

# A New Technique for Compacting Ciphertext in Multi-Channel Broadcast Encryption and Attribute-Based Encryption

S. Canard<sup>a</sup>, D.H. Phan<sup>b</sup>, D. Pointcheval<sup>c,e</sup>, V.C. Trinh<sup>d,\*</sup>

<sup>a</sup>*Orange Labs, Applied Crypto Group, Caen, France*

<sup>b</sup>*University of Limoges, Limoges, France*

<sup>c</sup>*Département d'informatique de l'ENS, École normale supérieure, CNRS, PSL Research University, 75005 Paris, France*

<sup>d</sup>*Hong Duc University, Thanh Hoa, Viet Nam*

<sup>e</sup>*INRIA*

---

## Abstract

Standard Broadcast Encryption (BE) and Attribute-Based Encryption (ABE) aim at sending a content to a large arbitrary group of users at once. Regarding Broadcast Encryption, 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. We consider the 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* or MCBE: one can hope for a much shorter global overhead and a much shorter zapping time since the decoder already has the information to decrypt any available channel at once. Regarding Attribute-Based Encryption, a scheme with constant-size ciphertext is still a challenging task.

In this paper, we introduce a new technique of optimizing the ciphertext-size for both MCBE and ABE schemes.

**Keywords:** *broadcast encryption, multi-channel broadcast encryption, attribute-based encryption.*

---

\*Principal corresponding author

Email address: [trinhvietcuong@hdu.edu.vn](mailto:trinhvietcuong@hdu.edu.vn) (V.C. Trinh)

## 1. Introduction

**MULTI-CHANNEL BROADCAST ENCRYPTION.** Broadcast encryption [19] aims at protecting a content sent to a large arbitrary group of users (called the target set). It has been widely and deeply studied as it is a core primitive for many concrete applications. Until now, the standard model for broadcast encryption only deals with one single content and one single target set. This has a shortcoming in practice. Let us consider a concrete example: a pay-TV system which contains many channels, each of them deals with a different set of privileged users. Applying an independent broadcast encryption scheme for each channel results in an inefficient global system: the bandwidth, or header size, linearly grows in the number of channels, which could become very large. Moreover, in case of zapping from one channel to another channel, one has to start from scratch, and 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 bandwidth and zapping-time problems lead to new efficiency criteria, with a common solution: a broadcast encryption with a short *global header* for multiple channels. The problem of optimizing the bandwidth already appeared in the context of classical (one-channel) broadcast encryption: a broadcast encryption can trivially be constructed from any encryption scheme, by encrypting the session key under each user's key. But this induces a cost that is linear in the number of users. It took more than a decade from the introduction of the primitive [19] to come up with an optimal solution: without considering the description of the target set, the header is of constant size [9]. This solution (hereafter called the BGW scheme) is particularly interesting even if it is still not practical, due to the high decryption complexity: the latter is indeed linear in the size of the target set.

Our new MCBE primitive, for Multi-Channel Broadcast Encryption, addresses both the bandwidth and zapping-time problems. In the following, we show that it is possible to solve these problems in an optimal way: a constant-size global header, independently of the number of channels (and of the users too).

**ATTRIBUTE-BASED ENCRYPTION.** Modern applications sometimes need more sophisticated access control. Addressing the problem of encrypting a message with a specific decryption policy, Sahai and Waters [41] introduced the concept of *attribute-based encryption* (or ABE) in which the encryption and decryption can be based on the attributes of the users. It exists two variants of ABE: *ciphertext-policy* attribute-based encryption (CP-ABE) and *key-policy* attribute-based encryption (KP-ABE). In CP-ABE scheme, the secret key is associated with a set of attributes and the ciphertext is associated with an access policy (structure) over a universe of attributes: a user can then decrypt a given ciphertext if the set of attributes related to his/her secret key satisfies the access policy underlying the ciphertext. In contrast, in KP-ABE scheme, the access policy is for the secret key and the set of attributes is for the ciphertext. As shown in [27], for some applications such as in Pay-TV systems the CP-ABE scheme is more desirable than the KP-ABE scheme, and for which the size of the ciphertext is essential.

### 1.1. Our Contributions

We propose a new technique, which is partially based on the technique from the BGW scheme, to construct MCBE and CP-ABE schemes with constant-size ciphertexts (or constant-size overhead).

#### 1.1.1. Multi-Channel Broadcast Encryption

Firstly, from the proposed technique we directly construct two MCBE schemes with the following properties:

- the first construction is, asymptotically, very competitive with the BGW scheme. In fact, it achieves the constant-size header, independently of the number of channels, 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.* 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 DBDHE, as in the original BGW scheme [9];
- 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* [7]. Our scheme is more efficient than the CCA version of the BGW scheme [9] but our dummy-helper technique actually requires the random oracle model for the security analysis.

*Dummy-helper technique.* In the multi-channel setting, since the session keys of all channels are compacted in only one ciphertext, even if they have to look independent for adversaries, there exists an implicit relation between them, which could be known by the simulator without the whole knowledge of the master key. The *dummy-helper technique* consists in adding a new channel for one additional dummy user. We then get the following interesting properties:

1. for the security analysis: 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, if the simulator knows the above implicit relation between the encapsulated keys;
2. in practice: 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. It can then be used to send them the program or ads.

We prove 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 still achieved under the standard assumption DBDHE as for the BGW scheme.

### 1.1.2. Attribute-Based Encryption

Secondly, we construct CP-ABE schemes with constant-size ciphertexts supporting any CNF access policy. For that purpose, we make use of the techniques given in the Junod-Karlov Attribute-Based Broadcast Encryption scheme (or ABBE) [27] to achieve CNF access policy and to fight against attribute collusion, and from the proposed technique in order to achieve the constant size of the ciphertext.

More precisely, we present two private CP-ABE schemes with the following properties:

- both schemes achieve the constant-size ciphertext. The key size is linear in the maximal number of attributes in the system. Regarding the access policy, both schemes support restricted CNF access policy in the sense that they introduce a parameter  $k_{max}$  in which each attribute can only appear  $k_{max}$  times in the access formula used during the encryption phase. The key size is then larger than a factor of  $k_{max}$ ;
- both of our schemes are naturally based on the use of an asymmetric bilinear pairing, contrary to previous work based on symmetric pairings (even if a generic construction [1] can permit to transform them into the asymmetric case);
- our first scheme achieves basic selective security in the generic group model, *i.e.* adversaries can only know the decryption keys of corrupted users and the challenge header in the selective model;
- our second scheme improves the first one regarding the security since it achieves selective CCA security in the generic group model, *i.e.* adversaries can know the decryption keys of corrupted users, the challenge header and can ask encryption queries as well as decryption queries in the selective model. However, we need to use the random oracle in the security proof.

When comparing to the three interesting existing schemes [5, 2, 3] which achieve the constant-size ciphertext, ours lead to schemes with better key size. However, the schemes in [5, 2, 3] are in the public-key setting and have better security level. We give in Table 1 a comparison among our schemes and some other existing CP-ABE schemes. We moreover argue that our approach to construct constant-size ciphertexts CP-ABE is new and can lead to better schemes in the future. We also notice that using the technique given in [27], we are able to efficiently turn our scheme into an attribute-based broadcast encryption [31] (or ABBE) with a constant-size ciphertext.

	Acce Policy	Ciphertext	Dec key	Enc key	Security Model
[18]	AND-gates	$O(1)$	$O(1)$	$O(n^2)$	standard
[25]	Threshold	$O(1)$	$O(n)$	$O(n)$	S - GG
[27]	CNF	$O(m)$	$O(n)$	$O(n)$	S - GG
[5]	LSSS	$O(1)$	$O(k^4.\ell^4)$	$O(k^2.\ell^2)$	standard
[2]	LSSS	$O(1)$	$O(n.\ell^2)$	$O(n.\ell)$	selective standard
[3]	LSSS	$O(1)$	$O(n.\ell^2)$	$O(n.\ell)$	standard
Ours1	R - CNF	$O(1)$	$O(n.k_{max})$	$O(n.k_{max})$	basic S - GG
Ours2	R - CNF	$O(1)$	$O(n.k_{max})$	$O(1)$	S-CCA-GG+ROM

Figure 1: Comparison among our schemes and some previous schemes, where  $n$  denotes the number of attributes in the system,  $m$  denotes the number of clauses in the CNF access policy,  $k$  denotes the maximal size of an attribute set associated with a secret key,  $\ell$  denotes the maximal number of rows of a span program matrix associated with a ciphertext (fixed at the setup, thus should be  $n$ ); R-CNF means that each attribute can appear only  $k_{max}$  times in an access formula; S-GG denotes selective generic group.

