Multi-Channel Broadcast Encryption

Duong Hieu Phan[1,2], David Pointcheval[2], and Viet Cuong Trinh[1]

[1]LAGA, University of Paris 8
[2]ENS / CNRS / INRIA

Abstract. Broadcast encryption aims at sending a content to a large arbitrary group of users at once. Currently, the most efficient schemes provide constant-size headers, that encapsulate ephemeral session keys under which the payload is encrypted. However, in practice, and namely for pay-TV, providers have to send various contents to different groups of users. Headers are thus specific to each group, one for each channel: as a consequence, the global overhead is linear in the number of channels. Furthermore, when one wants to zap to and watch another channel, one has to get the new header and decrypt it to learn the new session key: either the headers are sent quite frequently or one has to store all the headers, even if one watches one channel only. Otherwise, the zapping time becomes unacceptably long.
In this paper, we consider encapsulation of several ephemeral keys, for various groups and thus various channels, in one header only, and we call this new primitive *Multi-Channel Broadcast Encryption* – MIBE: one can hope for a much shorter global overhead and a short zapping time since the decoder already has the information to decrypt any available channel at once. Our candidates are private variants of the Boneh-Gentry-Waters scheme, with a constant-size global header, independently of the number of channels. In order to prove the CCA security of the scheme, we introduce a new *dummy-helper technique* and implement it in the random oracle model.

1 Introduction

Broadcast encryption has been widely and deeply studied as it is a core primitive for many concrete applications. In the following, we focus on the pay-TV scenario, in which users own decoders to decode only the channels they subscribed to. In this context, the broadcaster sends several channels at the same time, to different groups of users or target sets.

Unfortunately, previous broadcast encryption models only dealt with one single content and one single target set at a time. This was a first reasonable goal to get such an efficient broadcast encryption scheme, but not quite relevant to practice. In fact, TV systems contain many channels, with different sets of privileged users. One could argue that this scenario is covered by the usual systems, applying independent broadcast encryption schemes for each channel. However, this results in a very inefficient scheme: the bandwidth or header size grows linearly in the number of channels, which could be very large; in case of zapping to another channel, one has to start from scratch, and namely to wait for the reception of the new appropriate header, which can take some time, unless the decoder stores all the headers all the time.

These two problems, of the bandwidth that should not be too large and zapping time that should not be too long, lead to new efficiency criteria, with a common solution: a broadcast encryption with a short *global header*. Our new primitive MIBE, for MultI-channel Broadcast Encryption, addresses them. In the following, we show that it is possible to achieve this goal in an optimal way: a constant-size global header, independently of number of channels.

Broadcast encryption schemes. Broadcast encryption was first described by Fiat and Naor in [5] but receives much attention since the work of Naor, Naor, and Lotspiech [8] in which they presented a symmetric-key subset-cover framework along with a security model and a security analysis. Dodis and Fazio [4] presented the first public-key CCA-secure scheme. Boneh, Gentry, and Waters [2] designed a fully collusion-resistant scheme and proposed a security model where the adversary can corrupt any user, except the users in the challenge target set. With their scheme, the adversary had to precise this challenge target set before knowing the parameters of the system, hence the so-called *selective model*. Delerablée constructed a selectively secure ID-based BE [3] in the random oracle model. Thereafter, Gentry and Waters [6] defined the *adaptive model*, where the adversary can corrupt users and then adaptively choose the challenge target set, and provided adaptively secure schemes in the standard and the random oracle models. Waters [11] and Lewko *et. al.* [7] used dual system encryption to achieve adaptive security. Recently, a scheme that achieves all desired properties (constant-size ciphertexts, adaptive and CCA security) has been presented in [9] but it relies on rather non-standard assumptions.

Phan, Pointcheval and Strefler [10] recently gave a global picture of the relations between the security notions for broadcast encryption. However, our setting of multi-channel broadcast encryption go beyond their consideration, because the adversary could corrupt some user of one channel to break the security of the other channels. The sessions keys of all channels should indeed be compacted into one ciphertext only, there are thus some relations between these keys inside one session and the security model has to take these relation into account.

Contributions. We first propose a formalization of the problem, with the so-called *Multi-Channel Broadcast Encryption* – MIBE. Because of some constraints between the various target sets, we introduce the *dummy-helper technique* that helps to prove the security. We eventually propose two constructions, derived from the Boneh-Gentry-Waters (BGW) [2] scheme. They are private broadcast encryption schemes, with the following properties:

– The first construction is, asymptotically, very competitive with the BGW scheme. In fact, it achieves the constant-size header, while the private decryption key size remains linear in the number of the channels that a user has subscribed to. In addition, it is fully collusion resistant against basic selective adversaries, *i.e.* the adversaries who can only ask corruption queries to get the decryption keys of users in the selective security model (the challenge target set is announced before having seen the global parameters). This is also the security level that the original BGW scheme achieves and our security proof holds under the standard assumption $\mathsf{n}-\mathsf{BDHE}$, as in the original BGW scheme [2].
– The second construction improves on the previous one, to resist to strong selective adversaries who have the power of basic selective adversaries plus unlimited access to encryption and decryption queries, while keeping the parameter sizes and computational assumptions unchanged. To this aim, we introduce the *dummy-helper technique* and make use of a *random oracle* [1]. Our scheme is more efficient than the CCA version of the BGW scheme [2] but our dummy-helper technique actually works in the random oracle model.

Dummy-helper technique. In the multi-channel setting, because the session keys of all channels are compacted in only one ciphertext, there exists an implicit relation between the session keys of the channels which could be known by the simulator without the whole knowledge of the master key. By introducing the *dummy-helper technique*, which consists in adding a new channel for one additional dummy user, we get the following interesting properties:

1. it gives our simulator the possibility to decrypt this channel and get the corresponding session key. This is then sufficient for the simulator to derive the other session keys and successfully answer any decryption query.
2. by eventually publishing the decryption key of the dummy user, it introduces a channel that can be decoded by all the users registered in the system: to send the program or ads.

We implement this dummy-helper technique in the random oracle model. It is worth noting that, though working in a more complex setting of multi-channel broadcast encryption, the security is achieved under the standard assumption $\mathsf{n}-\mathsf{BDHE}$ as in the BGW scheme.

2 Multi-channel Broadcast Encryption

2.1 Syntax

In this section we describe the model for a multi-channel broadcast encryption system. Formally, such a system consists of four probabilistic algorithms:

Setup(λ)**:** Takes as input the parameter security λ, it generates the global parameters param of the system, and returns a master key MSK and an encryption key EK. If the scheme allows encryption, EK is public, otherwise EK is kept private, and can be seen as a part of MSK.

Extract(i, MSK)**:** Takes as input the user's index i, together with the master key, and outputs the user's private key d_i.

Encrypt$(S_1, S_2, \ldots, S_m, \mathsf{EK})$**:** Takes as input m subsets (or target sets) $S_1, S_2, \ldots, S_m$ where, for $i = 1, \ldots, m$, $S_i \subseteq \{1, \ldots, n\}$, and the encryption key EK. It outputs $(\mathsf{Hdr}, K_1, K_2, \ldots, K_m)$ where Hdr encapsulates the ephemeral keys $(K_i)_i \in \mathcal{K}$. The key K_i will be associated to the subset S_i. We will refer to Hdr as the broadcast ciphertext, or *header*, whereas this header together with the description of all the target sets is called the *full header*.

