Secure and Efficient Construction of Broadcast Encryption with Dealership

Kamalesh Acharya and Ratna Dutta

Department of Mathematics
Indian Institute of Technology Kharagpur
Kharagpur-721302, India
kamaleshiitkgp@gmail.com, ratna@maths.iitkgp.ernet.in

Abstract. Broadcast encryption with dealership (BED) has been proposed to achieve more innovative and scalable business models for broadcast services. It has an extensive application future. However, designing secure BED is a challenging task. The only known BED construction so far is by Gritti et al. We aim to raise the profile of BED primitives which has not received much attention despite of its importance. This paper presents a *selectively* chosen plaintext attack (CPA) secure BED scheme supporting *maximum number of accountability* and *privacy* (hides the group of users from broadcaster). Our scheme is a key encapsulation mechanism and practically more efficient. It reduces the parameter sizes and computation cost compared to Gritti et al. More interestingly, the broadcaster does not need to rely on users to detect the dishonest dealer. We provide concrete security analysis of our design under reasonable assumptions.

Keywords: broadcast encryption with dealership, chosen plaintext attack, maximum number of accountability, privacy.

1 Introduction

The increasing interests in the wide application of e-commerce raises issues regarding unauthorised distributions and use of digital content. Broadcast encryption provides enhanced confidentiality in the setting of practical threats against content distribution systems. Broadcast encryption was formally introduced by Fiat and Naor [8] in 1994, followed by a vast literature in various flavours [1–7, 10, 13, 15].

Broadcast encryption with dealership (BED), introduced by Gritti et al. [11], is a promising cryptographic primitive which has been developed very recently. It has greatly facilitated with sufficiently fine grained business model in broadcast environment. The core concept in BED is to enable a dealer to select the set of subscribed users and publishing a group token together with a threshold value on the group size. A broadcaster implicitly verifies the size of the group utilizing

This paper has been accepted in Provable Security 2016 and will appear on LNCS.

the group token without knowing the group explicitly. The broadcaster aborts if the group size exceeds the threshold value, otherwise produces a ciphertext.

Designing BED is not trivial mainly due to the difficulty in achieving the following three security issues:

(i) *Maximum number of accountability:* Dealer should not be able to cheat. If a dishonest dealer selects $k' > k$ users and pays money for k users to the broadcaster, then the business of the broadcaster will be ruined.
(ii) *Privacy:* The dealer should be able to keep the subscribed user set secret from the broadcaster. Otherwise, the broadcaster can directly approach to the subscribers and damage the business of the dealer.
(iii) *Security against illegal users:* Illegal users (including dealers) should not be able to decrypt the encrypted digital content (ciphertext) similar to other broadcast encryption schemes.

Efficiency is always the first priority in obtaining practical BED. Low cost delivery of content is a major challenge in this context apart from achieving the aforementioned security attributes.

Interest in designing BED primitives is due to its applications in the real world. It could solve several problems of security and trust. For instance, suppose a dealer purchases the access of some encrypted digital contents from the service provider (broadcaster) in a bulk and resells them to the subscribers with a better price compared to the broadcaster's price for individual content. The subscribers thus enjoy the cheaper rate. The dealer keeps the identities of these subscribers secret from the broadcaster to protect his business. On the other hand, the dealer should be made incapable of decrypting the digital content to forbid him from rebroadcasting the content. In the light of the above application requirements, BED is useful.

So far as we know, BED has received very little attention despite of its numerous applications in the real world. Our goal is to develop this direction of research further by finding more practical and more efficient solutions towards BED. Principally, a BED makes the existing business model more flexible by creating new business opportunities for the dealers. A local dealer can better explore potentially unknown markets for service provider (broadcaster) and make a strategy according to the market. In addition, the dealer can also help in handling different pricing structures of media in different countries and share with the broadcaster any information on price or demand fluctuation cost. The dealer gets commission from the broadcaster and eventually sale of company increases.

Our contribution: Considering the limited development in the area of broadcast encryption in dealership framework, BED is further studied in this paper. The closest related work to ours is that of Gritti et al. [11]; indeed their work was starting point of ours. However, in the attempt made by [11], the broadcaster does not have the full control to detect illegal behaviour of a dealer as the components of the group token generated by the dealer are not fully binded. A dishonest dealer could easily manipulate some components of a group token

$P(G)$ in such a way that the implicit verification of the size of group G by the broadcaster succeeds without following the actual protocol. In fact, in Section 2.3 we elaborate this issue. The broadcaster has to release the encrypted content once the verification passes and rely on the response from the user side who has given the power to detect a dishonest dealer on completion of the protocol. This is not a good solution as user may be dishonest themselves, thereby hampering the broadcaster's interest. The construction of [11] is claimed to achieve unconditional privacy. Unfortunately, the argument in the security proof provided to support unconditional privacy allows illegal users to recover messages, thereby leading to a contradiction to semantic security in semi-static security model. We put more light on this in Section 2.3. We emphasize that in our scheme, the components of group token are skillfully formed to enable the broadcaster to have full control in detecting the dishonest behaviour of a dealer.

Our BED construction, namely KEMD, adapts key encapsulation mechanism and reduces the parameter sizes and computation cost over the existing scheme [11] significantly. Our construction based upon the identity based encryption scheme of Delerablee et al. [5]. The scheme provides computational *privacy* under the discrete logarithm problem. It is proven to achieve *key indistinguishability* under chosen plaintext attack (CPA) in selective model assuming the hardness of the (f, ϕ, F)-General Decisional Diffie-Hellman Exponent ((f, ϕ, F)-GDDHE) problem. Furthermore, it supports *maximum number of accountability* under the (f, N)-Diffie-Hellman Exponent assumption. In addition, if a user gets revoked from the system, he will be unable to decrypt the ciphertext similar to other broadcast encryption schemes. The dealer can select a new group of users without changing the existing public parameter and secret key.

Organization: The rest of the paper is organized as follows. Section 2 provides necessary definitions and background materials. We describe our main construction in Section 3 and its security in Section 4. Efficiency and comparison with the existing work is presented in Section 5. We finally conclude in Section 6.

2 Preliminaries

Notation: We use the notation $x \in_R S$ to denote x is a random element of S and λ to represent bit size of prime integer p. Also, we use $[m]$ to denote integers from 1 to m and $[a, b]$ to denote integers from a to b. Let $\epsilon : \mathbb{N} \to \mathbb{R}$ be a function, where $\mathbb{N}$ and $\mathbb{R}$ are the sets of natural and real number respectively. The function ϵ is said to be a *negligible function* if $\exists\ d \in \mathbb{N}$ such that $\epsilon(\lambda) \leq \frac{1}{\lambda^d}$. Let $|G|$ denotes the cardinality of group G.

2.1 Broadcast Encryption with Dealership

Syntax of KEMD: A key encapsulation mechanism with dealership scheme KEMD = (KEMD.Setup, KEMD.KeyGen, KEMD.GroupGen, KEMD.Verify, KEMD.Encrypt, KEMD.Decrypt) consists of four probabilistic polynomial time (PPT) algorithms - KEMD.Setup, KEMD.KeyGen, KEMD.GroupGen, KEMD.Encrypt and

two deterministic polynomial time algorithms - $\mathsf{KEMD.Verify}$, $\mathsf{KEMD.Decrypt}$. Formally, KEMD is described as follows:

- $(\mathsf{PP}, \mathsf{MK}) \leftarrow \mathsf{KEMD.Setup}(N, \lambda)$: The PKGC takes as input the total number of users N in the system and security parameter λ and constructs the public parameter PP and a master key MK. It makes PP public and keeps MK secret to itself.
- $(sk_i) \leftarrow \mathsf{KEMD.KeyGen}(\mathsf{PP}, \mathsf{MK}, i)$: Taking as input PP, MK and a subscribed user i, the PKGC generates a secret key sk_i of user i and sends sk_i to user i through a secure communication channel between PKGC and user i.
- $(P(G), k) \leftarrow \mathsf{KEMD.GroupGen}(\mathsf{PP}, G)$: The dealer selects a set of subscribed users G and generates a group token $P(G)$ using PP. It outputs a threshold value k, where $|G| \leq k$. The dealer sends G to each subscribed user $u \in G$ through a secure communication channel between them. Subscribed users keep G secret.
- $(0 \vee 1) \leftarrow \mathsf{KEMD.Verify}(P(G), \mathsf{PP}, k)$: The broadcaster verifies implicitly group size $|G| \leq k$ using $P(G)$, PP, k and sets
$$\mathsf{KEMD.Verify}(P(G), \mathsf{PP}, k) = \begin{cases} 1, & \text{if } |G| \leq k \\ 0, & \text{otherwise.} \end{cases}$$
If the verification fails i.e., $\mathsf{KEMD.Verify}(P(G), \mathsf{PP}, k) = 0$, the broadcaster aborts.
- $(\mathsf{Hdr}, K) \leftarrow \mathsf{KEMD.Encrypt}(P(G), \mathsf{PP})$: Taking as input $P(G)$ and PP, the broadcaster produces a header Hdr and a session key K. It makes the header Hdr public and keeps the session key K secret to itself. This session key K can be used to generate a ciphertext for a message using a symmetric key encryption algorithm.
- $(K) \leftarrow \mathsf{KEMD.Decrypt}(\mathsf{PP}, sk_i, \mathsf{Hdr}, G)$: A subscribed user i with secret key sk_i outputs the session key K using PP, Hdr and subscribed user set G.