### 1.2. Related work

Broadcast encryption was first described by Fiat and Naor in [19] but has received much attention since the work of Naor, Naor, and Lotspiech [34] in which they presented a private-key subset-cover framework along with a security model and a security analysis. Dodis and Fazio [17] presented the first public-key CCA-secure scheme. Boneh, Gentry, and Waters [9] 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 [16] in the random oracle model. Thereafter, Gentry and Waters [22] 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 [42] and Lewko *et al.* [29] used dual system encryption to achieve adaptive security. A scheme that achieves all desired properties (constant-size ciphertexts, adaptive and CCA security) has been presented in [37] but it relies on rather non-standard assumptions. Broadcast encryption from multilinear map was introduced at [10], but as shown recently in [15, 26, 33], their real feasibility is questionable. Phan, Pointcheval and Streffer [38] gave a global picture of the relations between the security notions for broadcast encryption. However, our setting of multi-channel broadcast encryption goes beyond their considerations, because the adversary could corrupt some users 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 relations into account.

Regarding Attribute-Base Encryption, since their introduction in 2005, one can find a lot of papers proposing ABE schemes [41, 23, 36, 18, 27, 25, 6, 35, 40, 12, 45, 32] to name

a few. The authors in [6, 45] introduced KP-ABE schemes with constant-size ciphertext. The work in [23] extended the Sahai and Waters' work [41] to propose the first schemes supporting finer-grained access control, specified by a Boolean formula. Non-monotonic access structures permitting to handle the negation of attributes has been considered in subsequent works [36, 6, 45]. Thanks to multilinear maps and cryptographic obfuscation, ABE scheme supporting general access structures has been constructed [20], but as already said above, their real feasibility is questionable. Adaptive security for ABE schemes was considered in [28, 12, 4, 44] using composite order groups, and then in [35, 14] using prime order groups. Similarly, dynamic ABE scheme (unbounded attributes) was first investigated in [30] using composite order groups and then in [40] using prime order groups.

Among those constructions, six of them propose CP-ABE schemes with constant-size ciphertext supporting limited access structures. In [18, 13, 39], the access structure is constructed by AND-gates on multi-valued attributes. In [25, 21, 12], the access policy is *threshold*, meaning that there is no distinction among attributes in the access policy: anyone who possesses enough attributes (equal or bigger than a threshold chosen by the sender) will be able to decrypt.

To the best of our knowledge, there exists only two interesting approaches to construct CP-ABE schemes with constant-size ciphertext supporting fine-grained access control. The first one [5] makes use of the conversion technique between ABE and spatial encryption [24]. More precisely, starting from a KP-ABE scheme with constant-size ciphertext, such that [6, 45], one first converts it to a spatial encryption scheme with constant-size ciphertext. Then, from this spatial encryption scheme, one continues to convert it to a CP-ABE schemes with constant size ciphertext. The second approach [2, 3] comes from the pair encodings technique [4, 44], in which they proposed new relaxed but still information theoretic security properties that are sufficient to achieve CP-ABE schemes with constant-size ciphertext and full security. The weakness of these both approaches is that the key size is relatively large.

### 1.3. Organization of the Paper

The paper is organized as follows. We first give in Section 2 some preliminaries that we will use all along the paper. In Section 3, we introduce our proposed technique. Next, in Section 4, we introduce our new concept of multi-channel broadcast encryption, giving the main steps and the expected security properties. We also present our two MCBE constructions from the proposed technique and give the security theorems along with the proofs. Then, in Section 5, we formally recall the concept of ciphertext-policy attribute-based encryption and give two new constructions along with the proofs of security. We finally conclude in Section 6.

## 2. Preliminaries

### 2.1. Bilinear Maps

Let  $\mathbb{G}$ ,  $\tilde{\mathbb{G}}$  and  $\mathbb{G}_T$  denote three finite multiplicative Abelian groups of large prime order  $p > 2^\lambda$  where  $\lambda$  is the security parameter. Let  $g$  be a generator of  $\mathbb{G}$  and  $\tilde{g}$  be a generator of  $\tilde{\mathbb{G}}$ .

An admissible bilinear map is a function  $e : \mathbb{G} \times \tilde{\mathbb{G}} \rightarrow \mathbb{G}_T$ , which verifies the following properties for all  $a, b \in \mathbb{Z}_p$ :

1.  $e(g^a, \tilde{g}^b) = e(g, \tilde{g})^{ab}$ ,
2.  $e(g^a, \tilde{g}^b) = 1$  iff  $a = 0$  or  $b = 0$ ,
3.  $e(g^a, \tilde{g}^b)$  is efficiently computable.

If such a function exists, we say that  $(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e)$  is a bilinear map group system. We say that the bilinear map group system is

1. Type 1 Pairing, if  $\mathbb{G} = \tilde{\mathbb{G}}$ ;
2. Type 2 Pairing, if  $\mathbb{G} \neq \tilde{\mathbb{G}}$  but there is an efficiently computable homomorphism  $\phi : \tilde{\mathbb{G}} \rightarrow \mathbb{G}$ ;
3. Type 3 Pairing, if  $\mathbb{G} \neq \tilde{\mathbb{G}}$  but there is no efficiently computable homomorphism between  $\tilde{\mathbb{G}}$  and  $\mathbb{G}$ .

### 2.2. CDH and DBDHE Assumptions

We first recall the definition of the classical Computational Diffie-Hellman (CDH) assumption in the Type 3 Pairings:

**Definition 1** (CDH Assumption in Type 3 Pairings). *The  $(t, \epsilon)$ -CDH assumption says that for any  $t$ -time adversary  $\mathcal{A}$  that is given  $(g, g^r, h, \tilde{g}, \tilde{g}^r) \in \mathbb{G}^3 \times \tilde{\mathbb{G}}^2$ , its probability to output  $h^r$  is bounded by  $\epsilon$ :*

$$\text{Succ}^{\text{cdh}}(\mathcal{A}) = \Pr[\mathcal{A}(g, g^r, h, \tilde{g}, \tilde{g}^r) = h^r] \leq \epsilon.$$

Stronger assumptions have been introduced by Boneh-Gentry-Waters [9]. They both imply the above CDH assumption, here we describe it in the Type 3 Pairings.

**Definition 2** (BDHE Assumption in Type 3 Pairings). *The  $(t, n, \epsilon)$ -BDHE assumption says that for any  $t$ -time adversary  $\mathcal{A}$  that is given  $L = (g, g^{\alpha^1}, \dots, g^{\alpha^n}, g^{\alpha^{n+2}}, \dots, g^{\alpha^{2n}}, G, \tilde{g}, \tilde{g}^{\alpha^1}, \dots, \tilde{g}^{\alpha^n}, \tilde{g}^{\alpha^{n+2}}, \dots, \tilde{g}^{\alpha^{2n}}) \in \mathbb{G}^{2n+2} \times \tilde{\mathbb{G}}^{2n+1}$ , its probability to output  $e(G, \tilde{g})^{\alpha^{n+1}} \in \mathbb{G}_T$  is bounded by  $\epsilon$ :*

$$\text{Succ}^{\text{bdhe}}(\mathcal{A}) = \Pr[\mathcal{A}(L) = e(G, \tilde{g})^{\alpha^{n+1}}] \leq \epsilon.$$