Decrypt$(S_1, S_2, \ldots, S_m, \mathsf{Hdr}, j, d_j, i)$: Takes as input a full header $(S_1, S_2, \ldots, S_m, \mathsf{Hdr})$, a user $j \in \{1, \ldots, n\}$ and its private key d_j, together with a subgroup index $i \in \{1, \ldots, m\}$. If $j \in S_i$, then the algorithm outputs the ephemeral key $K_i \in \mathcal{K}$.

For correctness, we require that for all subsets $S_i \subseteq \{1, \ldots, n\}$ and all $j \in S_i$, if $(\mathsf{EK}, \mathsf{MSK}) \leftarrow$ **Setup**(λ), $d_j \leftarrow$ **Extract**(j, MSK) and $(\mathsf{Hdr}, K_1, \ldots, K_m) \leftarrow$ **Encrypt**$(S_1, S_2, \ldots, S_m, \mathsf{EK})$ then $K_i =$ **Decrypt**$(S_1, S_2, \ldots, S_m, \mathsf{Hdr}, j, d_j, i)$.

In practice, the goal of such ephemeral keys is to encrypt the payload, which consists of m messages $M_1, \ldots, M_m$ to be broadcast to the sets $S_1, \ldots, S_m$ respectively. They will thus be encrypted under the symmetric keys $K_1, \ldots, K_m$ into the ciphertexts $\mathsf{CM}_1, \ldots, \mathsf{CM}_m$ respectively. The broadcast to all users in $S_1, S_2, \ldots, S_m$ consists of $(S_1, S_2, \ldots, S_m, \mathsf{Hdr}, \mathsf{CM}_1, \mathsf{CM}_2, \ldots, \mathsf{CM}_m)$ where $(S_1, S_2, \ldots, S_m, \mathsf{Hdr})$ is the *full header* and $(\mathsf{CM}_1, \mathsf{CM}_2, \ldots, \mathsf{CM}_m)$ is often called the *encrypted payload.*

2.2 Security Model

We define the security of a multi-channel broadcast encryption system by the following game between an attacker $\mathcal{A}$ and a challenger:

Setup. The challenger runs the **Setup** algorithm to generate the global parameters param of the system, and returns a master key MSK and an encryption key EK. If the scheme is asymmetric, EK is given to $\mathcal{A}$, otherwise it can be seen as a part of the MSK, and thus kept secret. Corruption and decryption lists Λ_C, Λ_D are set to empty lists.

Query phase 1. The adversary $\mathcal{A}$ adaptively asks queries:

1. Corruption query for the i-th user: the challenger runs **Extract**(i, MSK) and forwards the resulting private key to the adversary. The user i is appended to the corruption list Λ_C;
2. Decryption query on the full header $(S_1, S_2, \ldots, S_m, \mathsf{Hdr})$ together with $u \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. The challenger answers with **Decrypt**$(S_1, S_2, \ldots, S_m, \mathsf{Hdr}, u, d_u, j)$. The pair (Hdr, S_j) is appended to the decryption list Λ_D

3. Encryption query (if EK is private) for the target sets $(S_1, S_2, \ldots, S_m)$. The challenger answers with $\mathbf{Encrypt}(S_1, S_2, \ldots, S_m, \mathsf{EK})$.

Challenge. The adversary $\mathcal{A}$ outputs t target sets $S_1^*, S_2^*, \ldots, S_t^* \subseteq \{1, \ldots, n\}$ and an index j, which specifies the attacked target set S_j^*.
The challenger runs $\mathbf{Encrypt}(S_1^*, S_2^*, \ldots, S_t^*, \mathsf{EK})$ and gets $(\mathsf{Hdr}^*, K_1^*, K_2^*, \ldots, K_t^*)$. Next, the challenger picks a random $b \overset{\$}{\leftarrow} \{0,1\}$, sets $K_{j,b}^* = K_j^*$ and picks random $K_{j,1-b}^* \overset{\$}{\leftarrow} \mathcal{K}$. It then outputs $(\mathsf{Hdr}^*, K_1^*, \ldots, K_{j-1}^*, K_{j+1}^*, \ldots, K_t^*, K_{j,0}^*, K_{j,1}^*)$ to $\mathcal{A}$.

Query phase 2. The adversary $\mathcal{A}$ continues to adaptively asks queries as in the first phase.

Guess. The adversary $\mathcal{A}$ eventually outputs its guess $b' \in \{0,1\}$ for b.

We say the adversary wins the game if $b' = b$, but only if $S_j^* \cap \Lambda_C = \emptyset$ and $(\mathsf{Hdr}^*, S_j^*) \notin \Lambda_D$. We then denote by $\mathbf{Succ}^{\mathsf{ind}}(\mathcal{A}) = \Pr[b' = b]$ the probability that $\mathcal{A}$ wins the game, and its advantage is

$$\mathbf{Adv}^{\mathsf{ind}}(\mathcal{A}) = 2 \times \mathbf{Succ}^{\mathsf{ind}}(\mathcal{A}) - 1 = \Pr[1 \leftarrow \mathcal{A}|b = 1] - \Pr[1 \leftarrow \mathcal{A}|b = 0].$$

Definition 1 (Full Security). A multi-channel broadcast encryption scheme is said $(t, \varepsilon, q_C, q_D, q_E)$-secure if for any t-time algorithm $\mathcal{A}$ that makes at most q_C corruption queries, q_D decryption queries, and q_E encryption queries, $\mathbf{Adv}^{\mathsf{ind}}(\mathcal{A}) \le \varepsilon$. We denote by $\mathbf{Adv}^{\mathsf{ind}}(t, q_C, q_D, q_E)$ the advantage of the best t-time adversary.

There are two classical restricted scenarios: a *selective* attacker provides the target sets $S_1^*, S_2^*, \ldots, S_t^* \subseteq \{1, \ldots, n\}$ at the beginning of the security game, and one can also restrict the adversary not to ask some queries.

Definition 2 (Basic Selective Security). A multi-channel broadcast encryption scheme is said (t, ε, q_C)-selectively secure if it is $(t, \varepsilon, q_C, 0, 0)$-secure against a selective adversary. We denote by $\mathbf{Adv}^{\mathsf{b-ind}}(t, q_C)$ the advantage of the best t-time basic selective adversary.

Definition 3 (Strong Selective Security). A multi-channel broadcast encryption scheme is said $(t, \varepsilon, q_C, q_D, q_E)$-selectively secure if it is $(t, \varepsilon, q_C, q_D, q_E)$-secure against a selective adversary. We denote by $\mathbf{Adv}^{\mathsf{s-ind}}(t, q_C, q_D, q_E)$ the advantages of the best t-time strong selective adversaries.

2.3 Disjoint Target Sets

As discussed in the introduction, our main motivation is pay-TV. For such systems, there are several channels, which are encrypted to sets of users. The users thus own decryption keys:

– When a user u registers to the system, he receives a smart card with decryption keys (d_u^i) for every channel i. But at the broadcast time, channel i is encrypted for the target set with the subscribers to this channel only (a subset of the decryption keys);
– Another possibility is to first define U_i the set of all the possible decryption keys for the channel i. When a user u subscribes to a channel i, he receives a key $d_u^i \in U_i$.

In both the above case, the target sets are subsets of predetermined and disjoint sets of keys. As a consequence, the target sets S_i are disjoint too. However, we have to define many keys in the system. In order to limit this number of keys, one could think about sharing keys for several channels. This would allow profiling on users, that can be an undesirable feature. But we can still limit a little bit the number of keys by reassigning keys when a user unsubscribes from a channel to another channel.

