\nolimits_{v_i \in \mathcal{S}} (X + x_i) = \sum\nolimits_{k=0}^{t} a_k X^k \pmod p$$

and compute the coefficient $a_k \in \mathbb{Z}_p$ for all $k \in [0,t]$. We can use the recursive process to obtain these coefficients. Let $a_i^{(j)}$ denote the value of a_i at the j-th cycle for $j = 1, \cdots, t$ and $i = 0, \cdots, j$. For each cycle, a new x_k is appended into the polynomial. After an inital coefficient is set as $a_0^{(0)} = 1$, the coefficients can be computed repeatedly in each cycle as

$$\left\{ \begin{array}{lcll} a_k^{(k)} &=& a_{k-1}^{(k-1)} & (k \geq 1) \\ a_{k-1}^{(k)} &=& a_{k-2}^{(k-1)} + a_{k-1}^{(k-1)} \cdot x_k & (k \geq 2) \\ \cdots && \cdots & \\ a_1^{(k)} &=& a_0^{(k-1)} + a_1^{(k-1)} \cdot x_k & (k \geq 2) \\ a_0^{(k)} &=& a_0^{(k-1)} \cdot x_k & (k \geq 1) \end{array} \right. .$$

After t cycles running, the value $a_0^{(t)}, \cdots, a_t^{(t)}$ are outputed as $a_0, \cdots, a_t$. It is obvious that the output of ZerosAppr is $G_\mathcal{S} = G^{\gamma \cdot f_\mathcal{S}(\gamma)}$. In face, although γ is unknown, we make use of $(G_1, \cdots, G_m)$ and $t < m$ to compute $G_\mathcal{S}$ as following

$$\begin{aligned} G_\mathcal{S} &= G^{\gamma \cdot f_\mathcal{S}(\gamma)} = G^{\gamma \cdot \prod_{v_i \in \mathcal{S}} (\gamma + x_i)} \\ &= G^{\sum_{k=1}^{t+1} a_{k-1} \cdot \gamma^k} = \prod\nolimits_{k=1}^{t+1} G_k^{a_{k-1}}. \end{aligned} \tag{6}$$

Note that, when $\mathcal{S} = \emptyset$ and $t = 0$, the output of this algorithm is $ZerosAggr(mpk, \mathcal{S}) = G_1$. This value is used to realize the broadcast for ALL users.

```
Algorithm ZerosAggr($mpk, \mathcal{S}$)
  Begin
  $B[1] = 1$;
  for $s = 1$ to $t$ do
    $B[s+1] = B[s]$;
    for $r = s$ downto 2 do
      $B[r] = B[r-1] + B[r] * x_s$;
    end for
    $B[1] = B[1] * x_s$;
  end for
  $sum = 0$;
  for $k = 1$ to $t+1$ do
    $sum = sum * pow(G_k, B[k])$;
  end for
  Return $sum$;
```

Figure 3: The zeros-based aggregation function.

Based on the above crecursive process, we present a fast algorithm, call ZoresAggr, show in Figure 3. In this algorithm, we design a double-loop structure for computing the coefficients of $F_\mathcal{S}(X)$, where $B[1] = a_0, \cdots, B[t+1] = a_t$. And then, the final result is computed by accumulating all $G_k^{a_{k-1}}$ for $k = 1, \cdots, t+1$. In this process, the function $pow(\cdot)$ is invoked to obtain the power of element in $\mathbb{G}_1$, that is, $pow(G_k, B[k]) = G_k^{B[k]} = G_k^{a_{k-1}}$.

3.4.2 Implement of PolesAggr function

To implement fast aggregation, we first extract the related information $\{(x_i, H_i = H^{\frac{\epsilon}{\gamma+x_i}})\}_{e_i\in\mathcal{S}}$ from mpk. Given H_i and H_j, it is easy to obtain the aggregation equation

$$(H_j/H_i)^{\frac{1}{x_i - x_j}} = (H^{\frac{\epsilon}{\gamma+x_j}} / H^{\frac{\epsilon}{\gamma+x_i}})^{\frac{1}{x_i - x_j}} = H^{\frac{\epsilon}{(\gamma+x_i)(\gamma+x_j)}},$$

where $x_i \neq x_j$ is a prerequisite for this equation. The value $\frac{1}{x_i - x_j}$ modulo p can be computed by using extended Euclidean algorithm (called xGCD) in $\mathbb{Z}_p^*$. Next, we expand this equation to multi-value cases. We define the following denotation $B_{s,r}$, where $1 \leq s < r \leq t$, and $B_{s,r} =$

$H^{\frac{\epsilon}{\prod_{k=s}^{r}(\gamma+x_k)}}$. In the same way, we can compute

$$\begin{array}{rcl} B_{s,r+1} = (B_{s,r}/B_{s+1,r+1})^{\frac{1}{x_{r+1}-x_s}} & & \\ & = & (H^{\frac{\epsilon}{\prod_{k=s}^{r}(\gamma+x_k)}}/H^{\frac{\epsilon}{\prod_{k=s+1}^{r+1}(\gamma+x_k)}})^{\frac{1}{x_{r+1}-x_s}} \\ & = & (H^{\frac{\epsilon}{(\gamma+x_s)}}/H^{\frac{\epsilon}{(\gamma+x_{r+1})}})^{\frac{1}{x_{r+1}-x_s}\cdot\frac{1}{\prod_{k=s+1}^{r}(\gamma+x_k)}} \\ & = & H^{\frac{\epsilon}{(\gamma+x_s)(\gamma+x_{r+1})}\cdot\frac{1}{\prod_{k=s+1}^{r}(\gamma+x_k)}} = H^{\frac{\epsilon}{\prod_{k=s}^{r+1}(\gamma+x_k)}}. \end{array}$$

Finally, the output value $H_{\mathcal{S}} = B_{1,t}$ can be completed by computing sequentially $B_{s,r}$ for $s = [1, t-1]$ and $r = [1, t-s]$, as well as the induction

$$\left\{ \begin{array}{rcl} B_{r,r} & = & H_r \quad \forall r \in [1,t], \\ B_{s,r+1} & = & (B_{s,r}/B_{s+1,r+1})^{\frac{1}{x_{r+1}-x_s}}, \\ & & s \in [1,t-1],\ r \in [1,t-s], \end{array} \right. \tag{7}$$

where $B_{r,r}$ is the initial input H_r for $r = [1, t]$.

```
Algorithm PolesAggr($mpk, \mathcal{S}$)
  Begin
  for $r = 1$ to $t$ do
    $B[r] = H_r$
  end for
  for $s = 1$ to $t-1$ do
    for $r = 1$ to $t-s$ do
      if $x_{r+s} = x_r$ then
        Return 0.
      end if
      $tmp_1 = x_r - x_{r+s}$;
      $tmp_2 = invert(tmp_1, p)$;
      $tmp_3 = B[r+1]/B[r]$;
      $B[r] = pow(tmp_3, tmp_2)$;
    end for
  end for
  Return $B[1]$;
```

Figure 4: The poles-based aggregation function.

We provide the algorithm of the above recursive process, called PolesAggr, in Figure 4. This fast algorithm is derived from Equation (7). In the aggregation process, double loops are used to compute the value of $B[r] = B_{r,r+s}$ for $s = 1, \cdots, t-1$ and $r = 1, \cdots, t-s$, repeatedly. In each cycle, the output value 0 denotes the error if there exist two equivalent values x_{r+s} and x_r. This means that two identities are identical or a hash collision occurred, but two identities ID_{r+s} and ID_r are equivalent with negligible probability according to the property of cryptographic hash function. Next, two functions, $invert(\cdot)$ and $pow(\cdot)$, are used to compute the inverse element and the power of element, that is, $tmp_2 = 1/tmp_1 \pmod p$ and $B[r] = tmp_3^{tmp_2} \in \mathbb{G}_2$.