**Definition 3** (DBDHE Assumption in Type 3 Pairings). *The  $(t, n, \epsilon)$ -DBDHE assumption says that for any  $t$ -time adversary  $\mathcal{A}$  that is given  $L \in \mathbb{G}^{2n+2} \times \tilde{\mathbb{G}}^{2n+1}$ , and a candidate to the BDHE problem, that is either  $e(G, \tilde{g})^{\alpha^{n+1}} \in \mathbb{G}_T$  or a random value  $T$ , cannot distinguish the two cases with advantage greater than  $\epsilon$ :*

$$\text{Adv}^{\text{dbdhe}}(\mathcal{A}) = \left| \frac{\Pr[\mathcal{A}(L, e(G, \tilde{g})^{\alpha^{n+1}}) = 1] - \Pr[\mathcal{A}(L, T) = 1]}{2} \right| \leq \epsilon.$$

### 2.3. $(P, Q, R, f)$ -GDDHE Assumptions

We finally recall the generalization of the Diffie-Hellman exponent assumption in Type 3 Pairings bilinear map group system (which was first introduced in [8]). We assume that there exists an admissible asymmetric bilinear map  $e : \mathbb{G} \times \tilde{\mathbb{G}} \rightarrow \mathbb{G}_T$ , meaning that for all  $a, b \in \mathbb{Z}_p$ , (i)  $e(g^a, \tilde{g}^b) = e(g, \tilde{g})^{ab}$ , (ii)  $e(g^a, \tilde{g}^b) = 1$  iff  $a = 0$  or  $b = 0$ , and (iii)  $e(g^a, \tilde{g}^b)$  is efficiently computable. In the sequel, the set  $(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e)$  is called a bilinear map group system.

Let  $(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e)$  be a bilinear map group system and  $g \in \mathbb{G}$  (resp.  $\tilde{g} \in \tilde{\mathbb{G}}$ ) be a generator of  $\mathbb{G}$  (resp.  $\tilde{\mathbb{G}}$ ). We set  $g_T = e(g, \tilde{g}) \in \mathbb{G}_T$ . Let  $s, n$  be positive integers and  $P, Q, R \in \mathbb{F}_p[X_1, \dots, X_n]^s$  be three  $s$ -tuples of  $n$ -variate polynomials over  $\mathbb{F}_p$ . Thus,  $P, Q$  and  $R$  are just three lists containing  $s$  multivariate polynomials each. We write  $P = (p_1, p_2, \dots, p_s)$ ,  $Q = (q_1, q_2, \dots, q_s)$ ,  $R = (r_1, r_2, \dots, r_s)$  and impose that  $p_1 = q_1 = r_1 = 1$ . For any function  $h : \mathbb{F}_p \rightarrow \Omega$  and any vector  $(x_1, \dots, x_n) \in \mathbb{F}_p^n$ ,  $h(P(x_1, \dots, x_n))$  stands for  $(h(p_1(x_1, \dots, x_n)), \dots, h(p_s(x_1, \dots, x_n))) \in \Omega^s$ . We use a similar notation for the  $s$ -tuples  $Q$  and  $R$ . Let  $f \in \mathbb{F}_p[X_1, \dots, X_n]$ . It is said that  $f$  depends on  $(P, Q, R)$ , which denotes  $f \in \langle P, Q, R \rangle$ , when there exists a linear decomposition:

$$f = \sum_{1 \leq i, j \leq s} a_{i,j} \cdot p_i \cdot q_j + \sum_{1 \leq i \leq s} c_i \cdot r_i,$$

where  $a_{i,j}, c_i \in \mathbb{Z}_p$ .

Let  $P, Q, R$  be as above and  $f \in \mathbb{F}_p[X_1, \dots, X_n]$ . The  $(P, Q, R, f)$  - GDHE problem is defined as follows.

**Definition 4.**  $(P, Q, R, f)$ -GDHE [8]

Given the tuple  $H(x_1, \dots, x_n) = (g^{P(x_1, \dots, x_n)}, \tilde{g}^{Q(x_1, \dots, x_n)}, g_T^{R(x_1, \dots, x_n)}) \in \mathbb{G}^s \times \tilde{\mathbb{G}}^s \times \mathbb{G}_T^s$  compute  $g_T^{f(x_1, \dots, x_n)}$ .

**Definition 5.**  $(P, Q, R, f)$ -GDDHE [8]

Given the tuple  $H(x_1, \dots, x_n) = (g^{P(x_1, \dots, x_n)}, \tilde{g}^{Q(x_1, \dots, x_n)}, g_T^{R(x_1, \dots, x_n)}) \in \mathbb{G}^s \times \tilde{\mathbb{G}}^s \times \mathbb{G}_T^s$  and  $T \in \mathbb{G}_T$  decide whether  $T = g_T^{f(x_1, \dots, x_n)}$ .

## 3. The Proposed Technique

In this section, we first give the overview of the BGW scheme, we will then rely on the technique from this scheme to describe our technique. For practical use, we use Type 3 Pairings to describe our technique.

### 3.1. BGW Overview

The main difference with the BGW scheme [9] is that we make use of an asymmetric Pairing of Type 3.

**Setup**( $\lambda$ ): Let  $\mathbb{G}$  be a bilinear group of prime order  $p$ . The algorithm first picks a random generator  $g \in \mathbb{G}$ ,  $\tilde{g} \in \tilde{\mathbb{G}}$  and a random scalar  $\alpha \in \mathbb{Z}_p$ . It computes  $g_i = g^{\alpha^i} \in \mathbb{G}$  and  $\tilde{g}_i = \tilde{g}^{\alpha^i} \in \mathbb{G}$  for  $i = 1, 2, \dots, n, n+2, \dots, 2n$ . Next, it picks a random scalar  $\gamma \in \mathbb{Z}_p$  and sets  $v = g^\gamma \in \mathbb{G}$  and  $\tilde{v} = \tilde{g}^\gamma \in \tilde{\mathbb{G}}$ .

The public key is  $\mathbf{EK} = (\tilde{g}, g_1, \tilde{g}_1, \dots, g_n, \tilde{g}_n, g_{n+2}, \tilde{g}_{n+2}, \dots, g_{2n}, \tilde{g}_{2n}, v, \tilde{v})$ , whereas the private decryption key of user  $i \in \{1, \dots, n\}$  is  $\tilde{d}_i = \tilde{v}^{\alpha^i}$ . These decryption keys are sent by the **Extract** algorithm.

**Encrypt**( $S, \mathbf{EK}$ ): Pick a random scalar  $r \in \mathbb{Z}_p$ , and set  $K = e(g_{n+1}, \tilde{g})^r$ , where  $e(g_{n+1}, \tilde{g})$  can be computed as  $e(g_n, \tilde{g}_1)$  from  $\mathbf{EK}$ .

Next, set:  $\mathbf{Hdr} = (g^r, (v \cdot \prod_{j \in S} g_{n+1-j})^r)$ , and output  $(\mathbf{Hdr}, K)$ .

**Decrypt**( $S, \mathbf{Hdr}, i, \tilde{d}_i, \mathbf{EK}$ ): Parse  $\mathbf{Hdr} = (C_1, C_2)$ , output  $K = e(C_2, \tilde{g}_i) / e(C_1, \tilde{d}_i \cdot \prod_{j \in S, j \neq i} \tilde{g}_{n+1-j+i})$ .

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

**Setup**( $\lambda$ ): As in the BGW scheme.

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

$$K_1 = e(g_{n+1}, \tilde{g})^{r_1}, \dots, K_m = e(g_{n+1}, \tilde{g})^{r_m}$$

$$\mathbf{Hdr} = \left( (g^{r_1}, (v \cdot \prod_{j \in S_1} g_{n+1-j})^{r_1}), \dots, (g^{r_m}, (v \cdot \prod_{j \in S_m} g_{n+1-j})^{r_m}) \right)$$

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

### 3.2. Intuition of the Proposed Technique

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  $m + 1$ :

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

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  $e(g_{n+1}, \tilde{g})^r$ . 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  $i = 1, \dots, n$ , and to adapt the session key and **Hdr** using scalars  $x_i$ , where  $X_i = g^{x_i}$ , for  $i = 1, \dots, n$ ,

$$K_1 = e(g_{n+1}, \tilde{g})^{r + \sum_{j \in S_1} x_j}, \dots, K_m = e(g_{n+1}, \tilde{g})^{r + \sum_{j \in S_m} x_j},$$

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

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:

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

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, based on this technique we can directly construct a **MCBE** scheme with constant-size ciphertext. Regarding the security, however, this **MCBE** scheme (we name **MCBE<sub>1</sub>**) only can achieve the basic selective security (not allow the adversary to ask decryption and encryption queries). Limitation not to ask decryption nor encryption queries is quite strong, and is the main drawback of the first scheme **MCBE<sub>1</sub>**. And thus, we provide a second construction **MCBE<sub>2</sub>** 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

In this section, we first formally describe the model for a multi-channel broadcast encryption system, we next rely on the proposed technique to construct two **MCBE** schemes in Type 3 Pairings. We also give the security analysis of those schemes at the end of the section.