Correctness: The scheme KEMD is said to be correct if the session key K can be retrieved from the header Hdr by any subscribed user in G. Suppose $(\mathsf{PP}, \mathsf{MK}) \leftarrow \mathsf{KEMD.Setup}(N, \lambda)$, $(P(G), k) \leftarrow \mathsf{KEMD.GroupGen}(\mathsf{PP}, G)$, $(\mathsf{Hdr}, K) \leftarrow \mathsf{KEMD.Encrypt}\big(P(G), \mathsf{PP}\big)$. Then for every subscribed user $i \in G$,
$$\mathsf{KEMD.Decrypt}\Big(\mathsf{PP}, \mathsf{KEMD.KeyGen}\big(\mathsf{PP}, \mathsf{MK}, i\big), \mathsf{Hdr}, G\Big) = K.$$

2.2 Security Framework

⟨I⟩ **Privacy:**

We define the privacy of the subscribed user set G of the protocol KEMD using the game as in Figure 1 between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. We have followed privacy model of [11].

Setup: The challenger $\mathcal{C}$ runs $\mathsf{KEMD.Setup}(N, \lambda)$ to generate the public parameter PP and master key MK. It sends PP to $\mathcal{A}$.

Challenge: The adversary $\mathcal{A}$ selects two sets of users G_0, G_1 of same size and submits G_0, G_1 to $\mathcal{C}$. The challenger $\mathcal{C}$ chooses $b \in_R \{0, 1\}$, generates a group token $P(G_b)$ by running $\mathsf{KEMD.GroupGen}(\mathsf{PP}, G_b)$ and sends $P(G_b)$ to $\mathcal{A}$.

Guess: The adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ of b and wins if $b'= b$.

Fig. 1: Privacy of protocol KEMD.

The advantage of the adversary $\mathcal{A}$ in the above privacy game is defined as $Adv_{\mathcal{A}}^{\mathsf{KEMD-P}} = |Pr(b^{'} = b) - \frac{1}{2}|$. The probability is taken over random bits used by $\mathcal{C}$ and $\mathcal{A}$.

Definition 1. *The BED scheme* KEMD *is said to be* (T, ϵ)*-secure under group privacy issue, if* $Adv_{\mathcal{A}}^{\mathsf{KEMD-P}} \leq \epsilon$ *for every PPT adversary* $\mathcal{A}$ *with running time at most* T*.*

⟨II⟩ **Maximum Number of Accountability:** The security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ addressing maximum number of accountability of the protocol KEMD follows the model in [11] and described in Figure 2.

Setup: The challenger $\mathcal{C}$ runs $\mathsf{KEMD.Setup}(N, \lambda)$ and generates public parameter PP and master key MK. It sends PP to $\mathcal{A}$.
Challenge: The challenger $\mathcal{C}$ sends an integer k to $\mathcal{A}$.
Guess: The adversary $\mathcal{A}$ computes $P(G^*)$, with $|G^*| > k$ by running $\mathsf{KEMD.GroupGen}(\mathsf{PP}, G^*)$ and sends $(P(G^*), G^*)$ to $\mathcal{C}$.
Win: The challenger $\mathcal{C}$ outputs $(P(G^*), G^*)$ if $\mathsf{KEMD.Verify}(P(G^*), \mathsf{PP}, k) = 1$; otherwise $\mathcal{C}$ aborts.

Fig. 2: Maximum number of accountability of protocol KEMD.

The adversary $\mathcal{A}$'s advantage in the above game for maximum number of accountability is defined as $Adv_{\mathcal{A}}^{\mathsf{KEMD\text{-}M}} = |\big(Pr(\mathsf{KEMD.Verify}(P(G^*), \mathsf{PP}, k)) = 1\big) - \frac{1}{2}|$ where $k < |G^*|$. The probability is taken over random bits used by $\mathcal{C}$ and $\mathcal{A}$.

Definition 2. *The BED scheme* KEMD *is said to be* (T, ϵ)*-secure under maximum number of accountability, if* $Adv_{\mathcal{A}}^{\mathsf{KEMD-M}} \leq \epsilon$ *for every PPT adversary* $\mathcal{A}$ *with running time at most* T*.*

⟨III⟩ **Key indistinguishability of KEMD under CPA**: We have followed [5] to design key indistinguishability against CPA security model. Selective security of the scheme KEMD is measured under the following key indistinguishability game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

Initialization: The adversary $\mathcal{A}$ selects a recipient set G and sends to $\mathcal{C}$.
Setup: The challenger $\mathcal{C}$ generates $(\mathsf{PP}, \mathsf{MK}) \leftarrow \mathsf{KEMD.Setup}(N, \lambda)$. It keeps the master key MK secret to itself and makes the public parameter PP public.

Phase 1: The adversary $\mathcal{A}$ sends key generation queries for $i_1, \ldots, i_m \notin G$ to $\mathcal{C}$ and receives the secret key $sk_i \leftarrow \mathsf{KEMD.KeyGen}(\mathsf{PP}, \mathsf{MK}, i)$ for user $i \in \{i_1, \ldots, i_m\}$.

Challenge: The challenger $\mathcal{C}$ generates $(\mathsf{Hdr}, K) \leftarrow \mathsf{Encrypt}\big(P(G), \mathsf{PP}\big)$, where $(P(G), k) \leftarrow \mathsf{KEMD.GroupGen}(\mathsf{PP}, G)$. It selects $b \in_R \{0,1\}$ and sets $K_b = K$, K_{1-b} a random value. Finally, $\mathcal{C}$ returns Hdr, K_0, K_1 to $\mathcal{A}$.

Phase 2: This is similar to Phase 1 key generation queries. The adversary $\mathcal{A}$ sends key generation queries for $i_{m+1}, \ldots, i_q \notin G$ to $\mathcal{C}$ and receives the secret key $sk_i \leftarrow \mathsf{KEMD.KeyGen}(\mathsf{PP}, \mathsf{MK}, i)$ for $i \in \{i_{m+1}, \ldots, i_q\}$.

Guess: The adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ of b and wins if $b' = b$.

Let t be the number of corrupted users and N be the total number of users. Adversary is allowed to get reply up to t key generation queries. In random oracle model t is number of hash queries and key generation queries. The adversary $\mathcal{A}$'s advantage in the above security game is defined as $Adv_{\mathcal{A}}^{\mathsf{KEMD-INDK}}(t,N) = |2Pr(b'=b)-1| = |Pr[b'=1|b=1] - Pr[b'=1|b=0]|$.

The probability is taken over random bits used by $\mathcal{C}$ and $\mathcal{A}$.

Definition 3. *Let* $Adv^{\mathsf{KEMD-INDK}}(t,N) = \max_{\mathcal{A}} \big[Adv_{\mathcal{A}}^{\mathsf{KEMD-INDK}}(t,N)\big]$, *where maximum is taken over all PPT algorithm running in* $poly(\lambda)$ *(polynomial of* λ*) time. The BED scheme* KEMD *is said to be* (t,N)*- secure if* $Adv^{\mathsf{KEMD-INDK}} = \epsilon(\lambda)$, *where* $\epsilon(\lambda)$ *is a negligible function in security parameter* λ.

2.3 The Drawbacks of [11]

We provide the overview of the BED construction of Gritti et al. [11] in Appendix A. In BED scheme of [11], the dealer generates the group token as

$$P(G) = \Big(w_1, w_2, w_3, w_4, w_5, w_6\Big)$$
$$= \Big(u_0^{t_1 \prod_{i \in G}(x_i+\alpha)}, v_0^{t_1 \prod_{i \in G}(x_i+\alpha)}, v_{N-k}^{t_1 \prod_{i \in G}(x_i+\alpha)}, \prod_{i \in G} f_i^{t_2}, g^{t_2}, e(g^{\gamma}, g)^{t_2}\Big).$$

Here $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ is bilinear mapping from source group $\mathbb{G}$ with generator g to target group $\mathbb{G}_1$, $u_i = h^{\gamma \alpha^i}, v_i = h^{\gamma \beta \alpha^i}$ for $i \in [0, N]$, $\alpha, \beta, \gamma, t_1, t_2 \in_R \mathbb{Z}_p$, $h \in_R \mathbb{G}$, public key of user i is $PK_i = (x_i + \alpha, f_i)$, $x_i \in_R \mathbb{Z}_p$, $f_i \in_R \mathbb{G}$, the group $G = \{i_1, i_2, \ldots, i_{k'}\} \in (\mathbb{Z}_p)^{k'}, k' \leq k$. The broadcaster verifies whether group size is $\leq k$ by checking $e(w_2, g_N) = e(w_3, g_k)$. It generates a ciphertext for message $M \in \mathbb{G}_1$ as $(w_5^r, w_4^r, M w_6^r)$ where $r \in_R \mathbb{Z}_p$. Note that the broadcaster does not involve w_1, w_2, w_3 in ciphertext components. A dishonest dealer can generate w_1, w_2, w_3 for less than k users while creating w_4, w_5, w_6 for greater than k users. In decryption phase, a user checks the group size that is received from the dealer during group token generation. If it is greater than k, then the user informs this to the broadcaster. The dealer will be blacklisted and excluded from further business. Consequently, the broadcaster does not have the full control on determining the dishonest dealer and has to rely on user's response to stop release of further encrypted content.