4. SECURE ANALYSIS

4.1 Security Analysis for Our SBE Scheme

We prove the semantic security of our system by relying on the General Diffie-Hellman Exponent (GDHE) framework in [3, 5]. We overview the GDHE framework in Appendix A. *We will not analyze ALL-mode because this mode can be realized as a special case of Exclude-mode*, that is, $\mathcal{S} = \varnothing$. We start by defining the following computational problem.

According to construction of two aggregation functions, we assume $f(x)$ and $g(x)$ be two known random polynomials of respective degree t and $n-t$ with pairwise distinct roots,

$$\left\{ \begin{array}{rcl} f(X) & = & \prod_{i=1}^{t}(X+x_i) = \sum_{i=0}^{t} a_i \cdot X^i, \\ g(X) & = & \prod_{i=1}^{n-t}(X+x'_i) = \sum_{i=0}^{n-t} b_i \cdot X^i. \end{array} \right.$$

Moreover, $h(x,y,z) = f(x)g(x)yz$ be a three-variable polynomial in a bilinear group system $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$.

Based on these two polynomials, we provide a new computational problem, called GDHE_1 problem, which is used to prove the semantic security of our SBE scheme for Include-mode ($u \in \mathcal{S}$). This problem is defined as follows:

THEOREM 1 ((n,t)-GDHE_1 PROBLEM). *Let $\gamma, \varsigma, \epsilon \in \mathbb{Z}_p^*$ be three secret random variables, $f(X)$ and $g(X)$ are two polynomials described above, and $\hat{G}, \hat{H}$ be generators of $\mathbb{S}$. Given the values in (F_1, F_2, F_3)-$GDHE_1$ problem with*

$$\left\{ \begin{array}{rcl} F_1(\gamma,\varsigma,\epsilon) & = & \left\langle\ \hat{G}^{\epsilon}, \hat{G}^{\gamma\epsilon}, \cdots, \hat{G}^{\gamma^{t-1}\epsilon}, \hat{G}^{\gamma f(\gamma)}, \cdots, \hat{G}^{\gamma^m f(\gamma)}\ \right\rangle, \\ F_2(\gamma,\varsigma,\epsilon) & = & \left\langle \begin{array}{c} \hat{H}^{\epsilon}, \hat{H}^{\gamma\epsilon}, \cdots, \hat{H}^{\gamma^n\epsilon}, \\ \hat{H}^{f(\gamma)g(\gamma)}, \hat{H}^{\varsigma f(\gamma)g(\gamma)}, \hat{H}^{\varsigma\epsilon f(\gamma)} \end{array} \right\rangle, \\ F_3(\gamma,\varsigma,\epsilon) & = & e(\hat{G},\hat{H})^{\epsilon f^2(\gamma)g(\gamma)}, \end{array} \right.$$

and $T \xleftarrow{R} \mathbb{G}_T$, decide whether $e(\hat{G},\hat{H})^{\varsigma\cdot\epsilon\cdot f^2(\gamma)\cdot g(\gamma)} = T$. For any algorithm $\mathcal{A}$ that makes a total of at most q queries to the oracles computing the group operation and the bilinear pairing, the advantage of $\mathcal{A}$ is $Adv_{\mathrm{GDHE}_1,\mathcal{A}}^{\mathrm{IND}}(n,t) \leq \frac{(q+2s+2)^2\cdot d}{2p}$, where $s = n+t+m+4$ and $d = 2n$.

We provide the proof of this theorem in Appendix B. In this proof, we reduce this problem to the weakest case $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$, so that the polynomials (F_1, F_2, F_3, T) in bilinear map group are converted to three polynomials (P, Q, h) in bilinear map group. Then we prove that h is independent of (P, Q) and complete the final proof based on Theorem 4 in Appendix. In fact, we also show the security of this assumption based on the following analysis: Considering that ς only appears in $\hat{H}^{\varsigma\epsilon f(\gamma)}$ and $\hat{H}^{\varsigma f(\gamma)g(\gamma)}$, we only pick them out for three following cases:

Case $\hat{H}^{\varsigma\epsilon f(\gamma)}$: we need to find $\hat{G}^{f(\gamma)g(\gamma)}$ to meet $e(\hat{G}^{f(\gamma)g(\gamma)}, \hat{H}^{\varsigma\epsilon f(\gamma)}) = T$. Let $g(x) = \prod_{i=1}^{n-t}(x+x'_i) = \sum_{i=0}^{n-t} b_i \cdot x^i$, where $b_0 = x'_1 x'_2 \cdots x'_{n-t} \neq 0$ because all $x'_i \neq 0$. So that, this polynomial $f(x)g(x)$ is represented as

$$f(x)g(x) = \sum_{i=0}^{n-t} a_i(x^i f(x)) = b_0 f(x) + \sum_{i=1}^{n-t} b_i(x^i f(x)).$$

But it is infeasible for computing $\hat{G}^{f(\gamma)g(\gamma)} = (\hat{G}^{f(\gamma)})^{b_0} \cdot (\hat{G}^{\gamma^i f(\gamma)})^{b_i}$ because there dose not exist the item $\hat{G}^{f(x)}$ from all known items $\hat{G}^{\gamma f(\gamma)}, \cdots, \hat{G}^{\gamma^m f(\gamma)}$.

Case $\hat{H}^{\varsigma f(\gamma)g(\gamma)}$: we need to find $\hat{G}^{\epsilon f(\gamma)}$ to meet $e(\hat{G}^{\epsilon f(\gamma)}, \hat{H}^{\varsigma f(\gamma)g(\gamma)}) = T$ and it is also infeasible because $f(x) = \prod_{i=1}^{t}(x+x_i) = \sum_{i=0}^{t} a_i \cdot x^i = \sum_{i=0}^{t-1} a_i \cdot x^i + x^t$ is a polynomial of degree t and $a_t = 1$, such that $\hat{G}^{\epsilon f(\gamma)} = \prod_{i=0}^{t-1}(\hat{G}^{\epsilon\gamma^i})^{a_i} \cdot \hat{G}^{\gamma^t}$ cannot be built from all known $\hat{G}^{\epsilon}, \hat{G}^{\gamma\epsilon}, \cdots, \hat{G}^{\gamma^{t-1}\epsilon}$.

Linear combination between $\hat{H}^{\varsigma\epsilon f(\gamma)}$ **and** $\hat{H}^{\varsigma f(\gamma)g(\gamma)}$: assume that there exist two coefficients a, b to satisfy

$$e(\hat{G}^{f(\gamma)\frac{a\gamma g(\gamma)}{\gamma+x'_i}}, \hat{H}^{\varsigma\epsilon f(\gamma)}) \cdot e(\hat{G}^{\epsilon\frac{bf(\gamma)}{\gamma+x_i}}, \hat{H}^{\varsigma f(\gamma)g(\gamma)}) = T$$

So that, we have $\frac{a\gamma}{\gamma+x'_i} + \frac{b}{\gamma+x_i} = 1$. To solve this equation, we have $(a-1)\gamma^2 + (ax_i + b - x_i - x'_i)\gamma + (bx'_i -$

$x_i x_i') = 0$. It is easy find that our solution is $a = 1$ and $b = x_i = x_i'$, but it contradicts with the assumption of $x_i \neq x_i'$ for all x_i and x_i' in $f(\gamma)$ and $g(\gamma)$.