### 4.1. Definition of *MCBE*

#### 4.1.1. Syntax

Formally, such a system consists of four probabilistic algorithms:

**Setup**( $\lambda$ ): takes as input the security parameter  $\lambda$ , it generates the global parameters **param** of the system, including  $n$  the maximal number of users (receivers are implicitly represented by integers in  $\{1, \dots, n\}$ ), and returns a master key **MSK** and an encryption key **EK**. If the scheme allows public encryption, **EK** is public, otherwise **EK** is kept private, and can be seen as part of **MSK**.

**Extract**( $i, \text{MSK}$ ): takes as input the user's index  $i$ , together with the master key, and outputs the user's private key  $d_i$ .

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

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

For correctness, we require that for all  $S_i \subseteq \{1, \dots, n\}$  and  $j \in S_i$ , if  $(\text{EK}, \text{MSK}) \leftarrow \text{Setup}(\lambda)$ ,  $d_j \leftarrow \text{Extract}(j, \text{MSK})$  and  $(\text{Hdr}, K_1, \dots, K_m) \leftarrow \text{Encrypt}(S_1, S_2, \dots, S_m, \text{EK})$ , one then should get  $K_i = \text{Decrypt}(S_1, S_2, \dots, S_m, \text{Hdr}, j, d_j, i)$ .

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

#### 4.1.2. 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, in the Real-or-Random setting:

**Setup.** The challenger runs the **Setup** algorithm to generate the global parameters  $\text{param}$  of the system, and returns a master key  $\text{MSK}$  and an encryption key  $\text{EK}$ . If the scheme is asymmetric,  $\text{EK}$  is given to  $\mathcal{A}$ , otherwise it is part of the  $\text{MSK}$ , and thus kept secret. Corruption and decryption lists  $\Lambda_C, \Lambda_D$  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**( $i, \text{MSK}$ ) and forwards the resulting private key to the adversary. The user  $i$  is appended to the corruption list  $\Lambda_C$ ;
2. Decryption query on the full header  $(S_1, \dots, S_m, \text{Hdr})$  with  $u \in \{1, \dots, n\}$  and  $j \in \{1, \dots, m\}$ . The challenger answers with **Decrypt**( $S_1, \dots, S_m, \text{Hdr}, u, d_u, j$ ). The pair  $(\text{Hdr}, S_j)$  is appended to the decryption list  $\Lambda_D$ ;
3. Encryption query (if  $\text{EK}$  is private) for the target sets  $(S_1, S_2, \dots, S_m)$ . The challenger answers with **Encrypt**( $S_1, S_2, \dots, S_m, \text{EK}$ ).

**Challenge.** The adversary  $\mathcal{A}$  outputs  $t$  target sets  $S_1^*, \dots, S_t^* \subseteq \{1, \dots, n\}$  and an index  $j$ , which specifies the attacked target set  $S_j^*$ .

The challenger runs **Encrypt**( $S_1^*, S_2^*, \dots, S_t^*, \text{EK}$ ) and gets  $(\text{Hdr}^*, K_1^*, K_2^*, \dots, K_t^*)$ .

Next, the challenger picks a random  $b \xleftarrow{\$} \{0, 1\}$ . If  $b = 1$ , it picks a random  $K_j^* \xleftarrow{\$} \mathcal{K}$ . It outputs  $(\text{Hdr}^*, K_1^*, \dots, K_t^*)$  to  $\mathcal{A}$ .

Note that if  $b = 0$ ,  $K_j^*$  is the real key, encapsulated in  $\text{Hdr}^*$ , and if  $b = 1$ ,  $K_j^*$  is random, independent of the header.

**Query phase 2.** The adversary  $\mathcal{A}$  continues to adaptively ask 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  $(\text{Hdr}^*, S_j^*) \notin \Lambda_D$ .

We then denote by  $\text{Succ}^{\text{ind}}(\mathcal{A}) = \Pr[b' = b]$  the probability that  $\mathcal{A}$  wins the game, and its advantage is

$$\begin{aligned} \text{Adv}^{\text{ind}}(\mathcal{A}) &= 2 \times \text{Succ}^{\text{ind}}(\mathcal{A}) - 1 \\ &= \Pr[1 \leftarrow \mathcal{A} | b = 1] - \Pr[1 \leftarrow \mathcal{A} | b = 0]. \end{aligned}$$

**Definition 6** (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, one has  $\text{Adv}^{\text{ind}}(\mathcal{A}) \leq \varepsilon$ . We denote by  $\text{Adv}^{\text{ind}}(t, q_C, q_D, q_E)$  the advantage of the best  $t$ -time adversary.*

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

**Definition 7** (Basic Selective Security). *A multi-channel broadcast encryption scheme is said to be  $(t, \varepsilon, q_C)$ -selectively secure if it is  $(t, \varepsilon, q_C, 0, 0)$ -secure against a selective adversary. We denote by  $\text{Adv}^{\text{b-ind}}(t, q_C)$  the advantage of the best  $t$ -time basic selective adversary.*

Note that in the public-key broadcast setting (where encryption is public), this just excludes decryption queries: we allow CPA adversaries.

**Definition 8** (Strong Selective Security). *A multi-channel broadcast encryption scheme is said to be  $(t, \varepsilon, q_C, q_D, q_E)$ -selectively secure if it is  $(t, \varepsilon, q_C, q_D, q_E)$ -secure against a selective adversary. We denote by  $\text{Adv}^{\text{s-ind}}(t, q_C, q_D, q_E)$  the advantages of the best  $t$ -time strong selective adversaries.*

This definition is much stronger since it not only allows decryption queries in the public setting, but also encryption queries in the private setting.

#### 4.1.3. Disjoint Target Sets

Before presenting our MCBE schemes in details, we want to stress that our solution requires that all the target sets of the distinct channels are disjoint. Fortunately, this is compatible with our target application of Pay-TV: whenever a user subscribes for a new channel, he is given a new key for decrypting that channel, and it is reasonable to consider that the two keys are independent. More formally, in our systems, we assume there are several channels, which are encrypted to independent target sets of users. The users in the appropriate target sets own decryption keys specific to each channel:

- 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 above cases, the target sets are subsets of predetermined and disjoint sets of keys. As a consequence, the target sets  $S_i$  are disjoint too. The drawback is that we have to define many keys in the system.

In a more general setting, in order to limit this number of keys, one could think about sharing keys for several channels. Then, it would make the setting incompatible with our solutions which require disjoint target sets. On the other hand, while reducing the number of keys, it would reduce privacy protection too since one would be able to know which channels are registered by similar users, and derive some profiles. Alternatively, in order to limit the global number of keys, one could reassign 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.

#### 4.2. First Construction – MCBE<sub>1</sub>

Let us now describe formally our first construction MCBE<sub>1</sub>. We will then prove its basic selective security.