In the *privacy* proof, Gritti et al. [11] argued that group privacy is preserved unconditionally since for each group of receivers G, there is a group G' of same size such that $P(G) = P(G')$. This argument in fact incorrect. It is not sufficient to show that there exists a group G', since the adversary is allowed to choose G_0 and G_1. It is required to prove that $P(G_b) = P(G_{1-b})$, $b \in \{0, 1\}$ for a group G_b. They have proved $P(G_b) = P(G_{b'})$ where $G_{b'}$ may not be equal to G_{1-b}. If unconditional privacy holds, then $P(G) = P(G')$ for all pairs of groups of same size with $G \neq G'$. Then the members of G' would also be able to decrypt the ciphertext generated using $P(G)$ as $P(G) = P(G')$. But if G is the set of legal users, then a user in $G' \setminus G$ is not entitled to decrypt the ciphertext using $P(G)$. This contradicts the semantic security against illegal users.

2.4 Complexity Assumptions

Definition 4. ***(Bilinear Map).*** *Let $\mathbb{G}$ and $\mathbb{G}_1$ be two multiplicative groups of prime order p. Let g be a generator of $\mathbb{G}$. A bilinear map $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_1$ is a function having the following properties:*

1. *$e(u^a, v^b) = e(u, v)^{ab}$, $\forall\ u, v \in \mathbb{G}$ and $\forall\ a, b \in \mathbb{Z}_p$.*
2. *The map is non-degenerate, i.e., $e(g, g)$ is a generator of $\mathbb{G}_1$.*

The tuple $\mathbb{S} = (p, \mathbb{G}, \mathbb{G}_1, e)$ is called a prime order bilinear group system.

⟨A⟩ The Discrete Logarithm (DL) Assumption:
Input: $\langle Z = (g^\alpha, g) \rangle$, where g is a generator of $\mathbb{G}$, $\alpha \in_R \mathbb{Z}_p$.
Output: α

Definition 5. *The (T, ϵ)-DL assumption holds if for every PPT adversary $\mathcal{A}$ with running time at most T, the advantage of solving the above problem is at most ϵ, i.e., $Adv_{\mathcal{A}}^{\mathsf{DL}} = |Pr[\mathcal{A}(Z) = \alpha]| \leq \epsilon(\lambda)$, where $\epsilon(\lambda)$ is a negligible function in security parameter λ.*

⟨B⟩ The (f, l)-Diffie-Hellman Exponent ((f, l)-DHE) Assumption [11]:

Input: $\langle Z = (\mathbb{S}, g, g^\alpha, \ldots, g^{\alpha^l}) \rangle$, where g is generator of $\mathbb{G}$, $\alpha \in_R \mathbb{Z}_p$.
Output: $f(x)$ and $g^{f(\alpha)}$ where $f(x)$ is polynomial of degree $l' > l$.

Definition 6. *The (f, l)-DHE assumption holds with (T, ϵ) if for every PPT adversary $\mathcal{A}$ with running time at most T, the advantage of solving the above problem is at most ϵ, i.e., $Adv_{\mathcal{A}}^{(f,l)-\mathsf{DHE}} = |Pr[\mathcal{A}(Z) = (f(x), g^{f(\alpha)})]| \leq \epsilon(\lambda)$, where $\epsilon(\lambda)$ is a negligible function in security parameter λ and $f(x)$ is polynomial of degree $l' > l$.*

⟨C⟩ The (f, ϕ, F)-General Decisional Diffie-Hellman Exponent ((f, ϕ, F)-GDDHE) Assumption [5]:

Input: $\langle Z = (\mathbb{S}, f(x), \phi(x), h_0, h_0^\alpha, h_0^{\alpha^2}, \ldots, h_0^{\alpha^{t-1}}, h_0^{\alpha f(\alpha)}, h_0^{k\alpha f(\alpha)}, g_0, g_0^\alpha, g_0^{\alpha^2}, \ldots, g_0^{\alpha^{2N}}, g_0^{k\phi(\alpha)}), K \rangle$, where g_0, h_0 are generators of $\mathbb{G}, \alpha \in_R \mathbb{Z}_p, f(x) =$

$\prod_{i=1}^{t}(x+x_i)$, $\phi(x) = \prod_{i=t+1}^{t+N}(x+x_i)$, $x_i \in \mathbb{Z}_p$ for $i \in [t+N]$ are distinct, K is either $e(g_0,h_0)^{F(\alpha)}$ where $F(\alpha) = kf(\alpha)$ or a random element $X \in \mathbb{G}_1$.
Output: Yes if $K = e(g_0,h_0)^{kf(\alpha)}$; No otherwise.

Definition 7. *The* (f,ϕ,F)-GDDHE *assumption holds with* (T,ϵ) *if for every PPT adversary* $\mathcal{A}$ *with running time at most* T*, the advantage of solving the above problem is at most* ϵ*, i.e.,* $Adv_{\mathcal{A}}^{(f,\phi,F)-\mathsf{GDDHE}} = |Pr[\mathcal{A}(Z, K = e(g_0,h_0)^{kf(\alpha)}) = 1] - Pr[\mathcal{A}(Z, K = X) = 1]| \leq \epsilon(\lambda)$*, where* $\epsilon(\lambda)$ *is a negligible function in security parameter* λ *and* X *is random element of* $\mathbb{G}_1$*.*

3 Our KEMD Construction

Our key encapsulation mechanism with dealership KEMD = (KEMD.Setup, KEMD. KeyGen, KEMD. GroupGen, KEMD.Verify, KEMD.Encrypt, KEMD.Decrypt) is described as follows:

- $(\mathsf{PP},\mathsf{MK}) \leftarrow \mathsf{KEMD.Setup}(N,\lambda)$: Given the security parameter λ and public identity $\mathsf{ID} = \{ID_1, ID_2, \ldots, ID_N\} \in (\mathbb{Z}^+)^N$ of a group of N users, the PKGC generates the public parameter PP and a master key MK as follows:
  1. Chooses a prime order bilinear group system $\mathbb{S} = (p, \mathbb{G}, \mathbb{G}_1, e)$, where $\mathbb{G}, \mathbb{G}_1$ are groups of prime order p and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ is a bilinear mapping. Let g, h be generators of group $\mathbb{G}$ and $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ be a cryptographically secure hash function.
  2. Selects $\alpha \in_R \mathbb{Z}_p$ and sets a master key MK and public parameter PP as $\mathsf{MK} = (\alpha, h), \mathsf{PP} = (\mathbb{S}, g, g_1, \ldots, g_N, v = e(g,h), w = h^{\alpha}, H, \mathsf{ID})$, where $g_i = g^{\alpha^i}$ for $i \in [1, N]$.
  3. Keeps MK secret to itself and makes PP public.

  Note that the public identity of the user i is $ID_i \in \mathbb{Z}^+$ for $i \in [N]$.
- $(sk_u) \leftarrow \mathsf{KEMD.KeyGen}(\mathsf{PP},\mathsf{MK},u)$: For each user $u \in [N]$, the PKGC extracts α, h from MK and ID_u from PP, generates a secret key as $sk_u = h^{\frac{1}{\alpha+H(ID_u)}}$ and sends it to user u through a secure communication channel between them.
- $(P(G),k) \leftarrow \mathsf{KEMD.GroupGen}(\mathsf{PP},G)$: The dealer selects a group of users $G = \{i_1, i_2, \ldots, i_{k'}\} \subseteq [N]$ and performs the following using PP:
  1. Sets a polynomial $F(x) = \prod_{i_j \in G}(x + H(ID_{i_j})) = \sum_{i=0}^{k'} F_i x^i$, where F_i's are function of $H(ID_j)$ for $j \in G$.
  2. Picks $t_1 \in_R \mathbb{Z}_p$ and generates the group token $P(G) = (w_1, w_2, w_3, w_4)$ by setting

$$w_1 = w^{-t_1} = h^{-\alpha t_1}, \quad w_2 = \prod_{i=0}^{k'} g_{N-k+i}^{t_1 F_i} = g_{N-k}^{\sum_{i=0}^{k'} t_1 \alpha^i F_i} = g_{N-k}^{t_1 F(\alpha)},$$

$$w_3 = g^{t_1 F_0} \prod_{i=1}^{k'} g_i^{t_1 F_i} = g^{\sum_{i=0}^{k'} t_1 \alpha^i F_i} = g^{t_1 F(\alpha)}, \quad w_4 = v^{t_1} = e(g,h)^{t_1}$$

  where w, g_i, for $i \in [1, k']$ and v are extracted from PP.