Anyway, in the following, at a time t, when the broadcaster encapsulates keys for several target sets S_i, we assume them to be disjoint.

3 Preliminaries

3.1 Computational Assumptions

We first recall the definition of the classical Computational Diffie-Hellman (CDH) assumption:

Definition 4 (CDH Assumption). The $(t, \varepsilon) - \mathsf{CDH}$ assumption says that for any t-time adversary $\mathcal{A}$ that is given $(g, g^r, h) \in \mathbb{G}$, its probability to output h^r is bounded by ε:

$$\mathbf{Succ}^{\mathsf{cdh}}(\mathcal{A}) = \Pr[\mathcal{A}(g, g^r, h) = h^r] \leq \varepsilon.$$

Stronger assumptions have been introduced by Boneh-Gentry-Waters [2]. They both imply the above CDH assumption.

Definition 5 (BDHE Assumption). The $(t, n, \varepsilon) - \mathsf{BDHE}$ assumption says that for any t-time adversary $\mathcal{A}$ that is given $(g, h, g^{\alpha^1}, \ldots, g^{\alpha^n}, g^{\alpha^{n+2}}, \ldots, g^{\alpha^{2n}}) \in \mathbb{G}^{2n+1}$, its probability to output $e(g, h)^{\alpha^{n+1}} \in \mathbb{G}$ is bounded by ε:

$$\mathbf{Succ}^{\mathsf{bdhe}}(\mathcal{A}) = \Pr[\mathcal{A}(g, h, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}) = e(g_{n+1}, h)] \leq \epsilon.$$

Definition 6 (DBDHE Assumption). The $(t, n, \varepsilon) - \mathsf{DBDHE}$ assumption says that for any t-time adversary $\mathcal{A}$ that is given $(g, h, g^{\alpha^1}, \ldots, g^{\alpha^n}, g^{\alpha^{n+2}}, \ldots, g^{\alpha^{2n}}) \in \mathbb{G}^{2n+1}$, and a candidate to the BDHE problem, that is either $e(g, h)^{\alpha^{n+1}} \in \mathbb{G}$ or a random value T, cannot distinguish the two cases with advantage greater than ε:

$$\mathbf{Adv}^{\mathsf{dbdhe}}(\mathcal{A}) = \left| \begin{array}{l} \Pr[\mathcal{A}(g, h, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, e(g_{n+1}, h)) = 1] \\ - \Pr[\mathcal{A}(g, h, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, T)) = 1] \end{array} \right| \leq \epsilon.$$

3.2 BGW Overview

To warm up, we first recall the BGW scheme [2], on which our constructions will rely.

Setup(λ)**:** Let $\mathbb{G}$ be a bilinear group of prime order p. The algorithm first picks a random generator $g \in \mathbb{G}$ and a random scalar $\alpha \in \mathbb{Z}_p$. It computes $g_i = g^{\alpha^i} \in \mathbb{G}$ for $i = 1, 2, \ldots, n, n+2, \ldots, 2n$. Next, it picks a random scalar $\gamma \in \mathbb{Z}_p$ and sets $v = g^\gamma \in \mathbb{G}$.
The public key is $\mathsf{EK} = (g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, v)$, whereas the private decryption key of user $i \in \{1, \ldots, n\}$ is $d_i = v^{\alpha^i}$. These decryption keys are sent by the **Extract** algorithm.

Encrypt(S, EK)**:** Pick a random scalar $r \in \mathbb{Z}_p$, and set $K = e(g_{n+1}, g)^r$, where $e(g_{n+1}, g)$ can be computed as $e(g_n, g_1)$ from EK. Next, set: $\mathsf{Hdr} = (g^r, (v \cdot \prod_{j \in S} g_{n+1-j})^r)$, and output (Hdr, K).

Decrypt$(S, \mathsf{Hdr}, i, d_i, \mathsf{EK})$**:** Parse $\mathsf{Hdr} = (C_1, C_2)$, output $K = e(g_i, C_2)/e(d_i \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, C_1)$.

Trivially, when one wants to broadcast m different messages to m different sets $S_1, S_2, \ldots, S_m$, one can combine m independent BGW schemes:

Setup(λ)**:** As in the BGW scheme.

Encrypt$(S_1, S_2, \ldots, S_m, \mathsf{EK})$**:** Pick random scalars $r_1, \ldots, r_m \in \mathbb{Z}_p$, and set

$$K_1 = e(g_{n+1}, g)^{r_1}, \ldots, K_m = e(g_{n+1}, g)^{r_m}$$
$$\mathsf{Hdr} = \Big((g^{r_1}, (v \cdot \textstyle\prod_{j \in S_1} g_{n+1-j})^{r_1}), \ldots, (g^{r_m}, (v \cdot \prod_{j \in S_m} g_{n+1-j})^{r_m}) \Big).$$

Decrypt$(S_1, \ldots, S_m, \mathsf{Hdr}, i, (\mathsf{EK}, d_i), j)$**:** Extract $C_1 = g^{r_j}, C_2 = (v \cdot \prod_{j \in S_j} g_{n+1-j})^{r_j}$ from Hdr and decrypt as in BGW.

3.3 Intuition

One can note that, in the above "trivial" construction, the number of elements in the header is $2m$, and we want to reduce it. A first attempt is by reusing the same random scalar in all the ciphertexts, which leads to a header of size $m+1$:

$$\mathsf{Hdr} = \left(g^r, (v \cdot \prod_{j \in S_1} g_{n+1-j})^r, \ldots, (v \cdot \prod_{j \in S_m} g_{n+1-j})^r \right).$$

However, this reuse of random coins suffers from a simple attack: the same random coins result in the same session keys for all channels and a subscriber of a channel can decrypt all channels, since the session key is $e(g_{n+1}, g)^r$. Different r's are thus required in each session keys, but not necessarily totally independent. Our idea is to add an element $X_i \in \mathbb{G}$ corresponding to users $i = 1, \ldots, n$, and to adapt the session key and Hdr using scalars x_i, where $X_i = g^{x_i}$, for $i = 1, \ldots, n$,

$$\begin{aligned} K_1 = e(g_{n+1}, g)^{r + \sum_{j \in S_1} x_j}, \ldots, K_m = e(g_{n+1}, g)^{r + \sum_{j \in S_m} x_j}, \\ \mathsf{Hdr} = \left(g^r, (v \cdot \textstyle\prod_{j \in S_1} g_{n+1-j})^{r + \sum_{j \in S_1} x_j}, \ldots, (v \cdot \prod_{j \in S_m} g_{n+1-j})^{r + \sum_{j \in S_m} x_j} \right) \end{aligned}$$

The above step shorten the header to $m+1$ elements, with no more easy attack. But our goal is to have a constant number of elements:

$$\mathsf{Hdr} = \left(g^r, (v \cdot \prod_{j \in S_1} g_{n+1-j})^{r + \sum_{j \in S_1} x_j} \times \cdots \times (v \cdot \prod_{j \in S_m} g_{n+1-j})^{r + \sum_{j \in S_m} x_j} \right)$$

where we essentially multiply all the ciphertexts together. And, magically, it works because a user in a set S_i can cancel out all the terms $(v \cdot \prod_{j \in S_k} g_{n+1-j})^{r + \sum_{j \in S_k} x_j}$ for $k \neq i$ in this product and transform it into his corresponding ciphertext in S_i.