##### 4.2.1. Description

**Setup**( $\lambda$ ): The algorithm takes as input the security parameter  $\lambda$ , it generates the global parameters **param** of the system as follows: Let  $\mathbb{G}, \tilde{\mathbb{G}}$  be bilinear groups of prime order  $p$ . The algorithm first picks random generators  $g \in \mathbb{G}, \tilde{g} \in \tilde{\mathbb{G}}$ , and a random  $\alpha \in \mathbb{Z}_p$ . It computes  $g_i = g^{\alpha^i} \in \mathbb{G}, \tilde{g}_i = \tilde{g}^{\alpha^i} \in \tilde{\mathbb{G}}$  for  $i = 1, 2, \dots, n, n+2, \dots, 2n$ . Next, it picks a random  $\gamma \in \mathbb{Z}_p$  and sets  $v = g^\gamma \in \mathbb{G}, \tilde{v} = \tilde{g}^\gamma \in \tilde{\mathbb{G}}$ . It also picks additional random scalars  $x_1, x_2, \dots, x_n \in \mathbb{Z}_p$  and sets  $X_1 = g^{x_1}, X_2 = g^{x_2}, \dots, X_n = g^{x_n}$ . The master secret key is **MSK** =  $(\alpha, g_{n+1}, x_1, x_2, \dots, x_n)$ , while the encryption key (that is private to the broadcaster) is **EK** =  $(g_{n+1}, x_1, x_2, \dots, x_n)$ . The public global parameters are

$(g, \tilde{g}, v, \tilde{v}, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n}, \tilde{g}_1, \dots, \tilde{g}_n, \tilde{g}_{n+2}, \dots, \tilde{g}_{2n}, X_1, X_2, \dots, X_n)$ , whereas the private decryption key is  $d_i = \tilde{v}^{\alpha^i}$ , for  $i \in \{1, \dots, n\}$ . These decryption keys are sent by the **Extract** algorithm.

We note that if a user registers to  $t$  different channels, he will possess  $t$  different private decryption keys:  $n$  will be the product of the number of users and the number of channels.

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

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

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

**Decrypt** $(S_1, \dots, S_m, \text{Hdr}, i, d_i, k)$ : Parse  $\text{Hdr} = (C_1, C_2)$ . If  $i \in S_k$  then one computes  $K_k$  as in the Figure 2.

$$\begin{aligned} &= \frac{e(C_2, \tilde{g}_i)}{e(C_1 \cdot \prod_{j \in S_k} X_j, d_i \cdot \prod_{\substack{j \in S_k \\ j \neq i}} \tilde{g}_{n+1-j+i}) \cdot \prod_{\substack{\ell=1 \\ \ell \neq k}}^{\ell=m} e(C_1 \cdot \prod_{j \in S_\ell} X_j, d_i \cdot \prod_{j \in S_\ell} \tilde{g}_{n+1-j+i})} \\ &= \frac{e(C_2, \tilde{g}_i)}{e(g^{r + \sum_{j \in S_k} x_j}, d_i \cdot \prod_{\substack{j \in S_k \\ j \neq i}} \tilde{g}_{n+1-j+i}) \cdot \prod_{\substack{\ell=1 \\ \ell \neq k}}^{\ell=m} e(g^{r + \sum_{j \in S_\ell} x_j}, d_i \cdot \prod_{j \in S_\ell} \tilde{g}_{n+1-j+i})} \\ &= \frac{e(\prod_{\ell=1}^{\ell=m} (v \cdot \prod_{j \in S_\ell} g_{n+1-j})^{r + \sum_{j \in S_\ell} x_j}, \tilde{g}^{\alpha^i})}{e(g^{r + \sum_{j \in S_k} x_j}, \tilde{v}^{\alpha^i} \cdot (\prod_{\substack{j \in S_k \\ j \neq i}} \tilde{g}_{n+1-j})^{\alpha^i}) \cdot \prod_{\substack{\ell=1 \\ \ell \neq k}}^{\ell=m} e(g^{r + \sum_{j \in S_\ell} x_j}, \tilde{v}^{\alpha^i} \cdot (\prod_{j \in S_\ell} \tilde{g}_{n+1-j})^{\alpha^i})} \\ &= \frac{e((v \cdot \prod_{j \in S_k} g_{n+1-j})^{r + \sum_{j \in S_k} x_j}, \tilde{g}^{\alpha^i})}{e(g^{r + \sum_{j \in S_k} x_j}, \tilde{v}^{\alpha^i} \cdot (\prod_{\substack{j \in S_k \\ j \neq i}} \tilde{g}_{n+1-j})^{\alpha^i})} \cdot \prod_{\substack{\ell=1 \\ \ell \neq k}}^{\ell=m} \frac{e((v \cdot \prod_{j \in S_\ell} g_{n+1-j})^{r + \sum_{j \in S_\ell} x_j}, \tilde{g}^{\alpha^i})}{e(g^{r + \sum_{j \in S_\ell} x_j}, \tilde{v}^{\alpha^i} \cdot (\prod_{j \in S_\ell} \tilde{g}_{n+1-j})^{\alpha^i})} \\ &= \frac{e(g^{r + \sum_{j \in S_k} x_j}, (\tilde{v} \cdot \prod_{j \in S_k} \tilde{g}_{n+1-j})^{\alpha^i})}{e(g^{r + \sum_{j \in S_k} x_j}, (\tilde{v} \cdot \prod_{\substack{j \in S_k \\ j \neq i}} \tilde{g}_{n+1-j})^{\alpha^i})} \cdot \prod_{\substack{\ell=1 \\ \ell \neq k}}^{\ell=m} \frac{e(g^{r + \sum_{j \in S_\ell} x_j}, (\tilde{v} \cdot \prod_{j \in S_\ell} \tilde{g}_{n+1-j})^{\alpha^i})}{e(g^{r + \sum_{j \in S_\ell} x_j}, (\tilde{v} \cdot \prod_{j \in S_\ell} \tilde{g}_{n+1-j})^{\alpha^i})} \\ &= e(g_{n+1-i}^{\alpha^i}, \tilde{g}^{r + \sum_{j \in S_k} x_j}) = e(g_{n+1}, \tilde{g}^{r + \sum_{j \in S_k} x_j}) = e(g_{n+1}, \tilde{g})^{r + \sum_{j \in S_k} x_j} = K_k \end{aligned}$$

Figure 2:

Here, we used the relations  $d_i = \tilde{v}^{\alpha^i}$ ,  $g_{n+1-j+i} = g_{n+1-j}^{\alpha^i}$ ,  $g_{n+1-i}^{\alpha^i} = g_{n+1}$ , and  $e(g^a, \tilde{g}^b) = e(g^b, \tilde{g}^a)$ .

**Remark 1.** In  $\text{MCBE}_1$ , the encryption key  $\text{EK}$  contains  $g_{n+1}, x_1, x_2, \dots, x_n$  and thus cannot be public: this is a private variant of BGW scheme. Actually,  $g_{n+1}$  is not really required, but the  $x_i$ 's would be enough to break the semantic security, and thus cannot be public either. However, the broadcaster does not need to know  $\alpha$  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).

#### 4.2.2. Security Result

We now prove the semantic security of the  $\text{MCBE}_1$  scheme.

**Theorem 9.** *The  $\text{MCBE}_1$  system achieves the basic selective security under the DBDHE assumption in  $\mathbb{G}, \tilde{\mathbb{G}}$ . More precisely, if there are  $n$  users,*

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

for  $t' \leq t + (mn + nq_C)T_e$  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  $\text{MCBE}_1$  scheme, we build an algorithm  $\mathcal{B}$  that has the same advantage in deciding the DBDHE assumption in  $\mathbb{G}, \tilde{\mathbb{G}}$ . This algorithm  $\mathcal{B}$  proceeds as follows:

**Init.** Algorithm  $\mathcal{B}$  first takes as input a DBDHE instance

$$(g, g^{\alpha^1}, \dots, g^{\alpha^n}, g^{\alpha^{n+2}}, \dots, g^{\alpha^{2n}}, G, \tilde{g}, \tilde{g}^{\alpha^1}, \dots, \tilde{g}^{\alpha^n}, \tilde{g}^{\alpha^{n+2}}, \dots, \tilde{g}^{\alpha^{2n}}, T),$$

where  $T$  is either  $e(G, \tilde{g}^{\alpha^{n+1}})$  or a random element of  $\mathbb{G}_T$ . It defines:  $g_i = g^{\alpha^i}, \tilde{g}_i = \tilde{g}^{\alpha^i}$ .  $\mathcal{B}$  then runs  $\mathcal{A}$ , and since we are in the selective model, it receives  $m$  sets  $S_1, \dots, S_m$  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  $i = 1, \dots, n$ . One chooses a random index  $\eta$  in  $S_k$ , and for  $i \in \{1, \dots, n\} \setminus \{\eta\}$ , one chooses a random scalar  $x_i \in \mathbb{Z}_p$ , and computes  $X_i = g^{x_i}$ . One eventually sets

$$X_\eta \stackrel{\text{def}}{=} G / \prod_{i \in S_k \setminus \{\eta\}} X_i = g^{x_\eta},$$

all the scalars  $x_i$  are known, excepted  $x_\eta$ .  $\mathcal{B}$  next chooses a random  $u \in \mathbb{Z}_p$  and sets

$$v \stackrel{\text{def}}{=} g^u \cdot \left( \prod_{j \in S_k} g_{n+1-j} \right)^{-1} \quad \tilde{v} \stackrel{\text{def}}{=} \tilde{g}^u \cdot \left( \prod_{j \in S_k} \tilde{g}_{n+1-j} \right)^{-1}$$

$\mathcal{B}$  gives  $\mathcal{A}$  the public global parameters:

$$(g, \tilde{g}, v, \tilde{v}, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n}, \tilde{g}_1, \dots, \tilde{g}_n, \tilde{g}_{n+2}, \dots, \tilde{g}_{2n}, X_1, X_2, \dots, X_n)$$

$\mathcal{B}$  has to compute all the private decryption keys  $d_i$  except for  $i \in S_k$ , he sets

$$d_i \stackrel{\text{def}}{=} \tilde{g}_i^u / \left( \prod_{j \in S_k} \tilde{g}_{n+1-j+i} \right) = \tilde{g}^{u \cdot \alpha^i} \cdot \left( \prod_{j \in S_k} \tilde{g}_{n+1-j} \right)^{-\alpha^i} = \tilde{v}^{\alpha^i}$$

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

**Challenge.** To generate the challenge for  $\mathcal{A}$ ,  $\mathcal{B}$  first computes  $\text{Hdr} = (C_1, C_2)$  by setting  $C_1 = h$ , and computes  $C_2$  as in the Figure 3.

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

Figure 3:

Here, 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  $x_\eta$ , that appears in  $h^u \cdot G^u = (g^u)^{r + \sum_{i \in S_k} x_i}$ . To generate session keys,  $\mathcal{B}$  first computes, for all  $i \neq k$ ,  $K_i = e(g_n, \tilde{g}_1)^{\sum_{j \in S_i} x_j}$ .

$e(h_1, \tilde{g}_n)$ , and sets  $K_k = T \cdot e(h_1, \tilde{g}_n)$ . It outputs  $(\text{Hdr}, K_1, \dots, K_m)$  as the challenge to  $\mathcal{A}$ .

Note that, for  $i \neq k$ ,  $K_i = e(g_{n+1}, \tilde{g})^{r + \sum_{j \in S_i} x_j}$ , and, if  $T$  is the correct value,  $K_k = e(G, \tilde{g}_{n+1}) \cdot e(h_1, \tilde{g}_n) = e(g_{n+1}, \tilde{g}^{\sum_{j \in S_k} x_j}) \cdot e(g^r, \tilde{g}_{n+1}) = e(g_{n+1}, \tilde{g})^{r + \sum_{j \in S_k} x_j}$ . If  $T$  is random, the latter is also random.

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

□

#### 4.3. Second Construction – MCB $E_2$

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

##### 4.3.1. Dummy-Helper Technique

First, in order to achieve semantic security, we still have to embed the critical element from the DBDHE instance in the challenge header related to the specific target set  $S_k$ . In the previous scheme, it was implicitly embedded in the  $X_\eta$ , 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. With 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 has to compute  $e(g_{n+1}, \tilde{g})^r$ . 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.

##### 4.3.2. Description

**Setup**( $\lambda$ ): it takes as input the security parameter  $\lambda$ , and generates the global parameters **param** of the system as follows: Let  $\mathbb{G}, \tilde{\mathbb{G}}$  be bilinear groups of prime order  $p$ ; pick random generators  $g \in \mathbb{G}, \tilde{g} \in \tilde{\mathbb{G}}$ , and a random  $\alpha \in \mathbb{Z}_p$ . It computes  $g_i = g^{\alpha^i} \in \mathbb{G}, \tilde{g}_i = \tilde{g}^{\alpha^i} \in \tilde{\mathbb{G}}$  for  $i = 1, 2, \dots, n, n+2, \dots, 2n$ . Next, it picks a random  $\gamma \in \mathbb{Z}_p$  and sets  $v = g^\gamma \in \mathbb{G}, \tilde{v} = \tilde{g}^\gamma \in \tilde{\mathbb{G}}$  and  $d_n = \tilde{v}^{\alpha^n}$ . The algorithm also uses a random oracle  $\mathcal{H}$  onto  $\mathbb{G}$ .

The master key is  $\text{MSK} = (\alpha, \gamma)$ , the private encryption key is  $\text{EK} = \text{MSK}$  and the public global parameters are  $(g, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n}, d_n, \tilde{g}, \tilde{g}_1, \dots, \tilde{g}_n, \tilde{g}_{n+2}, \dots, \tilde{g}_{2n}, \mathcal{H})$ , whereas the private decryption key is  $d_i = \tilde{v}^{\alpha^i}$ , for  $i \in \{1, \dots, n\}$ . These decryption keys are sent by the **Extract** algorithm.

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

$$K_i = e(Y_i, \tilde{g}_{n+1}) \cdot e(g_{n+1}, \tilde{g})^r = e(g_{n+1}, \tilde{g})^{r+y_i},$$

$i = 1, \dots, m+1$ . Eventually compute  $\text{Hdr} = (C_1, \tilde{C}_1, C_2, C_3)$  as follows:

$$\begin{aligned} C_1 &= g^r & \tilde{C}_1 &= \tilde{g}^r \\ C_2 &= \prod_{i=1}^{m+1} \left( Y_i^{\gamma + \sum_{j \in S_i} \alpha^{n+1-j}} \cdot \left( v \cdot \prod_{j \in S_i} g_{n+1-j} \right)^r \right) \\ &= \prod_{i=1}^{m+1} \left( v \cdot \prod_{j \in S_i} g_{n+1-j} \right)^{r+y_i} & C_3 &= \mathcal{H}(C_1, C_2)^r \end{aligned}$$

Note that the broadcaster knows both  $\alpha$  and  $\gamma$  to compute  $C_2$ . It outputs  $(\text{Hdr}, K_1, \dots, K_{m+1})$ .

**Decrypt** $(S_1, \dots, S_m, \text{Hdr}, i, d_i, k)$ : Set  $S_{m+1} = \{n\}$ , parse  $\text{Hdr} = (C_1, \tilde{C}_1, C_2, C_3)$ . If  $i \in S_k$  then one first checks whether  $e(\mathcal{H}(C_1, C_2), \tilde{C}_1) = e(C_3, \tilde{g})$  and  $e(C_1, \tilde{g}) = e(g, \tilde{C}_1)$ , computes  $Y_i = \mathcal{H}(i, g^r)$ , for  $i = 1, \dots, m+1$ , and computes (as in the previous scheme, where the  $Y_j$ 's replace some products of the  $X_i$ 's)

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

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

#### 4.3.3. Security Result

We now prove the semantic security of the  $\text{MCBE}_2$  scheme.