3. Selects a threshold value k on the group size G where $k \geq k' = |G|$.
4. Sends G to each subscribed user through a secure communication channel between the user and the dealer. The subscribed users keep G secret to themselves.
5. Publishes $P(G)$ together with the threshold value k.

- $(0 \vee 1) \leftarrow \mathsf{KEMD.Verify}(P(G), \mathsf{PP}, k)$: Taking as input the group token $P(G) = (w_1, w_2, w_3, w_4)$, the threshold value k, and g_k, g_N extracted from PP, the broadcaster sets $\mathsf{KEMD.Verify}(P(G), \mathsf{PP}, k) = \begin{cases} 1, & \text{if } e(w_2, g_k) = e(w_3, g_N) \\ 0, & \text{otherwise.} \end{cases}$

$$\text{Notice that, } e(w_2, g_k) = e(g_{N-k}^{t_1 F(\alpha)}, g_k) = e\Big(\prod_{i=0}^{k'} g^{(t_1 \alpha^{N-k+i} \cdot F_i)}, g^{\alpha^k}\Big)$$
$$= e(g,g)^{t_1 \alpha^k \left(\sum_{i=0}^{k'} \alpha^{N-k+i} \cdot F_i\right)} = e(g,g)^{t_1 \left(\sum_{i=0}^{k'} \alpha^{N+i} \cdot F_i\right)},$$
$$\text{and, } e(w_3, g_N) = e(g^{t_1 F(\alpha)}, g_N) = e(g,g)^{t_1 \left(\sum_{i=0}^{k'} \alpha^{N+i} \cdot F_i\right)}.$$

If the verification fails i.e., $\mathsf{KEMD.Verify}(P(G), \mathsf{PP}, k) = 0$, the broadcaster aborts. We point down here that only two components namely w_2, w_3 of $P(G)$ are used during this verification process.

- $(\mathsf{Hdr}, K) \leftarrow \mathsf{KEMD.Encrypt}(P(G), \mathsf{PP})$: Using PP and $P(G) = (w_1, w_2, w_3, w_4)$ with $\mathsf{KEMD.Verify}(P(G), \mathsf{PP}, k) = 1$, the broadcaster does the following:
  1. Chooses an integer $r \in_R \mathbb{Z}_p$ and sets a session key K, header Hdr as

$$K = w_4^r = e(g,h)^{t_1 r}, \mathsf{Hdr} = (C_1, C_2) = \Big(w_1^r, w_3^r\Big) = \Big(h^{-\alpha r t_1}, g^{r t_1 F(\alpha)}\Big).$$

  2. Finally, publishes Hdr and keeps K secret to itself.

  Note that this encryption process utilizes the two components w_1, w_4 of $P(G)$, together with w_3 which has already been used in combination with w_2 and passed the verification in procedure $\mathsf{KEMD.Verify}$ successfully.

- $(K) \leftarrow \mathsf{KEMD.Decrypt}(\mathsf{PP}, sk_u, \mathsf{Hdr}, G)$: A subscribed user u with secret key sk_u, uses PP, the header $\mathsf{Hdr} = (C_1, C_2)$, the set of subscribed users G and recovers the session key K as $K = \Big(e(C_1, g^{P_{u,G(\alpha)}}) e(sk_u, C_2)\Big)^{\frac{1}{\prod_{j \in G, j \neq u} H(ID_j)}}$ where $P_{u,G(\alpha)} = \frac{1}{\alpha}\Big\{\prod_{j \in G, j \neq u} (\alpha + H(ID_j)) - \prod_{j \in G, j \neq u} H(ID_j)\Big\}$.

  Observe that $g^{P_{u,G(\alpha)}}$ is computable with the knowledge of G as follows: The expression $\Big\{\prod_{j \in G, j \neq u} (\alpha + H(ID_j)) - \prod_{j \in G, j \neq u} H(ID_j)\Big\}$ is a polynomial of degree $(k'-1)$ in α without a constant term where $k' = |G|$ and thus the expression $\frac{1}{\alpha}\Big\{\prod_{j \in G, j \neq u} (\alpha + H(ID_j)) - \prod_{j \in G, j \neq u} H(ID_j)\Big\} = \sum_{i=0}^{k'-2} a_i \alpha^i$ is a polynomial of degree $(k'-2)$ in α. Here a_i, $i \in [0, k'-2]$ are constants and are functions of $H(ID_j)$ where $j \in G, j \neq u$. Since $g, g_i = g^{\alpha^i}$ for $i \in [1, k'-2]$

are all available in public parameter PP,

$$g^{P_{u,G(\alpha)}} = g^{\sum_{i=0}^{k'-2} a_i \alpha^i} = g^{a_0} \prod_{i=1}^{k'-2} g^{a_i \alpha^i} = g^{a_0} \prod_{i=1}^{k'-2} g_i^{a_i}$$

can be computed without the knowledge of α. However, this requires explicit knowledge of group G, which is intimated to each subscriber by the dealer during token generation in the procedure $\mathsf{KEMD.GroupGen}$ through a secure communication channel between them.

Correctness of our KEMD: The correctness of $\mathsf{KEMD.Decrypt}$ algorithm is as follows:

$$\begin{aligned}
K &= \Big[e(C_1, g^{P_{u,G(\alpha)}}) e(sk_u, C_2)\Big]^{\frac{1}{\prod_{j\in G, j\neq u} H(ID_j)}} \\
&= \Big[e\Big(h^{-\alpha r t_1}, g^{\frac{1}{\alpha}\left\{\prod_{j\in G, j\neq u}(\alpha + H(ID_j)) - \prod_{j\in G, j\neq u} H(ID_j)\right\}}\Big) \times \\
&\qquad e\Big(h^{\frac{1}{\alpha + H(ID_u)}}, g^{r t_1 \prod_{j\in G}(\alpha + H(ID_j))}\Big)\Big]^{\frac{1}{\prod_{j\in G, j\neq u} H(ID_j)}} \\
&= \Big[e(g,h)^{-r t_1 \left\{\prod_{j\in G, j\neq u}(\alpha + H(ID_j)) - \prod_{j\in G, j\neq u} H(ID_j)\right\}} \times \\
&\qquad e(h,g)^{r t_1 \left\{\prod_{j\in G, j\neq u}(\alpha + H(ID_j))\right\}}\Big]^{\frac{1}{\prod_{j\in G, j\neq u} H(ID_j)}} \\
&= e(g,h)^{t_1 r}.
\end{aligned}$$

Remark 1. If a user revokes then the selected user set G will be changed. Accordingly $P(G)$ will be changed. Moreover, a revoked user will not have the information about current subscribed users. Therefore he will unable to recover the session key.

Remark 2. In our scheme dealer can not act dishonestly as we use all the components of our group token

$$P(G) = \Big(w_1, w_2, w_3, w_4\Big) = \Big(w^{-t_1}, g_{N-k}^{t_1 F(\alpha)}, g^{t_1 F(\alpha)}, e(g,h)^{t_1}\Big)$$

either implicitly or explicitly in encryption phase. This property is not achievable in [11].

Remark 3. Note that the decryptor (legitimate subscribed user) needs the explicit knowledge of subscribed users in the decryption procedure. The dealer uses secure communication channel to inform the subscribed user set G while generating the group token $P(G)$. The dealer has to use these secure channels between him and the subscribed user each time a new group token is generated on group membership change. For dynamic group, it is essential to remove the reuse of secure communication channel which can be done by using a suitable public key encryption as follows: The dealer generates (public key, secret key) pair (p_i, s_i) for each user $i \in [N]$ during the procedure $\mathsf{KEMD.Setup}$ using a public key encryption mechanism and gives s_i to user i securely. Let at some stage, $j_1, \ldots, j_{k'} \in [N]$ are subscribed users with identities $ID_{j_1}, \ldots, ID_{j_{k'}}$. To represent a user index,

we need $s = \log_2 N$ bits for a network with maximum N users. Let message space of the public key encryption scheme $\mathcal{E}$ be at least $(N+2)s$ bits. The dealer generates ciphertext $y = \left(\left[\mathcal{E}_{p_i}(j_1||\ldots||j_{k'}||k'||X)\right]_{i=1}^{k'}, \left[\mathcal{E}_{\hat{p}_i}(R_i)\right]_{i=1}^{k-k'}, X\right)$ of size $k+1$ while generating group token in the procedure KEMD.GroupGen. Here R_i are random messages, $\hat{p}_i$ are random key values for $i \in [1, k-k']$, $||$ denotes concatenation of bits. Consider $j_1, \ldots, j_{k'}, k', X$ are of s bits. If it is not of s bits, fill up left part by zeros. Last s bits are parity checking bits. The dealer publishes y instead of sending the group G to the subscribed users through secure communication channels. User i decrypts the ciphertext components using the secret key s_i. If it finds a decrypted value whose last s bits matches with X, then it can extract $j_1, \ldots, j_{k'}$ from the decrypted value.