Of course, security has to be proven, this is the goal of the next section to prove the basic selective security. Limitation not to ask decryption nor encryption queries is quite strong, and is the main drawback of the first scheme MIBE_1. And thus, we provide a second construction MIBE_2 that covers strong selective adversaries. For that, we replace $\prod_{j \in S_k} X_j$ by a value outputted by a random oracle on the set S_k and the value g^r at the time of encryption. It will prevent malleability. The *dummy-helper technique* will make the rest.

4 Multi-Channel Broadcast Encryption I – MIBE_1

4.1 Description

Let us now describe formally our first construction MIBE_1. We will then prove its basic selective security.

Setup(λ)**:** The algorithm takes as input the parameter security λ, it generates the global parameters param of the system as follows: Let $\mathbb{G}$ be a bilinear group of prime order p. The algorithm first picks a random generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$. It computes $g_i = g^{\alpha^i} \in \mathbb{G}$ for $i = 1, 2, \ldots, n, n+2, \ldots, 2n$. Next, it picks a random $\gamma \in \mathbb{Z}_p$ and sets $v = g^\gamma \in \mathbb{G}$. It also picks additional random scalars $x_1, x_2, \ldots, x_n \in \mathbb{Z}_p$ and sets $X_1 = g^{x_1}, X_2 = g^{x_2}, \ldots, X_n = g^{x_n}$. The master secret key is $\mathsf{MSK} = (g, v, \alpha, \gamma, x_1, x_2, \ldots, x_n)$, while the encryption key (that is private to the broadcaster) is $\mathsf{EK} = (g, v, g_{n+1}, x_1, x_2, \ldots, x_n)$. The public global parameters are $(g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, X_1, X_2, \ldots, X_n)$, whereas the private decryption key of user $i \in \{1, \ldots, n\}$ is $d_i = v^{\alpha^i}$. These decryption keys are sent by the **Extract** algorithm.

Encrypt$(S_1, S_2, \ldots, S_m, \mathsf{EK})$**:** Pick a random scalar $r \xleftarrow{\$} \mathbb{Z}_p$, then set $K_k = e(g_{n+1}, g)^{r+\sum_{j\in S_k} x_j}$ for $k = 1, \ldots, m$. Next, set

$$\mathsf{Hdr} = \left(g^r, \prod_{k=1}^{k=m} \left(v \cdot \prod_{j\in S_k} g_{n+1-j}\right)^{r+\sum_{j\in S_k} x_j} \right).$$

The broadcaster knows $g_{n+1}, x_1, \ldots, x_n$ from EK. It eventually outputs $(\mathsf{Hdr}, K_1, K_2, \ldots, K_m)$.

Decrypt$(S_1, \ldots, S_m, \mathsf{Hdr}, i, d_i, k)$**:** Parse $\mathsf{Hdr} = (C_1, C_2)$. If $i \in S_k$ then one computes

$$\begin{aligned}
K_k &= \frac{e(g_i, C_2)}{e(d_i \cdot \prod_{\substack{j\in S_k\\ j\neq i}} g_{n+1-j+i}, g^r \cdot \prod_{j\in S_k} X_i) \cdot \prod_{\substack{\ell=1\\ \ell\neq k}}^{\ell=m} e(d_i \cdot \prod_{j\in S_\ell} g_{n+1-j+i}, g^r \cdot \prod_{j\in S_\ell} X_i)} \\
&= \frac{e(g_i, C_2)}{e(d_i \cdot \prod_{\substack{j\in S_k\\ j\neq i}} g_{n+1-j+i}, g^{r+\sum_{j\in S_k} x_j}) \cdot \prod_{\substack{\ell=1\\ \ell\neq k}}^{\ell=m} e(d_i \cdot \prod_{j\in S_\ell} g_{n+1-j+i}, g^{r+\sum_{j\in S_\ell} x_j})} \\
&= \frac{e(g^{\alpha^i}, \prod_{\ell=1}^{\ell=m} (v \cdot \prod_{j\in S_\ell} g_{n+1-j})^{r+\sum_{j\in S_\ell} x_j})}{e(v^{\alpha^i} \cdot (\prod_{\substack{j\in S_k\\ j\neq i}} g_{n+1-j})^{\alpha^i}, g^{r+\sum_{j\in S_k} x_j}) \cdot \prod_{\substack{\ell=1\\ \ell\neq k}}^{\ell=m} e(v^{\alpha^i} \cdot (\prod_{j\in S_\ell} g_{n+1-j})^{\alpha^i}, g^{r+\sum_{j\in S_\ell} x_j})} \\
&= \frac{e(g^{\alpha^i}, (v \cdot \prod_{j\in S_k} g_{n+1-j})^{r+\sum_{j\in S_k} x_j})}{e(v^{\alpha^i} \cdot (\prod_{\substack{j\in S_k\\ j\neq i}} g_{n+1-j})^{\alpha^i}, g^{r+\sum_{j\in S_k} x_j})} \cdot \prod_{\substack{\ell=1\\ \ell\neq k}}^{\ell=m} \frac{e(g^{\alpha^i}, (v \cdot \prod_{j\in S_\ell} g_{n+1-j})^{r+\sum_{j\in S_\ell} x_j})}{e(v^{\alpha^i} \cdot (\prod_{j\in S_\ell} g_{n+1-j})^{\alpha^i}, g^{r+\sum_{j\in S_\ell} x_j})} \\
&= \frac{e((v \cdot \prod_{j\in S_k} g_{n+1-j})^{\alpha^i}, g^{r+\sum_{j\in S_k} x_j})}{e((v \cdot \prod_{\substack{j\in S_k\\ j\neq i}} g_{n+1-j})^{\alpha^i}, g^{r+\sum_{j\in S_k} x_j})} \cdot \prod_{\substack{\ell=1\\ \ell\neq k}}^{\ell=m} \frac{e((v \cdot \prod_{j\in S_\ell} g_{n+1-j})^{\alpha^i}, g^{r+\sum_{j\in S_\ell} x_j})}{e((v \cdot \prod_{j\in S_\ell} g_{n+1-j})^{\alpha^i}, g^{r+\sum_{j\in S_\ell} x_j})} \\
&= e(g_{n+1-i}^{\alpha^i}, g^{r+\sum_{j\in S_k} x_j}) = e(g_{n+1}, g^{r+\sum_{j\in S_k} x_j}) = e(g_{n+1}, g)^{r+\sum_{j\in S_k} x_j}
\end{aligned}$$

We used the relations $d_i = v^{\alpha^i}, g_{n+1-j+i} = g_{n+1-j}^{\alpha^i}$, and $g_{n+1-i}^{\alpha^i} = g_{n+1}$.

Remark 7. In MIBE_1, the encryption key EK contains g_{n+1} and thus cannot be public: this is a private variant of BGW scheme. However, the broadcaster does not need to know α, γ to encrypt, and without them it cannot generate decryption keys for users. We can separate the role of group manager (who generates the decryption keys) and broadcaster (who encrypts and broadcasts the content).

4.2 Security Result

We now prove the semantic security of the first scheme.

Theorem 8. *The MIBE_1 system achieves the basic selective security under the DBDHE assumption in $\mathbb{G}$. More precisely, if there are n users,*

$$\mathbf{Adv}^{\mathsf{b-ind}}(t, q_C) \leq 2 \times \mathbf{Adv}^{\mathsf{dbdhe}}(t', n) + O(1),$$

for $t' \leq t + (mn + nq_C)T_e + O(1)$ where T_e is the time complexity for computing an exponentiation and m is the maximum number of channels in the system.