Next, we provide another problem, called GDHE$_2$ problem, which is used to prove the security of our SBE scheme for negative membership (Exclude-mode on $u \notin \mathcal{S}$):

THEOREM 2 ((n, t)-GDHE$_2$ PROBLEM). *Let $\gamma, \varsigma, \epsilon \in \mathbb{Z}_p^*$ be three secret random variables, $f(X)$ and $g(X)$ are two polynomials described above, and $\hat{G}, \hat{H}$ be generators of $\mathbb{S}$. Given the values in (F_1, F_2, F_3)-GDHE$_2$ problem with*

$$\begin{cases} F_1(\gamma,\varsigma,\epsilon) & = & \left\langle \begin{array}{c} \hat{G}^{\epsilon}, \hat{G}^{\gamma\epsilon}, \cdots, \hat{G}^{\gamma^{t-1}\epsilon}, \\ \hat{G}^{\gamma f(\gamma)}, \cdots, \hat{G}^{\gamma^m f(\gamma)}, \hat{G}^{\varsigma\gamma f^2(\gamma)} \end{array} \right\rangle, \\ F_2(\gamma,\varsigma,\epsilon) & = & \left\langle \begin{array}{c} \hat{H}^{\epsilon}, \hat{H}^{\gamma\epsilon}, \cdots, \hat{H}^{\gamma^n \epsilon}, \\ \hat{H}^{f(\gamma)g(\gamma)}, \hat{H}^{\varsigma f(\gamma)g(\gamma)} \end{array} \right\rangle, \\ F_3(\gamma,\varsigma,\epsilon) & = & e(\hat{G},\hat{H})^{\epsilon f^2(\gamma)g(\gamma)}, \end{cases}$$

and $T \xleftarrow{R} \mathbb{G}_T$, decide whether $e(\hat{G},\hat{H})^{\varsigma \cdot f^2(\gamma)\cdot g(\gamma)} = T$. For any algorithm $\mathcal{A}$ that makes a total of at most q queries to the oracles computing the group operation and the bilinear pairing, the advantage of $\mathcal{A}$ is $Adv^{\mathrm{IND}}_{\mathrm{GDHE}_2,\mathcal{A}}(n,t) \leq \frac{(q+2s+2)^2 \cdot d}{2p}$, where $s = n+t+m+4$ and $d = 2n$.

We provide the proof of this theorem in Appendix C. We here give a simple comparison with GDHE$_1$ and GDHE$_2$ in Table 1. As seen from this table, most of items are shared between two problems except a slight different: GDHE$_1$ has a unique item $\hat{G}^{\varsigma\gamma f^2(\gamma)}$ and GDHE$_2$ is $\hat{H}^{\varsigma\epsilon f(\gamma)}$. This means that there is a strong correlation between two problems.

Table 1: Comparison with GDHE$_1$ and GDHE$_2$

	Common Elements	GDHE$_1$	GDHE$_2$
F_1	$\hat{G}^{\epsilon}, \hat{G}^{\gamma\epsilon}, \cdots, \hat{G}^{\gamma^{t-1}\epsilon}$, $\hat{G}^{\gamma f(\gamma)}, \cdots, \hat{G}^{\gamma^m f(\gamma)}$		$\hat{G}^{\varsigma\gamma f^2(\gamma)}$
F_2	$\hat{H}^{\epsilon}, \hat{H}^{\gamma\epsilon}, \cdots, \hat{H}^{\gamma^n\epsilon}$, $\hat{H}^{f(\gamma)g(\gamma)}, \hat{H}^{\varsigma f(\gamma)g(\gamma)}$	$\hat{H}^{\varsigma\epsilon f(\gamma)}$	
F_3	$e(\hat{G},\hat{H})^{\epsilon f^2(\gamma)g(\gamma)}$		

We now prove the semantic security of our SBE scheme based on the security model in Section 2, which has the semantic security against chosen plaintext attacks (IND-CPA) with colluders. Usually, we need two separate proofs that prove the security of SBE scheme under two different modes, Include-mode and Exclude-mode, respectively. However, we found that these two proofs has too many similarities according to the comparison result in Table 1. Therefore, we combine two proofs into a full proof of our SBE scheme. Based on (n,t)-GTDHE$_1$ and (n,t)-GTDHE$_2$, the security of our SBE scheme satisfies the following theorem:

THEOREM 3 (SECURITY OF SBE SCHEME). *Our SBE scheme for both Include-mode and Exclude-mode is semantically secure against chosen plaintext attacks with colluders assuming the (n,t)-GDHE$_1$ and (n,t)-GDHE$_2$ problem is hard in $\mathbb{S}$ for $0 \leq t \leq n$.*

The proof of this theorem is presented in Appendix D. Note that, our scheme is secure for arbitrary large collusion of corrupted users because the number of corrupted users t is not restricted in the above theorem.

5. PERFORMANCE EVALUATION

5.1 Parameter Gneration

Our scheme is constructed on the general bilinear map group system $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$ with prime order p, where decisional Diffie-Hellman is hard. We set up our systems using bilinear pairings introduced by Boneh and Franklin [10]. We define the bilinear pairing takes the form $e : E(\mathbb{F}_q) \times E(\mathbb{F}_q) \to \mathbb{F}^*_{q^2}$ (The definition given here is from [11, 12]), where p is a prime and $k = 2$ is the embedding degree (or security multiplier). It turns out the total number of elements is $\sharp E(\mathbb{F}_q) = q+1$ and $\sharp E(\mathbb{F}_{q^2}) = (q+1)^2$. The order p is some prime factor of $q+1$. We invoke this kind of pairing directly based on the Stanford's PBC library [1]. In order to ensure the security of our scheme, we uses 256-bit base field, which is equivalent to 128-bit security (κ=128-bit) for symmetric encryption [13].

5.2 Performance Analysis

We first provide the performance analysis from two aspects: computation costs and communication overheads. Here, we assume that t denotes the size of included or excluded subset and m denotes the manixium number of aggregated users. We present the computation cost of our SBE scheme in Table 2. We use $[E]$ to denote the computation cost of an exponent operation in $\mathbb{G}$, namely, g^x, where x is a positive integer in $\mathbb{Z}_p$ and $g \in \mathbb{G}$ or $\mathbb{G}_T$. We neglect the computation cost of algebraic operations and simple modular arithmetic operations because they run fast enough [14]. The most complex operation is the computation of a bilinear map $e(\cdot,\cdot)$ between two elliptic points (denoted as $[B]$). The symbols $ZAgg(t)$ and $PAgg(t)$ denote two aggregation algorithms where t denotes the size of subset. It is easy to find that the encryption and decryption costs are related to the performance of aggregation algorithms, but if the aggregation algorithms are excluded, all algorithms have the constant number of operations after m, t are set.

Table 2: Computation overhead of our scheme.

	SBE		
	ALL	Include	Exclude
Setup	$(1+m)[E]+1[B]$		
KeyGen	$2[E]$		
Encrypt	$3[E]$	PAgg(t)+$3[E]$	ZAgg(t)+$3[E]$
Decrypt	$2[B]$	ZAgg(t)+$2[B]$	PAgg(t)+$2[B]$

Then, we analyze the storage and communication costs of our scheme. This means that the length of integer is $l_0 = |q|$ in $\mathbb{Z}_p$. Similarly, we have $l_1 = l_2 = 2|q|$ in $\mathbb{G}_1$ and $\mathbb{G}_2$, and $l_T = 2|q|$ in $\mathbb{G}_T$ for the embedding degree $k = 2$. The storage and communication costs of our scheme is shown in Table 3. We neglect the storage/communication cost of the subset of user's identities because the size of each identity cannot too large as a easy-to-remember string. It is also easy to find that the private key, the ciphertext, and the session key has the constant and short size (e.g., only 64-byte for a ciphertext with point compression) regardless of the size of subset in ciphertext.