4 Security

Theorem 1. *(Privacy). Our proposed BED scheme KEMD described in Section 3 is computationally secure under the hardness of the discrete logarithm problem as per the group privacy issue as described in Figure 1 in Section 2.2.*

Proof. We describe the privacy of KEMD using a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ as:

Setup: The challenger $\mathcal{C}$ generates the public parameter, $\mathsf{PP} = (\mathbb{S}, g, g_1, \ldots, g_N, v = e(g,h), w = h^{\alpha}, H, \mathsf{ID})$, and the master key $\mathsf{MK} = (\alpha, h)$ by calling $\mathsf{KEMD.Setup}(N, \lambda)$. Here $g_i = g^{\alpha^i}$ for $i \in [1, N]$, $\alpha \in \mathbb{Z}_p$, g, h are generators of group $\mathbb{G}$, $\mathsf{ID} = \{ID_1, ID_2, \ldots, ID_N\} \in (\mathbb{Z}^+)^N$ is the set of public identities of N users, $H : \{0,1\}^* \to \mathbb{Z}_p^*$ is a cryptographically secure hash function. It keeps MK secret to itself and hands PP to $\mathcal{A}$.

Challenge: The adversary $\mathcal{A}$ selects two sets of users G_0, G_1 of same size and submits G_0, G_1 to $\mathcal{C}$. The challenger $\mathcal{C}$ chooses $b \in_R \{0,1\}$ and generates a group token $P(G_b)$ by running $\mathsf{KEMD.GroupGen}(\mathsf{PP}, G_b)$ as

$$P(G_b) = (w_1, w_2, w_3, w_4) = (w^{-t_1}, \prod_{i=0}^{k'} g_{N-k+i}^{t_1 F_i}, \prod_{i=0}^{k'} g_i^{t_1 F_i}, v^{t_1})$$
$$= (h^{-\alpha t_1}, g_{N-k}^{t_1 F(\alpha)}, g^{t_1 F(\alpha)}, e(g,h)^{t_1})$$

where $t_1 \in \mathbb{Z}_p$, F_i, $0 \le i \le k'$ are coefficient of x^i in polynomial $F(x) = \prod_{j \in G_b}(x + H(ID_j))$. The challenger $\mathcal{C}$ hands $P(G_b)$ to $\mathcal{A}$.

Guess: The adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ of b and wins if $b' = b$.

Given $P(G_b)$, the adversary $\mathcal{A}$ can predict G_b if it can predict the random number t_1 chosen by the challenger $\mathcal{C}$. As $\mathcal{A}$ has G_0, G_1, he can compute $P(G_0)$ if he can know t_1. If $P(G_0)$ matches with $P(G_b)$, $\mathcal{A}$ predicts $b = 0$, else $b = 1$. Therefore, prediction of b is same as predicting t_1 from $P(G_b)$ i.e., computing t_1 from $w_1 = w^{-t_1}$ where w is available to $\mathcal{A}$ trough PP. So, security depends on the hardness of the discrete logarithm problem. Hence the theorem.

Theorem 2. *(Maximum number of accountability). Our proposed BED scheme KEMD described in Section 3 is secure as per maximum number of accountability security model as described in Figure 2 in Section 2.2 under the (f, N)-DHE hardness assumption.*

Proof. Let a PPT adversary $\mathcal{A}$ breaks the maximum number of accountability of our KEMD scheme with non-negligible advantage. We construct an algorithm $\mathcal{C}$ that attempts to solve an instance of the (f, N)-DHE problem using $\mathcal{A}$ as a sub-routine.

$\mathcal{C}$ is given an instance of the (f, N)-DHE problem $\big\langle Z = (\mathbb{S}, g, g_1, g_2, \ldots, g_N)\big\rangle$, where $g_i = g^{\alpha^i}$ for $i \in [N]$, $\alpha \in \mathbb{Z}_p$, $\mathbb{S}$ is a bilinear group system, g is a generator of the group $\mathbb{G}$. Now $\mathcal{C}$ plays the role of the challenger in the security game and interacts with $\mathcal{A}$ as follows:

Setup: Using Z, the challenger $\mathcal{C}$ sets public parameter $\mathsf{PP} = (\mathbb{S}, g, g_1, \ldots, g_N, v = e(g, g^x), w = g_1^x, H, \mathsf{ID})$ where $x \in_R \mathbb{Z}_p$, $H : \{0,1\}^* \to \mathbb{Z}_p^*$ is a cryptographically secure hash function, $\mathsf{ID} = \{ID_1, ID_2, \ldots, ID_N\} \in (\mathbb{Z}^+)^N$ is the set of public identities of N users and hands PP to $\mathcal{A}$. It sets $\mathsf{MK} = (\alpha, h = g^x)$. Note that α is not known to $\mathcal{C}$ explicitly and $w = g_1^x = g^{\alpha x} = h^\alpha, v = e(g, g^x) = e(g, h)$ as in the real scheme.

Challenge: The challenger $\mathcal{C}$ submits a threshold value $k \in [N]$ on the group size to $\mathcal{A}$.

Guess: The adversary $\mathcal{A}$ computes $P(G^*)$ by running $\mathsf{KEMD.GroupGen}(\mathsf{PP}, G^*)$ where $|G^*| = \hat{k} > k$ as

$$P(G^*) = (\hat{w}_1, \hat{w}_2, \hat{w}_3, \hat{w}_4) = (w^{-t_1}, \prod_{i=0}^{\hat{k}} g_{N-k+i}^{t_1 F_i}, \prod_{i=0}^{\hat{k}} g_i^{t_1 F_i}, v^{t_1})$$
$$= (h^{-\alpha t_1}, g_{N-k}^{t_1 \hat{F}(\alpha)}, g^{t_1 \hat{F}(\alpha)}, e(g,h)^{t_1})$$

where $t_1 \in \mathbb{Z}_p$, F_i, $0 \le i \le \hat{k}$ are coefficient of x^i in polynomial $\hat{F}(x) = \prod_{j \in G^*}(x + H(ID_j))$. The adversary $\mathcal{A}$ sends $(P(G^*), G^*)$ to $\mathcal{C}$.

Note that if the adversary $\mathcal{A}$ outputs a valid $P(G^*)$ for a group G^* of size $\hat{k} > k$ i.e., $\mathsf{KEMD.Verify}(P(G^*), \mathsf{PP}, k) = 1$, then $\hat{F}(x) = \prod_{j \in G^*}(x + H(ID_j))$ is a $\hat{k}(> k)$ degree polynomial and $\hat{w}_2 = g_{N-k}^{t_1 \hat{F}(\alpha)} = g^{t_1 \alpha^{N-k} \hat{F}(\alpha)}$. Let $f(x) = t_1 x^{N-k} \hat{F}(x)$. This is a polynomial of degree $N - k + \hat{k} > N$ as $\hat{k} > k$. Then $(f(x), \hat{w}_2 = g^{f(\alpha)})$ is a solution of the (f, N)-DHE problem. Therefore if $\mathcal{A}$ wins against maximum number of accountability game in Figure 2, then it can solve the (f, N)-DHE problem. This completes the proof.

Theorem 3. *(Key indistinguishability under CPA) Our proposed BED scheme KEMD described in Section 3 achieves selective semantic (indistinguishable under CPA) security in the random oracle model as per the key indistinguishability security game of Section 2.2 under the (f, ϕ, F)-GDDHE hardness assumption.*

Proof. Assume that there is a PPT adversary $\mathcal{A}$ that breaks the selective semantic security of our proposed KEMD scheme with a non-negligible advantage. We construct a distinguisher $\mathcal{C}$ that attempts to solve the (f, ϕ, F)-GDDHE problem using $\mathcal{A}$ as a subroutine. Both $\mathcal{A}$ and $\mathcal{C}$ are given N, the total number of users and t, the total number queries for key generation and random oracle. Let $\mathcal{C}$ be given an (f, ϕ, F)-GDDHE instance $\langle Z = (\mathbb{S}, f(x), \phi(x), h_0, h_0^{\alpha}, h_0^{\alpha^2}, \ldots, h_0^{\alpha^{t-1}}, h_0^{\alpha f(\alpha)}, h_0^{k\alpha f(\alpha)}, g_0, g_0^{\alpha}, g_0^{\alpha^2}, \ldots, g_0^{\alpha^{2N}}, g_0^{k\phi(\alpha)}), X\rangle$, where $f(x) = \prod_{i=1}^{t}(x + x_i)$, $\phi(x) = \prod_{t+1}^{t+N}(x + x_i)$ are two co-prime polynomials with pairwise distinct roots i.e., $x_i \in_R \mathbb{Z}_p, i \in [t + N]$ are all distinct, $\mathbb{S} = (p, \mathbb{G}, \mathbb{G}_1, e)$, g_0, h_0 are generators of group $\mathbb{G}$, $X = e(g_0, h_0)^{kf(\alpha)}$ or random element of $\mathbb{G}_1$. The distinguisher $\mathcal{C}$ attempts to output 0 if $X = e(g_0, h_0)^{kf(\alpha)}$ and 1 otherwise, using $\mathcal{A}$ as a subroutine. Let us denote $f_i(x) = \frac{f(x)}{x+x_i}$ for $i \in [t]$, $\phi_i(x) = \frac{\phi(x)}{x+x_i}$ for $i \in [t+1, t+N]$. Now $\mathcal{C}$ plays the role of a challenger in the security game described in Section 2.2 and interacts with $\mathcal{A}$ as follows:

Initialization: The adversary $\mathcal{A}$ selects a target recipient set G of s^* users with identity set $S = \{ID_1^*, \ldots, ID_{s^*}^*\} \subseteq \mathsf{ID} = \{ID_1, ID_2, \ldots, ID_N\} \in (\mathbb{Z}^+)^N$and declares it to $\mathcal{C}$. Here ID is the set of identities of the group of N users.

Setup: Using Z, the challenger $\mathcal{C}$ first computes $\prod_{i=t+s^*+1}^{t+N}(x + x_i) = \sum_{i=0}^{N-s^*} x^i A_i$ (say), where A_i's, are function of $x_j, j \in [t+s^*+1, t+N]$ for $i \in [0, N-s^*]$. We note down here that x_j are distinct roots of polynomial $\phi(x)$ which $\mathcal{C}$ can extract from the polynomial. Using these A_i values, $\mathcal{C}$ computes $\prod_{i=0}^{N-s^*}(g_0^{\alpha^i})^{A_i} = g_0^{\sum_{i=0}^{N-s^*}\alpha^i A_i} = g_0^{\prod_{i=t+s^*+1}^{t+N}(\alpha+x_i)}$ by extracting $g_0^{\alpha^i}$ values from Z and sets $g = g_0^{\prod_{i=t+s^*+1}^{t+N}(\alpha+x_i)}$, $g_j = g^{\alpha^j} = \prod_{i=0}^{N-s^*}(g_0^{\alpha^{i+j}})^{A_i}$.

Note that $\prod_{i=0}^{N-s^*}(g_0^{\alpha^{i+j}})^{A_i} = g_0^{\sum_{i=0}^{N-s^*}\alpha^{i+j}A_i} = g_0^{\alpha^j\left\{\prod_{i=t+s^*+1}^{t+N}(\alpha+x_i)\right\}} = g^{\alpha^j} = g_j$.

The challenger also computes

$f(x)\prod_{i=t+s^*+1}^{t+N}(x + x_i) = \prod_{i=1}^{t}(x + x_i)\prod_{i=t+s^*+1}^{t+N}(x + x_i) = \sum_{i=0}^{N-s^*+t} x^i C_i$ (say),

where C_i, are function of x_j, $j \in [t+s^*+1, t+N] \cup [1, t]$ for $i \in [0, N-s^*+t]$. Here x_j are distinct roots of $f(x)$ and $\phi(x)$, which are made available to $\mathcal{C}$ through $f(x), \phi(x)$ provided in Z. Using these C_i values, $\mathcal{C}$ computes $\prod_{i=0}^{N-s^*+t}(g_0^{\alpha^i})^{C_i} = g_0^{\sum_{i=0}^{N-s^*+t}\alpha^i C_i} = g_0^{f(\alpha)\left\{\prod_{i=t+s^*+1}^{t+N}(\alpha+x_i)\right\}}$ and $e\big(g_0^{f(\alpha)\left\{\prod_{i=t+s^*+1}^{t+N}(\alpha+x_i)\right\}}, h_0\big) = e(g_0, h_0)^{f(\alpha)\left\{\prod_{i=t+s^*+1}^{t+N}(\alpha+x_i)\right\}}$. Note that, $N - s^* + t \leq 2N$ as $t, s^* \leq N$. Therefore, all $g_0^{\alpha^i}$ values required for the above computation can be extracted by $\mathcal{C}$ from Z. The challenger $\mathcal{C}$ finally sets

$w = h_0^{\alpha f(\alpha)}, v = e(g_0, h_0)^{f(\alpha)\left\{\prod_{i=t+s^*+1}^{t+N}(\alpha+x_i)\right\}}$ and gives public parameter $\mathsf{PP} = (\mathbb{S}, g, g_1, \ldots, g_N, v, w, H, \mathsf{ID})$ to $\mathcal{A}$, where $H : \{0,1\}^* \to \mathbb{Z}_p^*$ is a cryptographic hash function selected by $\mathcal{C}$ himself.

Observe that, $w = h_0^{\alpha f(\alpha)} = h^{\alpha}$,
$v = e(g_0, h_0)^{f(\alpha)\left\{\prod_{i=t+s^*+1}^{t+N}(\alpha+x_i)\right\}} = e(g_0^{\prod_{i=t+s^*+1}^{t+N}(\alpha+x_i)}, h_0^{f(\alpha)}) = e(g, h)$,
where $h = h_0^{f(\alpha)}$ is set by $\mathcal{C}$ implicitly. This makes the distribution of PP simulated above identical as in the original scheme. As α or $h_0^{\alpha^t}$ is not known to adversary $\mathcal{A}$ or challenger $\mathcal{C}$, they can not compute h.

Hash queries: The challenger maintain hash list HL that contains at the beginning $\{*, x_i, *\}_{i=1}^{t}$, $\{ID_{i-t}^*, x_i, *\}_{i=t+1}^{t+s^*}$ ($*$ stands for empty entry) to reply at most $t - q$ hash queries, where q is number of key generation queries. If the queried identity already exists in HL, $\mathcal{C}$ responds with corresponding hash value. Else picks x_i for some $\{*, x_i, *\}$ in HL, returns $H(ID_i) = x_i$ to $\mathcal{A}$, adds $\{ID_i, x_i, *\}$ to HL.

Query Phase 1: The adversary $\mathcal{A}$ issues key generation queries on $\{ID_i\}_{i=1}^{m}$ with a restriction that $ID_i \notin S$. The challenger generates private key as:
If $\mathcal{A}$ already issued a key generation query on ID_i, $\mathcal{C}$ can find an entry (ID_i, x_i, sk_i) in HL and responds to $\mathcal{A}$ with this sk_i.
Else if $\mathcal{A}$ has already issued a hash query on ID_i, then $\mathcal{C}$ can find an entry $(ID_i, x_i, *)$ in HL, uses this x_i to compute $f_i(x) = \frac{f(x)}{x+x_i} = \sum_{i=0}^{t-1} D_i x^i$ (say), where D_i's are function of the roots x_j, $j \in [1, t]$ of $f(x)$ for $i \in [0, t-1]$, sets $sk_i = \prod_{i=0}^{t-1} h_0^{\alpha^i D_i} = h_0^{\sum_{i=0}^{t-1}(\alpha^i D_i)} = h_0^{f_i(\alpha)}$, adds (ID_i, x_i, sk_i) to HL and responds to $\mathcal{A}$ with this sk_i.
Note that $sk_i = h_0^{f_i(\alpha)} = h_0^{\frac{f(\alpha)}{\alpha+x_i}} = h^{\frac{1}{\alpha+x_i}} = h^{\frac{1}{\alpha+H(ID_i)}}$ has the same distribution as in the original scheme.
Else $\mathcal{C}$ sets $H(ID_i) = x_i$, (as in the **Hash queries** phase), computes the corresponding sk_i exactly as above, adds (ID_i, x_i, sk_i) to HL and responds to $\mathcal{A}$ with this sk_i.