Proof. Let us assume there exists an adversary $\mathcal{A}$ which breaks the semantic security of our first scheme, we build an algorithm $\mathcal{B}$ that has the same advantage in deciding the DBDHE problem in $\mathbb{G}$. This algorithm $\mathcal{B}$ proceeds as follows:

Init. Algorithm $\mathcal{B}$ first takes as input a DBDHE instance $(g, G, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, T)$, where T is either $e(g_{n+1}, G)$ or a random element of $\mathbb{G}$. It implicitly defines α: $g_i = g^{\alpha^i}$. $\mathcal{B}$ then runs $\mathcal{A}$, and since we are in the selective model, it receives m sets $S_1, \ldots, S_m$ and an index k that $\mathcal{A}$ wishes to be challenged on.

Setup. $\mathcal{B}$ now generates the public global parameters and private keys d_i, for $i \notin S_k$: it first chooses a random scalar $r \in \mathbb{Z}_p$ and sets $h = g^r$, and $h_i = g_i^r$, for $i = 1, \ldots, n$. One chooses a random index η in S_k, and for $i \in \{1, \ldots, n\} \backslash \{\eta\}$, one chooses a random scalar $x_i \in \mathbb{Z}_p$, and computes $X_i = g^{x_i}$. One eventually sets $X_\eta \overset{\mathsf{def}}{=} G / \prod_{i \in S_k \backslash \{\eta\}} X_i = g^{x_\eta}$: All the scalars x_i are known, excepted x_η. $\mathcal{B}$ gives $\mathcal{A}$ the public global parameters:

$$(g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, X_1, X_2, \ldots, X_n)$$

$\mathcal{B}$ has to compute all the private decryption keys d_i except for $i \in S_k$: It chooses a random $u \in \mathbb{Z}_p$ and sets

$$v \overset{\mathsf{def}}{=} g^u \cdot (\prod_{j \in S_k} g_{n+1-j})^{-1} \qquad d_i \overset{\mathsf{def}}{=} g_i^u / (\prod_{j \in S_k} g_{n+1-j+i}) = g^{u \cdot \alpha^i} \cdot (\prod_{j \in S_k} g_{n+1-j})^{-\alpha^i} = v^{\alpha^i}$$

On can remark that $\mathcal{B}$ can compute, without explicitly knowing α, $\prod_{j \in S_k} g_{n+1-j+i}$ for any $i \notin S_k$, and cannot when $i \in S_k$. Moreover, since $d_i = v^{\alpha^i}$, it satisfies the specifications of the schemes.

Challenge. To generate the challenge for $\mathcal{A}$, $\mathcal{B}$ first computes $\mathsf{Hdr} = (C_1, C_2)$ by setting $C_1 = h$, and

$$\begin{aligned}
C_2 &= (h^u \cdot G^u) \cdot \prod_{\substack{\ell=1 \\ \ell \neq k}}^{\ell=m} \left(h^u \cdot \left(\frac{\prod_{j \in S_\ell} h_{n+1-j}}{\prod_{j \in S_k} h_{n+1-j}} \right) \cdot (v \cdot \prod_{j \in S_\ell} g_{n+1-j})^{\sum_{j \in S_\ell} x_j} \right) \\
&= (g^u)^{r + \sum_{i \in S_k} x_i} \cdot \prod_{\substack{\ell=1 \\ \ell \neq k}}^{\ell=m} \left(g^{ur} \cdot \left(\frac{\prod_{j \in S_\ell} g_{n+1-j}}{\prod_{j \in S_k} g_{n+1-j}} \right)^r \cdot \left(v \prod_{j \in S_\ell} g_{n+1-j} \right)^{\sum_{j \in S_\ell} x_j} \right) \\
&= \left(v \prod_{j \in S_k} g_{n+1-j} \right)^{r + \sum_{i \in S_k} x_i} \prod_{\substack{\ell=1 \\ \ell \neq k}}^{\ell=m} \left(\frac{g^u}{\prod_{j \in S_k} g_{n+1-j}} \right)^r \left(\prod_{j \in S_\ell} g_{n+1-j} \right)^r \left(v \prod_{j \in S_\ell} g_{n+1-j} \right)^{\sum_{j \in S_\ell} x_j} \\
&= \left(v \prod_{j \in S_k} g_{n+1-j} \right)^{r + \sum_{i \in S_k} x_i} \prod_{\substack{\ell=1 \\ \ell \neq k}}^{\ell=m} \left(v \prod_{j \in S_\ell} g_{n+1-j} \right)^r \left(v \prod_{j \in S_\ell} g_{n+1-j} \right)^{\sum_{j \in S_\ell} x_j} \\
&= \left(v \prod_{j \in S_k} g_{n+1-j} \right)^{r + \sum_{i \in S_k} x_i} \prod_{\substack{\ell=1 \\ \ell \neq k}}^{\ell=m} \left(v \prod_{j \in S_\ell} g_{n+1-j} \right)^{r + \sum_{j \in S_\ell} x_j} = \prod_{\ell=1}^{\ell=m} \left(v \prod_{j \in S_\ell} g_{n+1-j} \right)^{r + \sum_{j \in S_\ell} x_j}
\end{aligned}$$

We used the following notations and relations $h = g^r$ and $g_{n+1-j}^r = h_{n+1-j}$. Note that $\mathcal{B}$ knows all the values x_i, excepted $x_{i_{k,t}}$, that appears in $h^u \cdot G^u = (v \cdot \prod_{j \in S_k} g_{n+1-j})^{r + \sum_{j \in S_k} x_j}$. To generate session keys, $\mathcal{B}$ first computes, for all $i \neq k$, $K_i = e(g_n, g_1)^{\sum_{j \in S_i} x_j} \cdot e(g_n, h_1)$. It then randomly chooses a bit $b \overset{\$}{\leftarrow} \{0, 1\}$ and sets $K_{k,b} = T \cdot e(g_n, h_1)$ and picks a random $K_{k,1-b}$ in $\mathbb{G}$. It outputs $(\mathsf{Hdr}, K_1, \ldots, K_{k-1}, K_{k+1}, \ldots, K_m, K_{k,0}, K_{k,1})$ as the challenge to $\mathcal{A}$.

Note that, for $i \neq k$, $K_i = e(g_{n+1}, g)^{r + \sum_{j \in S_i} x_j}$, and, if T is the correct value, $K_{k,b} = e(g_{n+1}, G) \cdot e(g_n, h_1) = e(g_{n+1}, g^{\sum_{j \in S_k} x_j}) \cdot e(g_{n+1}, g^r) = e(g_{n+1}, g)^{r + \sum_{j \in S_k} x_j}$.

Guess. $\mathcal{A}$ outputs its guess b' for b. If $b' = b$ the algorithm $\mathcal{B}$ outputs 0 (indicating that $T = e(g_{n+1}, G)$). Otherwise, it outputs 1 (indicating that T is random in $\mathbb{G}_1$). From the above remark, if T is the correct value, $\Pr[\mathcal{B} = 1] = \Pr[b' = b] = (\mathbf{Adv}^{\mathsf{ind}}(\mathcal{A}) + 1)/2$. However, if T is a random value, $\Pr[\mathcal{B} = 1] = 1/2$: $\mathbf{Adv}^{\mathsf{dbdhe}}(\mathcal{B}) = \mathbf{Adv}^{\mathsf{ind}}(\mathcal{A})/2$.

□

5 Multi-Channel Broadcast Encryption II – MIBE_2

We now improve the previous scheme to allow encryption and decryption queries. To this aim, we will need a random oracle.