**Theorem 10.** *The  $\text{MCBE}_2$  system achieves the strong selective security under the DBDHE assumption in  $\mathbb{G}, \tilde{\mathbb{G}}$ . More precisely, if there are  $n$  users,*

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

for  $t' \leq t + (nq_C + nmq_D + nmq_E)T_e + (mq_D + mq_E)T_p + mq_D T_{lu}$  and  $t'' \leq t + (q_C + q_D + nmq_E)T_e + (q_D + mq_E)T_p + q_D T_{lu}$ , 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  $\mathbf{Hdr} = (C_1, \tilde{C}_1, C_2, C_3)$  the challenge header, then any decryption query on a different header  $\mathbf{Hdr}' = (C_1, \tilde{C}'_1, C'_2, C'_3)$ , but with the same  $C_1$ , we answer  $\perp$  (*i.e.* invalid ciphertext). We can show that this exception happens with negligible probability under the CDH assumption.
3. **Game 2:** We can now safely answer all decryption queries  $\mathbf{Hdr}' = (C_1, \tilde{C}'_1, C'_2, C'_3)$  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  $\mathbf{Hdr}' = (C_1, \tilde{C}'_1, C'_2, C'_3)$ , with the same first  $C_1$  as in the challenge header. Our algorithm  $\mathcal{B}$  is given a CDH instance  $g, \tilde{g}, A = g^{r^*}, \tilde{A} = \tilde{g}^{r^*}, B$ , 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:  $C_1 = A, \tilde{C}_1 = \tilde{A}$ . Since  $\mathcal{B}$  knows  $\mathbf{MSK}$ , and namely  $\alpha$  and  $\gamma$ , 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 tuple  $(C_1, \tilde{C}_1, C_2, C_3)$  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  $\mathbf{MSK}$ .

Let now assume that  $\mathcal{A}$  asks for a valid decryption query  $(S'_1, \dots, S'_{m'+1}, k', \mathbf{Hdr}')$  in which  $C'_1 = C_1 = A$ . 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.  $\mathbf{Succ}^{\text{ind}}(\mathcal{A}) - \mathbf{Succ}_1(\mathcal{A}) \leq \mathbf{Succ}^{\text{cdh}}(\mathcal{B})$ .

**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  $\mathbb{G}, \tilde{\mathbb{G}}$ . 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  $(g, G, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n}, \tilde{g}, \tilde{g}_1, \dots, \tilde{g}_n, \tilde{g}_{n+2}, \dots, \tilde{g}_{2n}, T)$  where  $T = e(G, \tilde{g}_{n+1})$ . It implicitly defines  $\alpha$ :  $g_i = g^{\alpha^i}, \tilde{g}_i = \tilde{g}^{\alpha^i}$ .  $\mathcal{B}$  then runs  $\mathcal{A}$  to receive  $m^*$  sets  $S_1^*, \dots, S_{m^*}^*$  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  $\mathcal{H}$  whose 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

$$v \stackrel{\text{def}}{=} g^u \cdot \left( \prod_{j \in S_{k^*}^*} g_{n+1-j} \right)^{-1} \quad \tilde{v} \stackrel{\text{def}}{=} \tilde{g}^u \cdot \left( \prod_{j \in S_{k^*}^*} \tilde{g}_{n+1-j} \right)^{-1}$$

$$d_i \stackrel{\text{def}}{=} \tilde{g}_i^u / \left( \prod_{j \in S_{k^*}^*} \tilde{g}_{n+1-j+i} \right) = \tilde{g}^{u \cdot \alpha^i} \cdot \left( \prod_{j \in S_{k^*}^*} \tilde{g}_{n+1-j} \right)^{-\alpha^i} = \tilde{v}^{\alpha^i}$$

Eventually,  $\mathcal{B}$  gives  $\mathcal{A}$  the public global parameters  $(g, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n}, d_n, \tilde{g}, \tilde{g}_1, \dots, \tilde{g}_n, \tilde{g}_{n+2}, \dots, \tilde{g}_{2n}, \mathcal{H})$ .

**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  $y \xleftarrow{\$} \mathbb{Z}_p$  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  $(S_1, S_2, \dots, S_m)$ ,  $\mathcal{B}$  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  $i = 1, \dots, m+1$ :  $y_i$  is obtained from the hash List. To generate  $\mathbf{Hdr} = (C_1, \tilde{C}_1, C_2, C_3)$ ,  $\mathcal{B}$  sets  $C_1 = g^r, \tilde{C}_1 = \tilde{g}^r$ , and computes

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

and  $K_i = e(g_n, \tilde{g}_1)^{r+y_i}$ , for  $i = 1, \dots, m+1$ .

3. For a decryption query  $(S_1, \dots, S_{m+1}, \mathbf{Hdr}, i, k)$  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  $j \in S_k \setminus S_{k^*}^*$ , 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  $K_{m+1} = e(g_{n+1}, \tilde{g})^{r+y_{m+1}}$ ;
- $\mathcal{B}$  extracts, from the hash List for  $\mathcal{H}(m+1, C_1)$ , the value  $y_{m+1}$ , and computes

$$L = \frac{K_{m+1}}{e(g_n, \tilde{g}_1)^{y_{m+1}}} = e(g_n, \tilde{g}_1)^r$$

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

$$K_k = L \times e(g_n, \tilde{g}_1)^{y_k} = e(g_{n+1}, \tilde{g})^{r+y_k}$$

**Challenge.** The challenge has to be generated on the target sets  $S_1^*, \dots, S_{m^*}^*$ , 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  $i = 1, \dots, n$ ;
- 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  $i = 1, \dots, k^* - 1, k^* + 1, \dots, m^* + 1$

Note that  $S_{m^*+1}^* = \{n\}$ , then  $\mathcal{B}$  generates  $\mathbf{Hdr}^* = (C_1^*, \tilde{C}_1^*, C_2^*, C_3^*)$  by setting  $C_1^* = h, \tilde{C}_1^* = \tilde{g}^{r^*}$  and  $C_3^* = \mathcal{H}(C_1^*, C_2^*)^{r^*}$ , where (as in the previous proof)

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

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

$$K_i^* = e(g_n, \tilde{g}_1)^{y_i^*} \cdot e(h_1, \tilde{g}_n) = e(g_{n+1}, \tilde{g})^{r^* + y_i^*}, \quad i \neq k^*.$$

It then sets

$$K_{k^*}^* = \frac{T \cdot e(h_1, \tilde{g}_n)}{e(g_n, \tilde{g}_1)^{z^*}}$$

It gives  $(\mathbf{Hdr}^*, K_1^*, \dots, K_{m^*+1}^*)$  as the challenge to  $\mathcal{A}$ .

Note that when  $T = e(G, \tilde{g}_{n+1})$ , with  $G = Y_{k^*}^* g^{z^*}$ ,

$$K_{k^*}^* = \frac{e(Y_{k^*}^* g^{z^*}, \tilde{g}_{n+1}) \cdot e(h_1, \tilde{g}_n)}{e(g_n, \tilde{g}_1)^{z^*}} = e(g_{n+1}, \tilde{g})^{y_{k^*}^*} \cdot e(g_{n+1}, \tilde{g})^{r^*} = e(g_{n+1}, \tilde{g})^{r^* + y_{k^*}^*}$$

**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 programming 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  $\mathbf{Succ}_{2.1}(\mathcal{A}) - \mathbf{Succ}_{2.2}(\mathcal{A}) \leq \mathbf{Adv}^{\text{dbdhe}}(\mathcal{B})$ . As a consequence,

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

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

## 5. Private Ciphertext-Policy Attribute-Based Encryption

In this section, we first describe the model for a private-key CP-ABE scheme, we next rely on our proposed technique and the techniques given in the Junod-Karlov ABE scheme [27] to construct two private CP-ABE schemes. We also give the security analysis of those schemes at the end of the section.

### 5.1. Definition of Private-Key CP-ABE

#### 5.1.1. Syntax

Formally, we define a *private* CP-ABE scheme which consists of three probabilistic algorithms as follows.

**Setup**( $1^\lambda, \vartheta, \mathcal{B}(u_i)_{1 \leq i \leq \vartheta}$ ) : it takes as input the security parameter  $\lambda$ , the total number of users in the system  $\vartheta$ , and the attribute repartition  $\mathcal{B}(u_i)_{1 \leq i \leq \vartheta}$  for each user  $u_i$  ( $\mathcal{B}(u_i)$  is the attribute set of user  $u_i$ ), generates the global parameters **param** of the system, an encryption key **EK**, and  $\vartheta$  decryption keys  $d_{u_i}$ . The encryption key **EK** is kept private from users. The set  $\mathcal{K}$  corresponds to the key space for session keys.