Challenge: The challenger $\mathcal{C}$ first extracts $(h_0^{k\alpha f(\alpha)}, g_0^{k\phi(\alpha)})$ from the (f, ϕ, F)-GDDHE instance $\langle Z, X \rangle$ and sets the header Hdr as, $\mathsf{Hdr} = (h_0^{-k\alpha f(\alpha)}, g_0^{k\phi(\alpha)})$.
Observe that, $h_0^{-k\alpha f(\alpha)} = (h_0^{\alpha f(\alpha)})^{-k} = w^{-k}$,

$$g_0^{k\phi(\alpha)} = g_0^{k\left(\prod_{i=t+1}^{t+s^*}(\alpha+x_i)\prod_{i=t+s^*+1}^{t+N}(\alpha+x_i)\right)} = g^{k\left(\prod_{i=t+1}^{t+s^*}(\alpha+x_i)\right)} = g^{k\prod_{i=1}^{s^*}(\alpha+H(ID_i^*))}$$

are similar to our real construction from $\mathcal{A}$'s point of view.
The challenger $\mathcal{C}$ then computes the polynomial
$q(x) = \frac{1}{x}\left(\prod_{i=t+s^*+1}^{t+N}(x+x_i) - \prod_{i=t+s^*+1}^{t+N} x_i\right) = \sum_{i=0}^{N-s^*-1} x^i \bar{A}_i$ (say), where $\bar{A}_i$, are

function of x_j, $j \in [t+s^*+1, t+N]$ for $i \in [0, N-s^*-1]$. It then generates $\prod_{i=0}^{N-s^*-1} g_0^{\bar{A}_i \alpha^i} = g_0^{\sum_{i=0}^{N-s^*-1} \alpha^i \bar{A}_i} = g_0^{q(\alpha)}$ by extracting $g_0^{\alpha^i}$ from the given instance $\langle Z, X \rangle$ and sets session key K as, $K = \Big[(X)^{\prod_{i=t+s^*+1}^{t+N} x_i}\Big] e(h_0^{k\alpha f(\alpha)}, g_0^{q(\alpha)})$, where X is extracted from the (f,ϕ,F)-GDDHE instance. The challenger $\mathcal{C}$ finally chooses $b \in_R \{0,1\}$ and sets $K_b = K$, K_{1-b} as a random element of $\mathbb{G}_1$ and returns $(\mathsf{Hdr}, K_b, K_{1-b})$ to $\mathcal{A}$.

Here $X = e(g_0, h_0)^{kf(\alpha)}$ or random element of $\mathbb{G}_1$, if $X = e(g_0, h_0)^{kf(\alpha)}$ then

$$K = \Big[(X)^{\prod_{i=t+s^*+1}^{t+N} x_i}\Big] e(h_0^{k\alpha f(\alpha)}, g_0^{q(\alpha)})$$
$$= \Big[e(g_0, h_0)^{kf(\alpha)\left\{\prod_{i=t+s^*+1}^{t+N} x_i\right\}}\Big]\Big[e(g_0, h_0)^{kf(\alpha)\left(\prod_{i=t+s^*+1}^{t+N} (\alpha+x_i) - \prod_{i=t+s^*+1}^{t+N} x_i\right)}\Big]$$
$$= e(g_0, h_0)^{kf(\alpha)\left(\prod_{i=t+s^*+1}^{t+N} (\alpha+x_i)\right)} = e(g_0^{\prod_{i=t+s^*+1}^{t+N} (\alpha+x_i)}, h_0^{f(\alpha)})^k = e(g,h)^k = v^k.$$

Hence the simulated session key K has the same distribution as in original scheme.

Phase 2: This is similar to Phase 1 key generation queries. The adversary $\mathcal{A}$ sends key generation queries for $\{ID_i\}_{m+1}^q$ with a restriction that $ID_i \notin S$ and receives back secret keys $\{sk_i\}_{m+1}^q$ simulated in the same manner by $\mathcal{C}$ as in Phase 1.

Guess: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ of b to $\mathcal{C}$ and wins if $b' = b$.

We define $X = e(g_0, h_0)^{kf(\alpha)}$ as real event and X a random element of $\mathbb{G}_1$ as rand event. Therefore

$$\begin{aligned} Adv_{\mathcal{C}}^{(f,\phi,F)-\mathsf{GDDHE}} &= |Pr[b'=b|\mathsf{real}] - Pr[b'=b|\mathsf{rand}]| = |Pr[b'=b|\mathsf{real}] - \frac{1}{2}| \\ &= \Big|(\frac{1}{2}Pr[b'=1|b=1 \wedge \mathsf{real}] + \frac{1}{2}Pr[b'=0|b=0 \wedge \mathsf{real}]) - \frac{1}{2}\Big| \\ &= \Big|\frac{1}{2}Pr[b'=1|b=1 \wedge \mathsf{real}] - \frac{1}{2}Pr[b'=1|b=0 \wedge \mathsf{real}]\Big|. \\ &\quad [\text{ as } Pr[b'=0|b=0 \wedge \mathsf{real}] + Pr[b'=1|b=0 \wedge \mathsf{real}] = 1] \end{aligned}$$

In real case, the distribution of all the variables agrees with the semantic security game, thereby

$Adv_{\mathcal{A}}^{\mathsf{KEMD-INDK}}(t,N) = |Pr[b'=1|b=1 \wedge \mathsf{real}] - Pr[b'=1|b=0 \wedge \mathsf{real}]|$.

This implies $Adv_{\mathcal{C}}^{(f,\phi,F)-\mathsf{GDDHE}} = \frac{1}{2} Adv_{\mathcal{A}}^{\mathsf{KEMD-INDK}}(t,N)$. Therefore, if $\mathcal{A}$ has non-negligible advantage in correctly guessing b', then $\mathcal{C}$ solves (f,ϕ,F)-GDDHE instance given to $\mathcal{C}$ with non-negligible advantage. Hence the theorem follows.

Table 1. Comparative summaries of storage, communication bandwith and security of BED schemes.

Scheme	$\|\mathsf{PP}\|$	$\|\mathsf{PK}\|$	$\|\mathsf{SK}\|$	$\|P(G)\|$	$\|\mathsf{CT}\|$	SM	MC	SA
[11]	$(2N+4)\|\mathbb{G}\|$ $+1\|\mathbb{G}_1\|$	$N\|\mathbb{Z}_p\|$ $+N\|\mathbb{G}\|$	$(N+1)\|\mathbb{G}\|$	$5\|\mathbb{G}\|+1\|\mathbb{G}_1\|$	$2\|\mathbb{G}\|+1\|\mathbb{G}_1\|$	Semi-static	Semantic	N-DBDHE
Our KemD	$(N+2)\|\mathbb{G}\|$ $+1\|\mathbb{G}_1\|$	0	$1\|\mathbb{G}\|$	$3\|\mathbb{G}\|+1\|\mathbb{G}_1\|$	$2\|\mathbb{G}\|+1\|\mathbb{G}_1\|$	Selective	Semantic	GDDHE

$|\mathsf{PP}|$ = public parameter size, $|\mathsf{PK}|$ = public key size, $|\mathsf{SK}|$ = secret key size, $|P(G)|$ = group token size, $|\mathsf{CT}|$ = ciphertext size, N = total number of users, $|\mathbb{G}|$ = bit size of an element of $\mathbb{G}$, $|\mathbb{G}_1|$ = bit size of an element of $\mathbb{G}_1$, $|\mathbb{Z}_p|$ = bit size of an element of $\mathbb{Z}_p$, SM = security model, MC = message confidentiality, SA = security assumption, N-DBDHE = N- decisional bilinear diffie-hellman exponent, GDDHE = general decisional diffie-hellman exponent.

Table 2. Comparative summary of computation cost of parameter generation, encryption and decryption algorithm for BED schemes.

Scheme	PP		SK		$P(G)$		Verify	Enc		Dec		
	#exp	#pair	#exp	# inv	#exp	# inv	#pair	#exp	#pair	#exp	#pair	# inv
[11]	$2N+3$ in $\mathbb{G}$	1	$N+2$ in $\mathbb{G}$	1	$k'+4$ in $\mathbb{G}$, 1 in $\mathbb{G}_1$	0	2	2 in $\mathbb{G}$, 1 in $\mathbb{G}_1$	2	0	2	1 in $\mathbb{G}_1$
Our KemD	$N+1$ in $\mathbb{G}$	0	1 in $\mathbb{G}$	1	$2k'+3$ in $\mathbb{G}$, 1 in $\mathbb{G}_1$	1 in $\mathbb{G}_1$	2	2 in $\mathbb{G}$, 1 in $\mathbb{G}_1$	0	k'-1 in $\mathbb{G}$ 1 in $\mathbb{G}_1$	2	1 in $\mathbb{G}_1$

PP = public parameter, SK = secret key, $P(G)$ = group token, Enc = encryption, Dec = decryption, N = total number of users, k' = number of users selected by the dealer, #exp = number of exponentiations, #pair = number of pairings, #inv = number of inversions.

5 Efficiency

We compare our KEMD construction with the only known work of Gritti et al. [11] in Tables 1 and 2 which exhibit significant improvement in parameter sizes and computation overhead of our scheme over [11].

Our proposed scheme is essentially a key encapsulation mechanism in dealership framework whereas the construction of [11] is message encryption in dealership framework. Unlike [11], our construction does not require any public key and has constant size secret key. More interestingly, the sizes of the public parameter, secret key, group token and ciphertext are less in our KEMD design than those of [11]. Computation cost in our construction is also favourably comparable with that of [11]. The total number of exponentiation in our scheme is $3k' + N + 9$, whereas in [11] number of exponentiation is $3N + k' + 13$. Here N is the total number of users and k' is the number of subscribed users. As $k' \leq N$, our scheme requires less exponentiation. Our scheme needs 5 pairings whereas [11] needs 7 pairings. While [11] is semi-statically secure in the standard model, our KEMD is selectively secure in the random oracle model.