In short, our scheme has the advantages as follows:

- The computation cost is low if there exist fast algorithms to realize the aggregation of subset.

[1] http://crypto.standord.edu/pbc/times.html

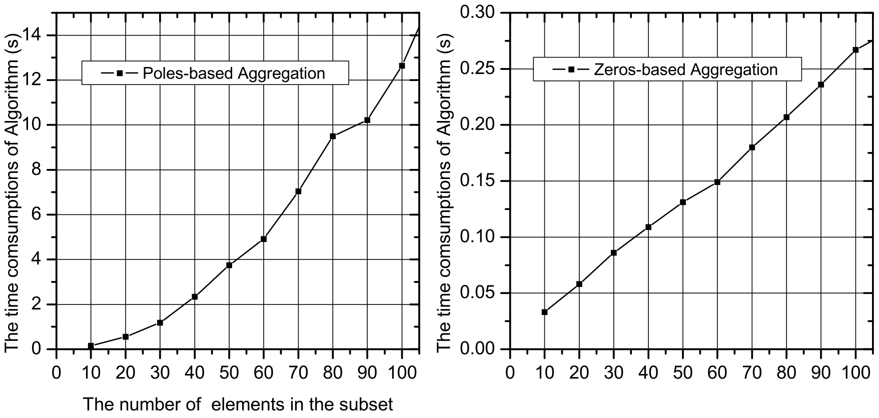

Figure 5: Time overheads of aggregation functions.

Table 3: The storage/communication overhead.

Algorithm		SBE	Our system
Setup	PK	$l_2 + l_T + ml_1 + nl_2$	(m+n+2)*64-Byte
	mk	$l_0 + l_T$	192Byte
KeyGen	SK$^{(k)}$	l_1	64-Byte
Encrypt	C	$l_1 + l_2$	128-Byte
Decrypt	ek	l_T	64-Byte

- The size of private key and ciphertext is constant and short regardless of the size of subset in ciphertext.

The size of public key is related to the size of community (or the total number of users). This is not a large problem in applications because the users have access to a large shared storage medium in which the profiles can be stored [2]. For example, we usually provide a on-line search service for identity query from the profile list in public key. In other applications, each user only needs to store part of profiles, such as his friends' identities, because encryption and decryption just need to input a subset of specified users rather than to know all users.

6. EXPERIMENTAL RESULTS

Using GMP and JPBC libraries, we have developed a Java-language cryptographic library upon which our SBE cryptosystem can be constructed. This Java library contains Pairing-based algorithms on elliptic curves and has been tested on both Windows and Mac OSX platforms. Our SBE cryptosystem is a lightweight software about 600 lines of code built on Eclipse. To evaluate the performance of our SBE cryptosystem, our experiments are run in a Mac laptop with 2.0GHz processor and 4G RAM.

6.0.1 Performance of Aggregation Functions

At first, we evaluate the performance of two aggregation functions. It is easy to analyze the computational costs of two functions from Section 3.4: the ZeorsAggr function needs $O(t^2)$ times multiplication operations in $\mathbb{Z}_p^*$ and $O(t)$ times exponent operations in $\mathbb{G}_1$; and the PolesAggr function needs $O(t^2)$ times exponent operations in $\mathbb{G}_2$ and $O(t^2)$ times xGCD operations in $\mathbb{Z}_p^*$. Hence, the computational costs of PolesAggr is far greater than (roughly t times as) that of ZerosAggr due to the cost of operations in $\mathbb{G}_1$ and $\mathbb{G}_2$ is much larger than that in $\mathbb{Z}_p^*$.

We give the result of experiments for two aggregation functions in Figure 5, in which time consumptions of two functions are showed under the different size of subsets (from 10 to 100). From this figure, it is easy to find that the computational costs of ZerosAggr is proportional to the size of subset, and the costs of PolesAggr grows rapidly for the sustained growth of subset sizes. The memory overheads of two algorithms are proportional to the size of subset. These results are completely consistent with our previous theoretical analysis. However, the time consumption is still high for a large-size subset, so that we can improve the performance by using parallel algorithm or fast elliptic curve algorithm.

6.0.2 Performance under Different Modes

We next analyze the performance of encryption and decryption processes according to three modes, including ALL, Include, and Exclude in Figure 6. The left sugfigure shows the time consumption of encryption for these three modes in which the encryption overhead in Include-mode is greatest of all. The similar result of decryption is showed in the right subfigure but the decryption overhead in Exclude-mode is greatest of all instead of Include-mode. When the system is not large, the users can choose Include-mode or Exclude-mode according to application requirements, e.g., the Include-mode is used to reduce the overhead of decryption for mobile terminals with limited power (smart phone or sensor node) while leaving heavy encryption for high-power servers. In addition, the aggregation functions could be preprocessed in some applications with the fixed receivers, such that encryption and decryption can be achieved rapidly.

The same results are listed in Table 4. As seen from this table, the overhead of encryption or decryption is constant in ALL-mode no matter how large the total number of users is in system. In Include-mode, the encryption overhead is more higher than that of decryption due to the time consumption of PolesAggr is much larger than that of ZerosAggr. The similar situation also appears in Exclude-mode where the decryption overhead is more higher than that of encryption based on the same reason. However, no matter what mode we pick, the ciphertext is only two points on an elliptic curve and the size of ciphertexts is constant for any mode.

6.0.3 Performance of SBE Cryptosystem

Finally, we provide the result of experiments to illustrate the performance of our SBE cryptosystem. Our experiments were implemented by a test (and demo) routine about 1000

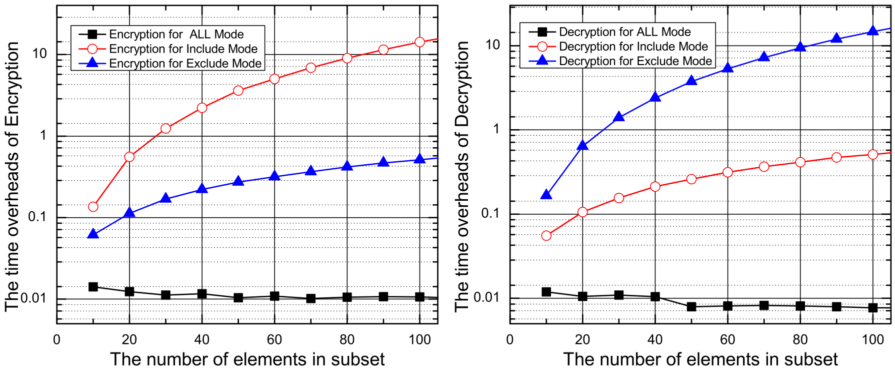

Figure 6: Time overheads of different modes.

Table 4: Time consumption of encryption and decryption under the different modes.

Size	Encryption			Decryption		
	ALL	Include	Exclude	ALL	Include	Exclude
20	0.01239	0.55603	0.11251	0.01056	0.10591	0.63935
40	0.01164	2.21912	0.22061	0.01046	0.21214	2.39276
60	0.0109	5.02113	0.3162	0.00815	0.31277	5.30531
80	0.01055	8.99909	0.418	0.0081	0.41469	9.40407
100	0.01066	14.19017	0.5107	0.00775	0.50979	14.6477

lines of code. At first, we tested the total overheads of encryption and decryption under the different modes. As seen from left of Figure 7, encryption and decryption in Include-mode has the same overhead as Exclude-mode. This result is the same with our theoretical analysis.