5.1 Dummy-Helper Technique

First, in order to achieve semantic security, we still have to embed the critical element from the $\mathsf{n}-\mathsf{BDHE}$ instance in the challenge header related to the specific target set S_k. In the previous scheme, it was implicitly embedded in the $X_{i_{k,j}}$, or at least in one of them. But then, if this element is involved in a decryption query, the simulator cannot answer, hence the limitation for the adversary not to ask decryption queries. For the same reason, it was not possible to simulate encryption queries with this critical value.

Using a random oracle, it is possible to embed this element at the challenge time only, and then, instead of a deterministic $\sum_{i \in S_j} x_i$ one can use a random y_j implicitly defined by Y_j given by a random oracle. The knowledge of the discrete logarithm y_j (excepted in the challenge ciphertext), the simulator is able to answer all encryption queries, but this is still not enough to answer decryption queries: the simulator has no idea about the random scalar r involved in the ciphertext, whereas it as to compute $e(g_{n+1}, g)^r$. But this can be done by adding a dummy set for which the session key can be computed by the simulator. In this case, we apply the *dummy-helper technique* to prove the security.

5.2 Description

Setup(λ)**:** it takes as input the security parameter λ, and generates the global parameters param of the system as follows: Let $\mathbb{G}$ be a bilinear group of prime order p; pick a random generator $g \in \mathbb{G}$ and a random scalar $\alpha \in \mathbb{Z}_p$; compute $g_i = g^{\alpha^i} \in \mathbb{G}$ for $i = 1, 2, \ldots, 2n$; pick a random scalar $\gamma \in \mathbb{Z}_p$ and set $v = g^\gamma \in \mathbb{G}$ and $d_n = v^{\alpha^n}$. The algorithm also uses a random oracle $\mathcal{H}$ onto $\mathbb{G}$.
The master key is $\mathsf{MSK} = (g, v, \alpha, \gamma)$, the private encryption key is $\mathsf{EK} = \mathsf{MSK}$ and the public global parameters are $(g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, d_n)$, whereas the private decryption key of user $i \in \{1, \ldots, n\}$ is $d_i = v^{\alpha^i}$. These decryption keys are sent by the **Extract** algorithm.

Encrypt$(S_1, \ldots, S_m, \mathsf{EK})$**:** Pick a random scalar $r \in \mathbb{Z}_p$; set $S_{m+1} = \{n\}$, for each set S_i, for $i = 1, \ldots, m+1$ compute $Y_i = \mathcal{H}(i, g^r)$ ($Y_i = g^{y_i}$, for some unknown scalar y_i), and

$$K_i = e(g_{n+1}, Y_i) \cdot e(g_{n+1}, g)^r = e(g_{n+1}, g)^{r+y_i}, \quad i = 1, \ldots, m+1$$

Eventually compute $\mathsf{Hdr} = (C_1, C_2, C_3)$ as follows:

$$C_1 = g^r$$
$$C_2 = \prod_{i=1}^{i=m+1} \left(Y_i^{\gamma + \sum_{j \in S_i} \alpha^{n+1-j}} \cdot \left(v \cdot \prod_{j \in S_i} g_{n+1-j} \right)^r \right) = \prod_{i=1}^{i=m+1} \left(v \cdot \prod_{j \in S_i} g_{n+1-j} \right)^{r+y_i}$$
$$C_3 = \mathcal{H}(C_1, C_2)^r$$

Note that the broadcaster knows both α and γ to compute C_2. It outputs $(\mathsf{Hdr}, K_1, \ldots, K_{m+1})$.

Decrypt$(S_1,\ldots,S_m,\mathsf{Hdr},i,d_i,k)$**:** Set $S_{m+1}=\{n\}$, parse $\mathsf{Hdr}=(C_1,C_2,C_3)$. If $i\in S_k$ then one first checks whether $e(C_1,\mathcal{H}(C_1,C_2))=e(g,C_3)$, computes $Y_i=\mathcal{H}(i,g^r)$, for $i=1,\ldots,m+1$, and computes

$$\begin{aligned}
K_k &= \frac{e(g_i,C_2)}{e(d_i\cdot\prod_{\substack{j\in S_k\\ j\neq i}} g_{n+1-j+i}, C_1\cdot Y_k)\cdot\prod_{\substack{\ell=1\\ \ell\neq k}}^{\ell=m+1} e(d_i\cdot\prod_{j\in S_\ell} g_{n+1-j+i}, C_1\cdot Y_\ell)}\\
&= \frac{e(g^{\alpha^i},\prod_{\ell=1}^{\ell=m+1}(v\cdot\prod_{j\in S_\ell} g_{n+1-j})^{r+y_\ell})}{e(v^{\alpha^i}\cdot(\prod_{\substack{j\in S_k\\ j\neq i}} g_{n+1-j})^{\alpha^i},g^{r+y_k})\cdot\prod_{\substack{\ell=1\\ \ell\neq k}}^{\ell=m+1} e(v^{\alpha^i}\cdot(\prod_{j\in S_\ell} g_{n+1-j})^{\alpha^i},g^{r+y_\ell})}\\
&= \frac{e(g^{\alpha^i},(v\cdot\prod_{j\in S_k} g_{n+1-j})^{r+y_k})}{e(v^{\alpha^i}\cdot(\prod_{\substack{j\in S_k\\ j\neq i}} g_{n+1-j})^{\alpha^i},g^{r+y_k})}\cdot\prod_{\substack{\ell=1\\ \ell\neq k}}^{\ell=m+1}\frac{e(g^{\alpha^i},(v\cdot\prod_{j\in S_\ell} g_{n+1-j})^{r+y_\ell})}{e(v^{\alpha^i}\cdot(\prod_{j\in S_\ell} g_{n+1-j})^{\alpha^i},g^{r+y_\ell})}\\
&= \frac{e((v\cdot\prod_{j\in S_k} g_{n+1-j})^{\alpha^i},g^{r+y_k})}{e((v\cdot\prod_{\substack{j\in S_k\\ j\neq i}} g_{n+1-j})^{\alpha^i},g^{r+y_k})}\cdot\prod_{\substack{\ell=1\\ \ell\neq k}}^{\ell=m+1}\frac{e((v\cdot\prod_{j\in S_\ell} g_{n+1-j})^{\alpha^i},g^{r+y_\ell})}{e((v\cdot\prod_{j\in S_\ell} g_{n+1-j})^{\alpha^i},g^{r+y_\ell})}\\
&= e(g_{n+1-i}^{\alpha^i},g^{r+y_k}) = e(g_{n+1},g^{r+y_k}) = e(g_{n+1},g)^{r+y_k}
\end{aligned}$$

Note that $d_i=v^{\alpha^i}, g_{n+1-j+i}=g_{n+1-j}^{\alpha^i}$, and $g_{n+1-i}^{\alpha^i}=g_{n+1}$.