Remark 4. Session key K is used for message encryption. If we compare with a message encryption scheme, we can consider ciphertext CT as $\mathsf{CT} = (\mathsf{Hdr}, MK)$. In our scheme, we can consider ciphertext size as 1 more to the size of header.

6 Conclusion

We have proposed a BED scheme in key encapsulation mode, namely KEMD which significantly reduces the parameter sizes and computation cost compared to the only existing BED scheme constructed by Gritti et al. [11]. The scheme is selectively secure against CPA under reasonable assumption. We have also discussed privacy and maximum number of accountability issues. Furthermore, unlike [11] the broadcaster in our scheme does not have to wait for response from user's side to detect illegal behaviour of a dealer.

References

1. A. Barth, D. Boneh, B. Waters, Privacy in encrypted content distribution using private broadcast encryption, in: G. Di Crescenzo, A. Rubin (eds.), Financial Cryptography and Data Security, vol. 4107 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006, pp. 52–64.
2. D. Boneh, C. Gentry, B. Waters, Collusion resistant broadcast encryption with short ciphertexts and private keys, in: Proceedings of the 25th Annual International Conference on Advances in Cryptology, CRYPTO'05, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 258–275.
3. D. Boneh, B. Waters, M. Zhandry, Low overhead broadcast encryption from multilinear maps, in: J. Garay, R. Gennaro (eds.), Advances in Cryptology CRYPTO 2014, vol. 8616 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2014, pp. 206–223.
4. B. Chor, A. Fiat, M. Naor, Tracing traitors, in: Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO '94, Springer-Verlag, London, UK, 1994, pp. 257–270.
5. C. Delerablée, Identity-based broadcast encryption with constant size ciphertexts and private keys, in: Proceedings of the Advances in Crypotology 13th International Conference on Theory and Application of Cryptology and Information Security, ASIACRYPT'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 200–215.
6. C. Delerablée, P. Paillier, D. Pointcheval, Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys., in: T. Takagi, T. Okamoto, E. Okamoto, T. Okamoto (eds.), Pairing, vol. 4575 of Lecture Notes in Computer Science, Springer, 2007, pp. 39–59.
7. Y. Dodis, N. Fazio, Public key broadcast encryption for stateless receivers, in: J. Feigenbaum (ed.), Digital Rights Management, vol. 2696 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2003, pp. 61–80.
8. A. Fiat, M. Naor, Broadcast encryption, in: Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO '93, Springer-Verlag New York, USA, 1994, pp. 480–491.
9. C. Gentry, Practical identity-based encryption without random oracles, in: Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques, EUROCRYPT'06, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 445-464.
10. C. Gentry, B. Waters, Adaptive security in broadcast encryption systems (with short ciphertexts), in: A. Joux (ed.), Advances in Cryptology - EUROCRYPT 2009, vol. 5479 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 171–188.

11. C. Gritti, W. Susilo, T. Plantard, K. Liang, D. Wong, Broadcast encryption with dealership, International Journal of Information Security (2015) 1–13.
12. F. Guo, Y. Mu, W. Susilo, V. Varadharajan, Membership encryption and its applications, in: C. Boyd, L. Simpson (eds.), Information Security and Privacy, vol. 7959 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 219–234.
13. A. Lewko, A. Sahai, B. Waters, Revocation systems with very small private keys, in: IEEE Symposium on Security and Privacy (SP), 2010, pp. 273–285.
14. D. Naor, M. Naor, J. Lotspiech, Revocation and tracing schemes for stateless receivers, in: J. Kilian (ed.), Advances in Cryptology CRYPTO 2001, vol. 2139 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2001, pp. 41–62.
15. D. H. Phan, D. Pointcheval, S. Shahandashti, M. Strefler, Adaptive CCA broadcast encryption with constant-size secret keys and ciphertexts, International Journal of Information Security 12 (4) (2013) 251–265.

A The BED construction of [11]

The portions in the following scheme of [11] framed by boxes indicates those terms which were added or modified in transition from the syntax of KEMD as described in Section 2.1 to the syntax of BED of [11].

$(\mathsf{PP}, \mathsf{MK}) \leftarrow \mathsf{Setup}(N, \lambda)$: The PKGC chooses a bilinear group system $\mathbb{S} = (p, \mathbb{G}, \mathbb{G}_1, e)$, where $\mathbb{G}, \mathbb{G}_1$ are groups of prime order p and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a bilinear mapping. Let g be a generator of $\mathbb{G}$ and $h \in_R \mathbb{G}$. It selects $\alpha, \beta, \gamma \in_R \mathbb{Z}_p$, computes $u_i = h^{\gamma\alpha^i}, v_i = h^{\gamma\beta\alpha^i}$ for $i \in [0, N]$ and sets public parameter PP and master key MK as

$$\mathsf{MK} = (\alpha, \beta, \gamma), \mathsf{PP} = (\mathbb{S}, g, h, e(g^\gamma, g), \{u_i\}_{i=0}^N, \{v_i\}_{i=0}^N).$$

$(sk_i, \boxed{\mathsf{PK}_i}) \leftarrow \mathsf{KeyGen}(\mathsf{PP}, \mathsf{MK}, i)$: The PKGC takes $s_i \in_R \mathbb{Z}_p$, $f_i \in_R \mathbb{G}$ for $i \in [1, N]$ and generates a secret key for user i as $sk_i = (d_{i,0}, \ldots, d_{i,N})$, where $d_{i,0} = g^{-s_i}, d_{i,i} = g^\gamma f_i^{s_i}, d_{i,j} = f_j^{s_i}$ for $i \neq j$. The PKGC additionally generates the public key for user i as $\mathsf{PK}_i = (x_i + \alpha, f_i)$ where $x_i \in_R \mathbb{Z}_p$. It makes PK_i public and sends sk_i to user i securely through a secure communication channel.

$(P(G), k) \leftarrow \mathsf{GroupGen}(\mathsf{PP}, \boxed{\{\mathsf{PK}_i\}_{i=1}^N}, G)$: A dealer selects a group G of $k' (\leq k)$ users and generates a group token $P(G)$ as

$$\begin{aligned} P(G) &= (w_1, w_2, w_3, w_4, w_5, w_6) \\ &= (u_0^{t_1 \prod_{i \in G}(x_i+\alpha)}, v_0^{t_1 \prod_{i \in G}(x_i+\alpha)}, v_{N-k}^{t_1 \prod_{i \in G}(x_i+\alpha)}, \prod_{i \in G} f_i^{t_2}, g^{t_2}, e(g^\gamma, g)^{t_2}) \end{aligned}$$

where $t_1, t_2 \in_R \mathbb{Z}_p$, u_i, v_i are extracted from PP, $x_i + \alpha$, f_i are extracted from PK_i for $i \in [N]$. The dealer sends G to each subscribed user through a secure communication channel.

$(0 \vee 1) \leftarrow \mathsf{KEMD.Verify}(P(G), \mathsf{PP}, k)$: The broadcaster implicitly verifies that the size of G does not exceed k by checking the pairing $e(w_2, u_N) = e(w_3, u_k)$. If the verification succeeds, the broadcaster outputs 1 and proceeds; otherwise it outputs 0 and aborts.

$(\boxed{C}) \leftarrow \mathsf{Encrypt}(P(G), \mathsf{PP}, \boxed{M})$: The broadcaster verifies that $w_2 = w_1^{\beta}$ by checking $e(w_1, v_0) = e(w_2, u_0)$. If the verification succeeds the broadcaster generates a ciphertext C using $P(G) = (w_1, w_2, w_3, w_4, w_5, w_6)$, PP and a message $M \in \mathbb{G}_1$ as
$C = (C_1, C_2, C_3) = (w_5^r, w_4^r, Mw_6^r) = (g^{rt_2}, \prod_{i \in G} f_i^{rt_2}, M.e(g^{\gamma}, g)^{rt_2})$
where $r \in_R \mathbb{Z}_p$.

$(\boxed{M}) \leftarrow \mathsf{Decrypt}(\mathsf{PP}, sk_i, \boxed{C}, G)$: User i checks the cardinality of G which he receives from the dealer. If it is greater than k, then user i informs this to the broadcaster. User i retrieves M by coupling $C = (C_1, C_2, C_3)$ with $d_{i,j}$'s extracted from sk_i as follows:

$$\begin{aligned} X &= e(d_{i,i} \prod_{j \in G, j \neq i} d_{i,j}, C_1) e(d_{i,0}, C_2) \\ &= e(g^{\gamma} \prod_{j \in G} f_j^{s_i}, g^{rt_2}) e(g^{-s_i}, \prod_{j \in G} f_j^{rt_2}) = e(g^{\gamma}, g^{rt_2}) \\ X^{-1} C_3 &= e(g^{\gamma}, g^{rt_2})^{-1} M e(g^{\gamma}, g^{rt_2}) = M. \end{aligned}$$