Next, we show time consumption of four functions, Setup, KeyGen, Encrypt and Decrypt, under different size of subsets (from 10 to 100) in right of Figure 7. In this experiment, we require two functions, Setup and GenKey, deal with all elements in a given subsets. Their overheads are proportional to the subset size but are still small. From this figure, the overhead of Setup or KeyGen is far less than that of encryption and decryption. In summary, our experiments show that our SBE scheme is simple, easy-to-implement, and high performance.

7. RELATED WORK

Fiat and Naor [1] were the first to formally explore broadcast encryption. They presented a private-key solution which was secure against a collusion of t users and has ciphertext size of $O(t\log^2 t\log n)$. Naor et al. [6] presented a fully collusion secure BE system that is efficient for broadcasting to all, but a small set of revoked users. However that systems do not support public-key encryption, the first public-key BE scheme was proposed by Dois et al. [9]. Boneh and Silverberg also show that n-linear maps give the ultimate *fully collusion secure* scheme with constant public key, private key, and ciphertext size. Soon after this, the bilinear maps became the basis for many subsequent proposals, including Delerablée et al. [5] proposed identity-based broadcast encryption and gave a selective CPA secure scheme. Existing broadcast encryption can be divided into two category:

One category is broadcast with multiple revoked receivers, meaning that we broadcast to all but a small set of revoked users $\mathcal{R}$ and $|\mathcal{R}| \ll n$. The best known systems are the scheme of Delerablée et al.[5] which achieves the optimal bound of $O(1)$-size either for ciphertexts or decryption keys for any subset of revoked users. More importantly, several dynamic behaviors, e.g., dynamic user joining, key updating, were supported by their BE systems.

Another category is broadcast for multiple designated receivers, where message is sent to a small subset of users $\mathcal{S}$, and $|\mathcal{S}| \ll n$. Untill now, the best known systems are the scheme of Boneh, Gentry and Waters [2] which achieves $O(\sqrt{n})$-size ciphertexts and public keys for any subset of receivers, where each user's private keys are of constant size. Their trivial scheme where both ciphertexts and private keys are of constant size and public key size is linear in total number of users is more efficient when $|\mathcal{S}| < O(\sqrt{n})$. However, their upper bound on the number of possible users n must be chosen at initialization time, so that new users cannot join dynamically the system. In summary, these work gave us many important inspirations for our research.

8. CONCLUSION

In this paper we present a new group-oriented cryptosystem, called set-based broadcast encryption. Our main contribution is to give the first cryptographic constructions for the decision problem of set membership and negative membership. for future work, we will improve the performance and apply our scheme into some practical applications, e.g., broadcasting, keyword searching, and voting.

9. REFERENCES

[1] A. Fiat and M Naor. Broadcast encryption. In *Advances in Cryptology (CRYPTO'93)*, volume 773 of LNCS, pages 480–491, 1994.

[2] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology (CRYPTO'2005)*, volume 3621 of LNCS, pages 258–275, 2005.

[3] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology (EUROCRYPT'2005)*, volume 3494 of LNCS, pages 440–456, 2005.

[4] Dan Boneh and M Franklin. An efficient public key traitor tracing scheme. In *Advances in Cryptology (CRYPTO'1999)*, volume 1666 of LNCS, pages 338–353, 1999.

[5] Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In

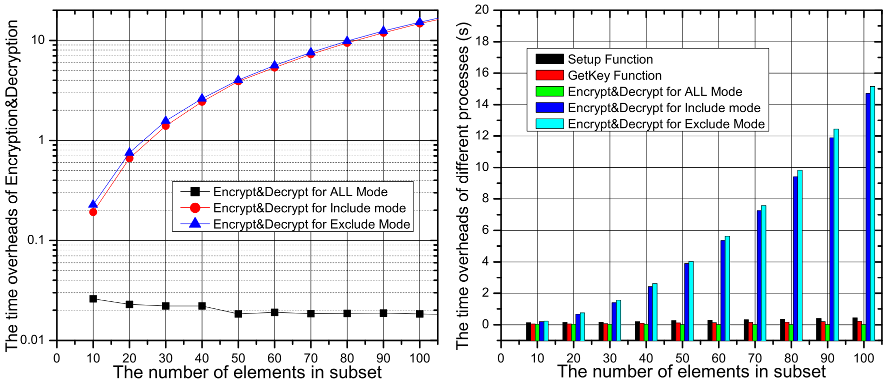

Figure 7: Time overhead of Algorithms in SBE.

Advances in Cryptology–ASIACRYPT 2007, pages 200–215. Springer, 2007.

[6] D Naor, M. Naor, and J Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology (Crypto'2001)*, volume 2139 of LNCS, pages 41–62, 2001.

[7] Duong Hieu Phan, David Pointcheval, Siamak Fayyaz Shahandashti, and Mario Strefler. Adaptive cca broadcast encryption with constant-size secret keys and ciphertexts. In *ACISP*, pages 308–321, 2012.

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

[9] Y. Dodis and N. Fazio. Public key broadcast encryption for stateless receives. In *Proceedings of the Digital Rights Management Workshop 2002*, volume 2696 of LNCS, pages 61–80, 2002.

[10] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology (CRYPTO'2001)*, volume 2139 of LNCS, pages 213–229, 2001.

[11] Jean-Luc Beuchat, Nicolas Brisebarre, Jérémie Detrey, and Eiji Okamoto. Arithmetic operators for pairing-based cryptography. In *CHES*, pages 239–255, 2007.

[12] Honggang Hu, Lei Hu, and Dengguo Feng. On a class of pseudorandom sequences from elliptic curves over finite fields. *IEEE Transactions on Information Theory*, 53(7):2598–2605, 2007.

[13] Bogdan Warinschi Gaven Watson Nigel P. Smart, Vincent Rijmen. Algorithms, key sizes and parameters report.

[14] Paulo S. L. M. Barreto, Steven D. Galbraith, Colm O'Eigeartaigh, and Michael Scott. Efficient pairing computation on supersingular abelian varieties. *Des. Codes Cryptography*, 42(3):239–271, 2007.

APPENDIX

A. GDHE ASSUMPTION

We give a rough overview of the General Diffie-Hellman Exponent (GDHE) assumption introduced by Boneh, Boyen and Goh [3] that will be used to analyze our schemes. Let $\mathbb{G}, \mathbb{G}_T$ be groups of order p and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a non-degenerate bilinear map.

Definition 5 (GDHE Problem). *Let $P, Q \in \mathbb{F}_p[X_1, \cdots, X_m]^s$ be two s-tuples of m-variate polynomials over $\mathbb{F}_p$ and generators $\hat{G}, \hat{H} \in \mathbb{G}_1$ and $\mathbb{G}_2$, where $s, m \in \mathbb{Z}^+$. We write $P = (p_1, \cdots, p_s)$ and $Q = (q_1, \cdots, q_s)$. Given a vector*

$$S = \left(\hat{G}^{P(x_1,\cdots,x_m)}, e(\hat{G}, \hat{H})^{Q(x_1,\cdots,x_m)} \right) \in \mathbb{G}^s \times \mathbb{G}_T^s$$

and $T \xleftarrow{R} \mathbb{G}_T$, decide whether $T = e(\hat{G}, \hat{H})^{h(x_1,\cdots,x_m)}$, where the polynomial $h \in \mathbb{F}_p[X_1, \cdots, X_m]$.