5.3 Security

Theorem 9. *The* MIBE_2 *system achieves the strong selective security under the* DBDHE *assumption in* $\mathbb{G}$*. More precisely, if there are n users,*

$$\mathbf{Adv}^{\mathsf{s-ind}}(t,q_C,q_D,q_E)\le 2\times\mathbf{Adv}^{\mathsf{dbdhe}}(t',n)+2\times\mathbf{Succ}^{\mathsf{cdh}}(t'')+2/p,$$

for $t'\le t+(nq_C+nmq_D+nmq_E)T_e+(mq_D+mq_E)T_p+mq_DT_{lu}+O(1)$ *and* $t''\le t+(q_C+q_D+nmq_E)T_e+(q_D+mq_E)T_p+q_DT_{lu}+O(1)$*, where* T_e,T_p *are the time complexity for computing an exponentiation, a pairings,* T_{lu} *is the time complexity for a look up in a list, and* m *is the maximum number of channels in the system.*

Proof. We organize our proof in three games:

1. **Game 0:** The real strong selective security game between an adversary and a challenger.
2. **Game 1:** This is similar to Game 0 with a following exception: if we denote $\mathsf{Hdr}=(C_1,C_2,C_3)$ the challenge header, then any decryption query on a different header $\mathsf{Hdr}'=(C_1,C_2',C_3')$, but with the same C_1, we answer $\perp$ (*i.e.* invalid ciphertext). We can shown that this exception happens with negligible probability under the CDH assumption.
3. **Game 2:** We can now safely answer all decryption queries $\mathsf{Hdr}'=(C_1,C_2',C_3')$ by $\perp$ and the others using either a valid decryption key or d_n. Using the programmability of the random oracle, and thus the knowledge of the y_i, one can easily simulate the encryption queries. Eventually, the semantic security then relies on the DBDHE assumption.

Game 1: In this game, we know all the secret keys, but answer $\perp$ to a decryption query $\mathsf{Hdr}'=(C_1,C_2',C_3')$, with the same first C_1 as in the challenge header. Our algorithm $\mathcal{B}$ is given a CDH instance $g,A=g^{r^*},B$, and should answer $C=B^{r^*}$. It runs the adversary $\mathcal{A}$:

– since we consider selective attacks only, the target sets are known from the beginning, and $\mathcal{B}$ can thus first generate the challenge header using r^* as random scalar, without knowing it: $C_1 = A$. Since $\mathcal{B}$ knows MSK, and namely α and γ, it can compute the appropriate C_2: $v^{r^*} = A^\gamma$ and $g_i^r = A^{\alpha^i}$. It then programs $\mathcal{H}(C_1, C_2) = g^u$ for a random scalar u and sets $C_3 = A^u$. The triple (C_1, C_2, C_3) is a perfect header;
– answers all the hash queries $\mathcal{H}(A, X)$, for any X, by B^t for some randomly chosen scalar t;
– answers all the other queries with MSK.

Let now assume that $\mathcal{A}$ asks for a valid decryption query $(S'_1, \ldots, S'_{m'+1}, k', \mathsf{Hdr}')$ in which $C'_1 = A$. Since $C'_3 = \mathcal{H}(C_1, C'_2)^{r^*} = B^{r^* \cdot t}$ for a known value t, one can extract $C = B^{r^*} = (C'_3)^{1/t}$, which breaks the CDH assumption. $\mathbf{Succ}^{\mathsf{ind}}(\mathcal{A}) - \mathbf{Succ}_1(\mathcal{A}) \leq \mathbf{Succ}^{\mathsf{cdh}}(\mathcal{B})$.

Game 2: We now assume there exists a selective adversary $\mathcal{A}$ that breaks the semantic security of our scheme while decryption queries with the same C_1 as in the challenge are answered by $\perp$. We build an algorithm $\mathcal{B}$ that has twice the advantage in deciding the DBDHE in $\mathbb{G}$. As said above, the programmability of the random oracle will help simulating the encryption queries, and the dummy set will help answering the decryption queries. In game 2.1, the algorithm $\mathcal{B}$ is defined as follows:

Init. Algorithm $\mathcal{B}$ first takes as input a DBDHE instance $(g, G, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, T)$ where $T = e(g_{n+1}, G)$. It implicitly defines α: $g_i = g^{\alpha^i}$. $\mathcal{B}$ then runs $\mathcal{A}$ to receive m^* sets $S_1^*, \ldots, S_{m^*}^*$ and an index k^* that $\mathcal{A}$ wishes to be challenged on. Note that $n \notin S_{k^*}^*$ because the decryption key d_n is public. $\mathcal{B}$ makes use of a random oracle $\mathcal{H}$ which output is a random element in $\mathbb{G}$, and a hash List is initially set empty list, to store all the query-answer, with additional information, when possible. Namely, for a query q, with answer $Y = g^y$, the tuple (q, Y, y) is stored. Sometimes, y will not be known, and thus replaced by $\perp$.

Setup. $\mathcal{B}$ needs to generate the public global parameters and decryption keys d_i, $i \notin S_{k^*}^*$: it chooses a random $u \in \mathbb{Z}_p$ and sets $v \overset{\mathsf{def}}{=} g^u / \prod_{j \in S_{k^*}^*} g_{n+1-j}$. It then computes

$$d_i \overset{\mathsf{def}}{=} g_i^u / \prod_{j \in S_{k^*}^*} g_{n+1-j+i} = g^{u \cdot \alpha^i} \cdot \left(\prod_{j \in S_{k^*}^*} g_{n+1-j} \right)^{-\alpha^i} = v^{\alpha^i}$$

Eventually, $\mathcal{B}$ gives $\mathcal{A}$ the public global parameters $(g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, d_n)$.

Phase 1. Since we now allow encryption and decryption queries, let show how they can be answered. We first start by the hash queries:

1. There are two kinds of useful hash queries, $(j, u) \in \mathbb{Z}_p \times \mathbb{G}$ or $(u_1, u_2) \in \mathbb{G}^2$. But for any query q, if it has already been asked, the same answer is sent back. Otherwise, $\mathcal{B}$ chooses a random scalar $y \overset{\$}{\leftarrow} \mathbb{Z}_p$ and sets $\mathcal{H}(q) = g^y$. It appends the appropriate tuple (q, g^y, y) to the hash List.
2. For an encryption query $(S_1, S_2, \ldots, S_m)$, $\mathcal{B}$ makes the ciphertext as follows: it first chooses a random scalar $r \in \mathbb{Z}_p$ and sets $S_{m+1} = \{n\}$, and $Y_i = \mathcal{H}(i, g^r) = g^{y_i}$ for $i = 1, \ldots, m+1$: y_i is obtained from the hash List. To generate $\mathsf{Hdr} = (C_1, C_2, C_3)$, $\mathcal{B}$ sets $C_1 = g^r$, and computes

$$C_2 = \prod_{i=1}^{m+1} \left(v \cdot \prod_{j \in S_i} g_{n+1-j} \right)^{r + y_i} \qquad C_3 = \mathcal{H}(C_1, C_2)^r$$

   and $K_i = e(g_n, g_1)^{r + y_i}$, for $i = 1, \ldots, m+1$.
3. For a decryption query $(S_1, \ldots, S_{m+1}, \mathsf{Hdr}, i, k)$ in the name of user $i \in S_k$, $\mathcal{B}$ decrypts as follows: it first checks whether $S_k \subseteq S_{k^*}^*$ or not. In the negative case, it finds $j \in S_k \backslash S_{k^*}^*$, and using d_j it can decrypt as the decryption oracle would do; in the positive case

- $\mathcal{B}$ uses d_n to decrypt, using the decryption oracle, and obtain $K_{m+1} = e(g_{n+1}, g)^{r+y_{m+1}}$;
- $\mathcal{B}$ extracts, from the hash List for $\mathcal{H}(m+1, C_1)$, the value y_{m+1}, and computes

$$L = \frac{K_{m+1}}{e(g_{n+1}, g)^{y_{m+1}}} = e(g_n, g_1)^r$$

- $\mathcal{B}$ extracts, from the hash List for $\mathcal{H}(k, C_1)$, the value y_k, and computes the session key

$$K_k = L \times e(g_n, g_1)^{y_k} = e(g_{n+1}, g)^{r+y_k}$$

Challenge. The challenge has to be generated on the target sets $S_1^*, \ldots, S_{m^*}^*$, with the index k^* for the indistinguishability of the key:

- $\mathcal{B}$ first chooses a random scalar $r^* \in \mathbb{Z}_p$ and sets $h = g^{r^*}$, and $h_i = g_i^{r^*}$ for $i = 1, \ldots, n$;
- it chooses a random scalar $z^* \in \mathbb{Z}_p$ and sets $\mathcal{H}(k^*, h) = Y_{k^*}^* = G/g^{z^*}$, which is the value $Y_{k^*}^* = g^{y_{k^*}^*}$ for an unknown $y_{k^*}^*$. The tuple $((k^*, h), Y_{k^*}^*, \perp)$ is appended to the hash List;
- $\mathcal{B}$ asks for the other values $Y_i^* = \mathcal{H}(i, h) = g^{y_i^*}$, for $i = 1, \ldots, k^*-1, k^*+1, \ldots, m^*+1$

Note that $S_{m^*+1}^* = \{n\}$, then $\mathcal{B}$ generates $\mathsf{Hdr}^* = (C_1^*, C_2^*, C_3^*)$ by setting $C_1^* = h$ and $C_3^* = \mathcal{H}(C_1^*, C_2^*)^{r^*}$, where

$$\begin{aligned}
C_2^* &= \left(h^u \cdot (Y_{k^*}^*)^u\right) \prod_{\substack{\ell=1\\ \ell\neq k^*}}^{\ell=m^*} \left(h^u \cdot \left(\frac{\prod_{j\in S_\ell^*} h_{n+1-j}}{\prod_{j\in S_{k^*}^*} h_{n+1-j}} \right) \left(v \prod_{j\in S_\ell^*} g_{n+1-j} \right)^{y_\ell^*} \right) \\
&= (g^u)^{r^*+y_{k^*}^*} \cdot \prod_{\substack{\ell=1\\ \ell\neq k^*}}^{\ell=m^*} \left(\frac{g^u}{\prod_{j\in S_{k^*}^*} g_{n+1-j}} \right)^{r^*} \left(\prod_{j\in S_\ell^*} g_{n+1-j} \right)^{r^*} \left(v \prod_{j\in S_\ell^*} g_{n+1-j} \right)^{y_\ell^*} \\
&= \left(v \prod_{j\in S_{k^*}^*} g_{n+1-j} \right)^{r^*+y_{k^*}^*} \prod_{\substack{\ell=1\\ \ell\neq k^*}}^{\ell=m^*} \left(v \prod_{j\in S_\ell^*} g_{n+1-j} \right)^{r^*} \left(v \prod_{j\in S_\ell^*} g_{n+1-j} \right)^{y_\ell^*} \\
&= \prod_{\ell=1}^{\ell=m^*} \left(v \prod_{j\in S_\ell^*} g_{n+1-j} \right)^{r^*+y_\ell^*}
\end{aligned}$$

To generate the session keys, $\mathcal{B}$ first computes

$$K_i^* = e(g_n, g_1)^{y_i^*} \cdot e(g_n, h_1) = e(g_{n+1}, g)^{r^*+y_i^*}, \qquad i \neq k^*.$$

It then randomly chooses a bit $b \in \{0,1\}$, picks a random $K_{k^*,1-b}^*$ in $\mathbb{G}$ and sets

$$K_{k^*,b}^* = \frac{T \cdot e(g_n, h_1)}{e(g_{n+1}, g^{z^*})}$$

It gives $(\mathsf{Hdr}^*, K_1^*, \ldots, K_{k^*-1}^*, K_{k^*+1}, \ldots, K_{m^*+1}^*, K_{k^*,0}^*, K_{k^*,1}^*)$ as the challenge to $\mathcal{A}$.
Note that since $T = e(g_{n+1}, G)$, with $G = Y_{k^*}^* g^{z^*}$,

$$K_{k^*,b}^* = \frac{e(g_{n+1}, Y_{k^*}^* g^{z^*}) \cdot e(g_n, h_1)}{e(g_{n+1}, g^{z^*})} = e(g_{n+1}, g)^{y_{k^*}^*} \cdot e(g_{n+1}, g)^{r^*} = e(g_{n+1}, g)^{r^*+y_{k^*}^*}$$

Phase 2. $\mathcal{B}$ responds as in the first phase. Note that, if $\mathcal{A}$ asks a decryption query with $C_1 = C_1^*$, $\mathcal{B}$ simply answers $\perp$.

In this game 2.1, the advantage of $\mathcal{A}$ is unchanged, except in case of problem during the programmation of $\mathcal{H}$, which is required once only, and the query has already been asked with probability $1/p$: $\mathbf{Succ}_1(\mathcal{A}) - \mathbf{Succ}_{2.1}(\mathcal{A}) \leq 1/p$. In a game 2.2, we replace T by a random element in $\mathbb{G}$: $\mathbf{Succ}_{2.2}(\mathcal{A}) = 1/2$, whereas $\mathbf{Succ}_{2.1}(\mathcal{A}) - \mathbf{Succ}_{2.2}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{dbdhe}}(\mathcal{B})$.

As a consequence,

$$\mathbf{Succ}^{\mathsf{s-ind}}(\mathcal{A}) \leq \mathbf{Succ}^{\mathsf{cdh}}(\mathcal{B}_1) + \mathbf{Adv}^{\mathsf{dbdhe}}(\mathcal{B}_2) + 1/p + 1/2,$$

where $\mathcal{B}_i$ denotes the simulator $\mathcal{B}$ in Game i. $\square$

6 Conclusion

We initiate the new research line on multi-channel broadcast encryption and propose two efficient schemes with constant-size ciphertexts, while computationally similar to the original BGW scheme in the single-channel setting. We leave some challenging open problems:

– While privacy concerns imply independent keys for all the channels a user subscribed to, this however also leads to large decryption keys for users (linear in the number of channels). One could prefer to have shorter or even constant size keys, sacrificing on privacy, contrary to our priority goal.
– Our first scheme achieves the basic selective security level in the standard model while our second scheme achieves the strong selective security level, which resists to both CPA and CCA, but in the random oracle model. Ruling out the random oracle seems quite challenging because of the implicit relations between session keys.

References

1. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, Nov. 1993.
2. D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, Aug. 2005.
3. C. Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In K. Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 200–215. Springer, Dec. 2007.
4. Y. Dodis and N. Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In Y. Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 100–115. Springer, Jan. 2003.
5. A. Fiat and M. Naor. Broadcast encryption. In D. R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, Aug. 1994.
6. C. Gentry and B. Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In A. Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 171–188. Springer, Apr. 2009.
7. A. B. Lewko, A. Sahai, and B. Waters. Revocation systems with very small private keys. In *2010 IEEE Symposium on Security and Privacy*, pages 273–285. IEEE Computer Society Press, May 2010.

8. D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, Aug. 2001.
9. D. H. Phan, D. Pointcheval, S. F. Shahandashti, and M. Strefler. Adaptive CCA broadcast encryption with constant-size secret keys and ciphertexts. IACR Cryptology ePrint Archive 2012: 216, 2012. `http://eprint.iacr.org/2012/216.pdf`.
10. D. H. Phan, D. Pointcheval, and M. Strefler. Security notions for broadcast encryption. In J. Lopez and G. Tsudik, editors, *ACNS 11: 9th International Conference on Applied Cryptography and Network Security*, volume 6715 of *Lecture Notes in Computer Science*, pages 377–394. Springer, June 2011.
11. B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In S. Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer, Aug. 2009.