It is easy to find that almost previous decisional Diffie-Hellman assumptions can be reduced into GDHE assumption. We say that an algorithm $\mathcal{A}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving the GDHE problem if

$$Adv_{\mathrm{GDHE}}^{\mathrm{IND}}(\mathcal{A}) = \left| \begin{array}{c} \Pr[\mathcal{A}(S, e(\hat{G}, \hat{H})^{h(x_1,\cdots,x_m)}) = 0] \\ - \Pr[\mathcal{A}(S, T) = 0] \end{array} \right| > \epsilon.$$

The following theorem gives a lower bound on the advantage of a generic algorithm in solving the decision (P, Q, h)-Diffie-Hellman problem.

Theorem 4 ([3],Theorem A.2). *Let $P, Q \in \mathbb{F}_p[X_1, \cdots, X_m]^s$ be two s-tuples of m-variate polynomials over $\mathbb{F}_p$ and $h \in \mathbb{F}_p[X_1,$ $\cdots, X_m]$. Let $d = max(2d_P, d_Q, d_h)$, where d_P (resp. d_Q, d_h) denote the maximal degree of elements of P (resp. of Q, h). If h is **independent** of (P, Q) then for any $\mathcal{A}$ that makes a total of at most q queries to the oracles (computing the group operation in $\mathbb{G}, \mathbb{G}_T$ and the bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$), one has $Adv_{\mathrm{GDHE}}^{\mathrm{IND}}(\mathcal{A}) \leq \frac{(q+2s+2)^3 \cdot d}{2p}$.*

In this theorem, we define that a polynomial h is independent on the sets (P, Q) if there does not exist $s^2 + s$ constant $\{a_{ij}\}_{i,j=1}^s, \{b_k\}_{k=1}^s$, such that $h = \sum_{i,j=1}^s a_{ij} p_i p_j + \sum_{k=1}^s b_k q_k$.

B. PROOF OF THEOREM 1

Proof. We consider the weakest case $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ and thus pose $\hat{H} = \hat{G}^u$, where u is a random variate. In the (F_1, F_2, F_3, T)-GDHE$_1$ problem, if one replace γ by x, ϵ by v, and ς by y, we see that our problem is reformulated as (P, Q, h)-GDHE, where F_1, F_2 are integrated into P, F_3 and

T correspond to Q and h, respectively. Such that, we have

$$\begin{cases} P(x,y,u,v) & = & \begin{pmatrix} 1, v, xv, x^2v, \cdots, x^{t-1}v, \\ xf(x), \cdots, x^m f(x), \\ uv, uvx, \cdots, uvx^n, uf(x)g(x), \\ uvyf(x), uyf(x)g(x) \end{pmatrix} \\ Q(x,y,u,v) & = & (1, uvf^2(x)g(x)) \\ h(x,y,u,v) & = & uvyf^2(x)g(x) \end{cases}$$

where $d = 2n$, $m = 4$ and $s = t + m + n + 4$. According to the analysis method in [3], we next show that the polynomial h is independent of (P, Q), that is, no coefficients $\{a_{ij}\}_{i,j=1}^{s}, \{b_k\}_{k=1}^{s}$ exist such that $h = \sum_{i,j=1}^{s} a_{ij} p_i p_j + \sum_{l=1}^{s} b_l q_l$, where the polynomials p_i and q_i are the one listed in P and Q above. Considering that y only appears in $(uvyf(x), uyf(x)g(x))$, we must pick them out for all cases. A simple analysis method is to find the multiples of uvy from all possible products of two polynomials $(uvyf(x), uyf(x)g(x))$ in P. The polynomials $(uv, uvx, \cdots, uvx^n)$ can be excluded because $(uvyf(x), uyf(x)g(x))$ have contained u. Further, the polynomials $(v, xv, x^2v, \cdots, x^{t-1}v)$ also conflict with $uvyf(x)$ because they all contain v. So, we consider three cases:

- For the former $uvyf(x)$, we need to find the polynomial $h(x,y,u,v)/uvyf(x) = f(x)g(x)$. Let $g(x) = \prod_{i=1}^{n-t}(x + x'_i) = \sum_{i=0}^{n-t} b_i \cdot x^i$, where $b_0 = x'_1 x'_2 \cdots x'_{n-t} \neq 0$ because all $x'_i \neq 0$. So, $f(x)g(x)$ is represented as

  $$f(x)g(x) = \sum_{i=0}^{n-t} a_i(x^i f(x)) = b_0 f(x) + \sum_{i=1}^{n-t} b_i(x^i f(x)).$$

  but it is infeasible for computing it because there dose not exist the polynomial $f(x)$ in $P(x,y,u,v)$ even though we can get $(xf(x), \cdots, x^l f(x))$ from $P(x,y,u,v)$.
- For the latter $uyf(x)g(x)$, we need to find the polynomial $h(x,y,u,v)/uyf(x)g(x) = vf(x)$ and it is also infeasible because the polynomial $f(x) = \prod_{i=1}^{t}(x + x_i) = \sum_{i=0}^{t} a_i \cdot x^i = \sum_{i=0}^{t-1} a_i \cdot x^i + x^t$, which is a polynomial of t degree and $a_t = 1$, cannot be built from $(v, xv, \cdots, x^{t-1}v)$.
- For linear combination between $uvyf(x)$ and $uyf(x)g(x)$, assume that there exist two coefficients a, b to satisfy

  $$h(x,y,u,v) = a \cdot uvyf(x) \cdot p_1 + b \cdot uyf(x)g(x) \cdot p_2.$$

  So that, we have $vf(x)g(x) = avp_1 + bg(x)p_2$. To satisfy it, p_1 is merely derived from $(xf(x), \cdots, x^l f(x))$ and p_2 is from $(v, xv, \cdots, x^{t-1}v)$. This means $xf(x)|p_1$. Next we have $avp_1 = g(x)(vf(x) - bp_2)$, so we have $g(x)|p_1$. These two results show $xf(x)g(x)|p_1$. Since $p_1|g(x)(vf(x) - bp_2)$, we have $xf(x)|(vf(x) - bp_2)$. However, we know p_2 is a polynomial of x with at most degree $t - 1$, so $(vf(x) - bp_2)$ is a polynomial of x with degree t. This is a contradiction with the fact that $xf(x)$ has degree $t + 1$.

Hence, no linear combination among the polynomials from the known P, Q loads to h. This means that h is independent of (P, Q). Therefore, we obtain the advantage of the adversary $Adv_{\mathrm{GDHE}_1,\mathcal{A}}^{\mathrm{IND}}(n,t) \leq \frac{(q+2s+2)^3 \cdot d}{2p}$ according to Theorem (4). □

C. PROOF OF THEOREM 2

Proof. This proof is similar to the previous proof. We also consider the weakest case $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$. We see that our problem (F_1, F_2, F_3, T)-GDHE is reformulated as (P, Q, h)-GDHE, where the result after conversion is

$$\begin{cases} P(x,y,u,v) & = & \begin{pmatrix} 1, v, xv, x^2v, \cdots, x^{t-1}v, \\ xf(x), \cdots, x^m f(x), \\ uv, uvx, \cdots, uvx^n, uf(x)g(x), \\ yxf^2(x), uyf(x)g(x) \end{pmatrix} \\ Q(x,y,u,v) & = & (1, uvf^2(x)g(x)) \\ h(x,y,u,v) & = & uvyf^2(x)g(x) \end{cases}$$

where $d = 2n$, $m = 4$ and $s = n + t + m + 4$. We show that the polynomial h is independent of (P, Q). Considering that y only appears in $(yxf^2(x), uyf(x)g(x))$, we must pick them out for all cases. Next, there only exist the polynomials, $(uv, uvx, \cdots, uvx^n)$, which contain uv, so we have the candidate combination between $yxf^2(x)$ and these polynomials, but the combination between $uyf(x)g(x))$ and them will be excluded. Such that, we only need to consider three cases:

- For the former $(yxf^2(x)$, we need to find the polynomial $h(x,y,u,v)/yxf^2(x) = uvg(x)/x$, but it is infeasible for all $(uv, uvx, \cdots, uvx^n)$ because $g(x) = \prod_{i=1}^{n-t}(x + x'_i)$ is not divisible by x when all $x'_i \neq 0$ for $i = 1, \cdots, n - t$.
- For the latter $uyf(x)g(x))$, we need to find the polynomial $h(x,y,u,v)/uyf(x)g(x) = vf(x)$ and it is also infeasible because $f(x) = \prod_{i=1}^{t}(x + x_i) = \sum_{i=0}^{t} a_i x^i = x^t + \sum_{i=0}^{t-1} a_i x^i$, which is a polynomial of t degree, cannot be built from $(v, xv, \cdots, x^{t-1}v)$.
- For linear combination between $yxf^2(x)$ and $uyf(x)g(x)$, assume that there exist two coefficients a, b to satisfy

  $$h(x,y,u,v) = a(yxf^2(x))(uv\tfrac{g(x)}{x+x'_i}) + b(uyf(x)g(x))(\tfrac{vf(x)}{x+x_i}).$$

  So that, we have $\frac{ax}{x+x'_i} + \frac{b}{x+x_i} = 1$. To solve this equation, we have $(a-1)x^2 + (ax_i + b - x_i - x'_i)x + (bx'_i - x_i x'_i) = 0$. It is easy find that our solution is $a = 1$ and $b = x_i = x'_i$, but it contradicts with the assumption of $x_i \neq x'_i$ for all x_i and x'_i in $f(x)$ and $g(x)$.

Hence, no linear combination among the polynomials from P, Q loads to h. This also means that h is independent of (P, Q). Therefore, we obtain the advantage of adversary $Adv_{\mathrm{GDHE}_2,\mathcal{A}}^{\mathrm{IND}}(n,t) \leq \frac{(q+2s+2)^3 \cdot d}{2p}$ according to Theorem (4). □

D. PROOF OF THEOREM 3

Proof. Suppose there exists an adversary $\mathcal{A}$ can break the security of our SBE scheme with the advantage $Adv_{\mathrm{SBE},\mathcal{A}}^{\mathrm{IND},mode}(n,t)$ for either Include-mode or Exclude-mode. Our objective is to build a PPT algorithm $\mathcal{B}$ to solve the above (n,t)-GDHE$_1$ or (n,t)-GDHE$_2$ problem by using the advantage of $\mathcal{A}$. Since there merely exists a slight different between GDHE$_1$ and GDHE$_2$: $\hat{H}^{\varsigma\epsilon f(\gamma)}$ lies in GDHE$_1$ but $\hat{G}^{\varsigma\gamma f^2(\gamma)}$ in GDHE$_2$ (see Table 1), such that we merge them into one complete proof, as follows:

Initial. According to the assumption that $\mathcal{B}$ is given as input an (n,t)-GDHE$_1$ or (n,t)-GDHE$_2$ instance. From this instance, $\mathcal{B}$ does not know $\gamma, \varsigma, \epsilon$ but knows $2n$ random integers $x_i, x'_i, a_i, b_i \in \mathbb{Z}_p^*$ in $f(x)$ and $g(x)$, where any pairwise (x_i, x'_i) are not equal to each other. Firstly, $\mathcal{B}$ defines a set of identities $\mathcal{U}$, and then require $\mathcal{A}$

to choose randomly a challenge subset $\mathcal{S}$ from $\mathcal{U}$ such that $\mathcal{R} = \mathcal{U} \setminus \mathcal{S}$, where $|\mathcal{U}| = n$ and $\mathcal{R} = t$. Secondly, $\mathcal{B}$ formally sets $G = \hat{G}^{f(\gamma)}$ (but he can by no means compute the value of G) and $H = \hat{H}^{f(\gamma)g(\gamma)}$. In order to compute H, we first need to compute the polynomial coefficients c_i of $f(X) \cdot g(X) = \prod_{i=1}^{t}(X + x_i) \cdot \prod_{i=1}^{n-t}(X + x'_i) = \sum_{i=0}^{n} c_i \cdot X^i$, where all x_i and x'_i are known, and then $H = \prod_{i=0}^{n}(\hat{H}^{\epsilon\gamma^i})^{c_i}$ is computed from $\hat{H}^{\epsilon}, \hat{H}^{\gamma\epsilon}, \cdots, \hat{H}^{\gamma^n\epsilon}$. Based on them, it computes easily the public parameter:

$$mpk = \begin{cases} H &= \hat{H}^{f(\gamma)g(\gamma)} &= \prod_{i=0}^{n}(\hat{H}^{\epsilon\gamma^i})^{c_i}, \\ R &= e(G,H)^{\epsilon} &= e(\hat{G},\hat{H})^{\epsilon f^2(\gamma)g(\gamma)}, \\ G_k &= G^{\gamma^k} &= \hat{G}^{\gamma^k f(\gamma)} \quad \text{for } k = [1,m]. \end{cases}$$

Learning. In this phase, the adversary $\mathcal{A}$ can issue up to n times secret-key queries $\{\mathrm{ID}_i\}$ to gain the information of this cryptosystem. For each query ID_i, we consider two following cases: $\mathrm{ID} \in \mathcal{S}$ and $\mathrm{ID}_i \in \mathcal{R}$:

$\mathrm{ID}_i \in \mathcal{R}$: $\mathcal{B}$ chooses randomly an element x_i and let $x_i = hash(\mathrm{ID}_i)$. Then $\mathcal{B}$ defines the polynomial

$$f_i(X) = \frac{f(X)}{X + x_i} = \prod_{k=1,k\neq i}^{t}(X+x_i) = \sum_{k=0}^{t-1} a'_k X^k$$

of degree $t-1$ for $i \in [1,t]$. Based on the known values $(\hat{G}^{\epsilon}, \hat{G}^{\epsilon\gamma}, \cdots, \hat{G}^{\epsilon\gamma^{t-1}})$, $\mathcal{B}$ generates the secret key of the corrupted users

$$\begin{aligned} sk^{(i)} &= G^{\frac{x_i\epsilon}{\gamma+x_i}} = \hat{G}^{\epsilon\frac{x_i f(\gamma)}{\gamma+x_i}} = \hat{G}^{\epsilon x_i f_i(\gamma)} \\ &= \left(\hat{G}^{\epsilon\sum_{j=0}^{t-1} a'_j\gamma^j}\right)^{x_i} = \left(\prod_{j=0}^{t-1}\left(\hat{G}^{\epsilon\gamma^j}\right)^{a'_j}\right)^{x_i}. \end{aligned}$$

Similarly, $\mathcal{B}$ computes the coefficients b'_k of $f_i(X)g(X) = \sum_{k=1}^{n-1} b'_k X^k$ and uses $(\hat{H}^{\epsilon}, \hat{H}^{\epsilon\gamma}, \cdots, \hat{H}^{\epsilon\gamma^n})$ to compute

$$H_i = H^{\frac{\epsilon}{\gamma+x_i}} = \hat{H}^{\epsilon f_i(\gamma)g(\gamma)} = \prod_{k=0}^{n-1}(\hat{H}^{\gamma^k\epsilon})^{b'_k},$$

where b'_k is a known integer for all $k = [0, n-1]$. Finally, $\mathcal{B}$ sends $sk^{(i)}$ and $pp_i = (\mathrm{ID}_i, H_i)$ to $\mathcal{A}$. Note that, all these keys $\{sk^{(i)}\}_{\mathrm{ID}_i\in\mathcal{R}}$ are available for the ciphertext which is encrypted by the public encryption key.

$\mathrm{ID}_i \in \mathcal{S}$: $\mathcal{B}$ chooses randomly an element x'_i and let $x'_i = hash(\mathrm{ID}_i)$. $\mathcal{B}$ is merely required to use the above-mentioned approach to generate (ID_i, H_i), that is, $\mathcal{B}$ define $g_i(X) = \frac{g(X)}{X+x'_i}$ for $i \in [1, n-t]$ and computes $H_i = \hat{H}^{f(\gamma)g_i(\gamma)}$ in terms of $(\hat{H}^{\epsilon}, \hat{H}^{\epsilon\gamma}, \cdots, \hat{H}^{\epsilon\gamma^n})$. Finally, $\mathcal{B}$ sends $pp_i = (\mathrm{ID}_i, H_i)$ to $\mathcal{A}$.

Note that the total number of attribute values is n. H_i can be computed easily because the polynomials $f_i g$ and $f g_i$ are of degree $n-1$. In addition, $\mathcal{A}$ can query the hash value $x_i = h(\mathrm{ID}_i)$ at any time if $\mathrm{ID}_i \in \mathcal{R}$. Otherwise, $\mathcal{B}$ returns the hash value $x'_i = h(\mathrm{ID}_i)$.

Challenge. $\mathcal{B}$ defines $\varsigma = s$, but ς is unknown for $\mathcal{B}$. Next, according to the given challenge $mode^*$, $\mathcal{B}$ chooses one of two modes, $u \in \mathcal{S}$ and $u \notin \mathcal{S}$ to compute the challenge ciphertext from two given instances, GDHE_1 or GDHE_2, respectively. The ciphertext is generated as follows:

- $u \in \mathcal{S}$: $\mathcal{B}$ constructs a ciphertext as follows:

$$\begin{cases} C_1 &= H^s &= \hat{H}^{\varsigma f(\gamma)g(\gamma)}, \\ C_2 &= H^{\frac{s\epsilon}{g(\gamma)}} &= \hat{H}^{\varsigma\epsilon f(\gamma)}, \end{cases}$$

where $\mathcal{B}$ get these two values from GDHE_1 instance.

- $u \notin \mathcal{S}$: $\mathcal{B}$ constructs a ciphertext as follows:

$$\begin{cases} C_1 &= H^s &= \hat{H}^{\varsigma f(\gamma)g(\gamma)}, \\ C_2 &= G^{s\gamma f(\gamma)} &= \hat{G}^{\varsigma\gamma f^2(\gamma)}, \end{cases}$$

where $\mathcal{B}$ get these two values from GDHE_2 instance.

For both cases, $\mathcal{B}$ defines the session key ek as T. $\mathcal{B}$ now selects a random bit $b \leftarrow \{0,1\}$, sets $ek_b = T$ and sets ek_{1-b} to a random element of $\mathbb{G}_T$. Finally, $\mathcal{B}$ sends the challenge $\mathcal{C} = (C^*, ek_0, ek_1)$ to $\mathcal{A}$, where $C^* = (\mathcal{S}, mode^*, C_1, C_2)$.

Note that, as seen from Section 4.1, there exists a slight different between GDHE_1 and GDHE_2: $\hat{H}^{\varsigma\epsilon f(\gamma)}$ lies in GDHE_1 but $\hat{G}^{\varsigma\gamma f^2(\gamma)}$ in GDHE_2, which is responsible for the different of the above ciphertexts.

Guess. $\mathcal{A}$ returns a guess $b' \in \{0,1\}$ to $\mathcal{B}$. If $b = b'$, $\mathcal{B}$ outputs 1, otherwise 0.

We now analyze the validate of $\mathcal{B}$ as follows: if the given value T is valid, i.e., $T = e(\hat{G},\hat{H})^{\varsigma\epsilon f^2(\gamma)g(\gamma)}$, the challenge C^* and ek will be valid because

$$\begin{aligned} ek &= e(G,H)^{s\epsilon} = e(\hat{G}^{f(\gamma)}, \hat{H}^{f(\gamma)g(\gamma)})^{\varsigma\epsilon} \\ &= e(\hat{G},\hat{H})^{\varsigma\epsilon f^2(\gamma)g(\gamma)} = T. \end{aligned}$$

In this case, we have the success probability of $\mathcal{B}$ is

$$\begin{aligned} \Pr[b = \mathcal{A}(\mathcal{C}) | T = ek] &= Pr[b = b' | T = ek] \\ &= \Pr[b' = 1 | b = 1 \wedge T = ek] \cdot \Pr[b = 1] + \\ &\quad \Pr[b' = 0 | b = 0 \wedge T = ek] \cdot \Pr[b = 0] \\ &= \Pr[b' = 1 | b = 1 \wedge T = ek] \cdot \tfrac{1}{2} + \\ &\quad \Pr[b' = 0 | b = 0 \wedge T = ek] \cdot \tfrac{1}{2} \end{aligned}$$

Otherwise, the adversary $\mathcal{A}$ randomly guesses and picks an answer b' with 1/2 possibility, such that $\Pr[b = b' | T \xleftarrow{R} \mathbb{G}_T] = \Pr[b \neq b' | T \xleftarrow{R} \mathbb{G}_T] = 1/2$,. Based on them, we have

$$\begin{aligned} Adv^{\mathrm{IND}}_{\mathrm{GDHE}_{1,2},\mathcal{B}}(n,t) &= |\Pr[b = b' | T = ek] - \Pr[b = b' | T \xleftarrow{R} \mathbb{G}_T]| \\ &= |\Pr[b' = 1 | b = 1 \wedge T = ek] \cdot \tfrac{1}{2} + \\ &\quad \Pr[b' = 0 | b = 0 \wedge T = ek] \cdot \tfrac{1}{2} - \tfrac{1}{2}| \\ &= \frac{1}{2}\left| \begin{array}{c} \Pr[b' = 1 | b = 1 \wedge T = ek] - \\ \Pr[b' = 1 | b = 0 \wedge T = ek] \end{array} \right| \end{aligned}$$

Based on the definition of $Adv^{\mathrm{IND}}_{\mathrm{SBE},\mathcal{A}}(n,t)$ in Section 2, we have the following equation under the precondition of $T = ek$:

$$\begin{aligned} Adv^{\mathrm{IND},mode}_{\mathrm{SBE},\mathcal{A}}(n,t) &= |\Pr[b' = b] - 1/2| \\ &= \tfrac{1}{2}|\Pr[b' = b] - \Pr[b' \neq b]| \\ &= \tfrac{1}{2}|\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]|. \end{aligned}$$

Summing up, we get that $Adv^{\mathrm{IND}}_{\mathrm{GDHE}_{1,2},\mathcal{B}}(n,t) = Adv^{\mathrm{IND},mode}_{\mathrm{SBE},\mathcal{A}}(n,t)$ for two modes. According to Theorem 1 and 2, we have proved that $Adv^{\mathrm{IND}}_{\mathrm{GDHE}_{1,2},\mathcal{B}}(n,t) \leq \frac{(q+2s+2)^2\cdot d}{2p}$, where $s = n + t + m + 4$ and $d = 2n$. Thus, $Adv^{\mathrm{IND},mode}_{\mathrm{SBE},\mathcal{A}}(n,t)$ also satisfies the same negligible success probability. □