**Encrypt**( $\mathbb{A}, \mathbf{EK}, \mathbf{param}$ ) : it takes as input an access policy  $\mathbb{A}$  and the encryption key **EK**. It outputs the session keys  $K \in \mathcal{K}$  and the header **Hdr** which includes the access policy  $\mathbb{A}$ .

**Decrypt**(**Hdr**,  $d_{u_i}, \mathcal{B}(u_i), \mathbf{param}$ ) : it takes as input the header **Hdr**, a decryption key  $d_{u_i}$  and the attribute set  $\mathcal{B}(u_i)$  of user  $u_i$ , together with the parameters **param**. It outputs the session keys  $K$  if and only if  $\mathcal{B}(u_i)$  satisfies  $\mathbb{A}$ . Otherwise, it outputs  $\perp$ .

### 5.1.2. Security Model

In this paper, we consider the same security model as in [27] which is called *semantic security with full static collusions*, that in fact is the selective security in the generic group model. A private CP-ABE scheme is said to be secure in this model if given to an adversary (i) a challenge header, (ii) all the decryption keys of revoked users and (iii) an access to both encryption and decryption oracles, it is impossible for the adversary to infer any information about the session key. Formally, we now define the security model for a private CP-ABE scheme by the following probabilistic game between an attacker  $\mathcal{A}$  and a challenger  $\mathcal{C}$ .

**Initialiation.** Both  $\mathcal{A}$  and  $\mathcal{C}$  are given a system consisting of  $n$  attributes  $A_1, \dots, A_n$ .  $\mathcal{A}$  outputs target access policy  $\mathbb{A}^*$  as well as a repartition  $\mathcal{B}(u_i)_{1 \leq i \leq \vartheta}$  which he intends to attack.

**Setup**( $1^\lambda, \vartheta, \mathcal{B}(u_i)_{1 \leq i \leq \vartheta}$ ) The challenger runs the **Setup**( $1^\lambda, \vartheta, \mathcal{B}(u_i)_{1 \leq i \leq \vartheta}$ ) algorithm, he gives to  $\mathcal{A}$  the decryption keys  $d_{u_i}$  where  $\mathcal{B}(u_i)$  does not satisfy the target access policy  $\mathbb{A}^*$  and **param**. Decryption lists  $\Lambda_D$  is set to empty list.

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

1. Decryption query on the header **Hdr** with  $u_i$ . The challenger answers with **Decrypt**(**Hdr**,  $d_{u_i}$ ,  $\mathcal{B}(u_i)$ , **param**). The full header **Hdr** is appended to the decryption list  $\Lambda_D$ ;
2. Encryption query for the access policy  $\mathbb{A}$ . The challenger answers with **Encrypt**( $\mathbb{A}$ , **EK**, **param**). Remark that he/she can ask encryption query on target access policy  $\mathbb{A}^*$  since the encryption algorithm uses a fresh random coin for each time of the encryption.

**Challenge.** The challenger runs **Encrypt**( $\mathbb{A}^*$ , **EK**, **param**) and gets  $(K^*, \text{Hdr}^*)$ . Next, the challenger picks a random  $b \xleftarrow{\$} \{0, 1\}$ . If  $b = 0$ , the challenger sets  $K = K^*$ . Else, it picks a random  $K \xleftarrow{\$} \mathcal{K}$ . It outputs  $(K, \text{Hdr}^*)$  to  $\mathcal{A}$ . Note that if  $b = 0$ ,  $K$  is the real key, encapsulated in  $\text{Hdr}^*$ , and if  $b = 1$ ,  $K$  is random, independent of the header.

**Query phase 2.** The adversary  $\mathcal{A}$  continues to adaptively ask 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  $\text{Hdr}^* \notin \Lambda_D$ . We then denote the advantage of the adversary to win the game by

$$\mathbf{Adv}^{\text{ind}}(1^\lambda, \vartheta, \mathcal{B}(u_i)_{1 \leq i \leq \vartheta}, \mathcal{A}) = |2\Pr[b = b'] - 1|.$$

**Definition 11** (Basic Selective Security). *A private-key CP-ABE scheme is said to be basic selective security if the advantage of the adversary in the above security game is negligible where the adversary cannot ask the encryption query and the decryption query.*

**Definition 12** (Selective-CCA Security). *A private-key CP-ABE scheme is said to be selectively-CCA secure if the advantage of the adversary in the above security game is negligible where the adversary can ask any types of queries.*

### 5.2. Our First CP-ABE<sub>1</sub> Scheme

In this section, we introduce our first CP-ABE scheme that is secure in the generic group model, and achieves the basic selective security.

#### 5.2.1. Transition from Broadcast Encryption to CP-ABE

In the Junod-Karlov scheme [27], the authors implicitly give a transition from a Broadcast scheme to a CP-ABE scheme. To this aim, they first consider each user in a Broadcast encryption system as an identity in a CP-ABE system (so  $n$  now denotes the number of attributes in the ABE system), they then propose a new technique to fight against attribute collusion to transform many instances of a Broadcast encryption scheme to a CP-ABE scheme, such that one instance of the Broadcast encryption scheme corresponds to one clause in the CNF access policy of CP-ABE scheme. Note that the session key of the resulting CP-ABE scheme will be the product of partial session keys of Broadcast encryption schemes. However, when they implement their approach with the BGW scheme, it leads to a CP-ABE scheme with large ciphertext-size, linear in the number of instances of BGW scheme (or number of clauses in CNF policy).

#### 5.2.2. Intuition of our CP-ABE Scheme

Observing their approach, we manage to use our proposed technique to multiply many instances of BGW scheme in only one single value as in the case of MCBE scheme. This interestingly leads to a CP-ABE scheme with constant-size ciphertext. More precisely, in BGW, each element of the header has the form

$$\left(g^r, (v \cdot \prod_{j \in \beta_k} g_{n+1-j})^r\right).$$

In the Junod-Karlov scheme [27], the authors manage to transform many instances of the BGW scheme [9] to an attribute-based encryption scheme, such that one instance of the BGW scheme corresponds to one clause in the CNF access policy. The resulting attribute-based encryption scheme then contains  $m$  BGW instances where  $m$  is the maximal number of clauses in the CNF access policy. However, this leads to a ciphertext with  $m + 1$  parts. More precisely, for a CNF access policy  $\mathbb{A} = \beta_1 \wedge \dots \wedge \beta_m$ , each component  $\beta_k, k \in [m]$ , is related to a BGW header as

$$\left(g^{r \cdot t_k}, (v^r \prod_{j \in \beta_k} g_{n+1-j}^r)^{t_k}\right).$$

In our proposed technique, one can multiply many BGW instances in one single value in order to support the new property of multi-channel for broadcast encryption. For this purpose, we introduce new integers  $x_j$  and provide a unique header given by

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

Inspired by this, we manage to multiply the  $m$  instances of the BGW schemes to achieve a CP-ABE scheme with constant-size ciphertext. Our CP-ABE scheme therefore inherits the properties of the MCBE scheme, as the private property and the basic selective security.

### 5.2.3. Description

We now give the details of our first CP-ABE by describing each procedure.

**Setup**( $1^\lambda, \vartheta, \mathcal{B}(u_i)_{1 \leq i \leq \vartheta}$ ) : the algorithm takes as input the security parameter  $\lambda$ , the total number of users in the system  $\vartheta$ , and the attribute repartition  $\mathcal{B}(u_i)_{1 \leq i \leq \vartheta}$  for each user  $u_i$ , generates the global parameters **param** of the system, the encryption key **EK**, and  $\vartheta$  decryption keys  $d_{u_i}, 1 \leq i \leq \vartheta$  as follows:

Let  $(p, \mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e)$  be a bilinear map group system and let  $n$  be the maximal number of attributes in the system. The set of all possible attributes is  $\{A_1, \dots, A_n\}$ . All these elements are considered to be known to each participant.

The algorithm first picks random generators  $g \in \mathbb{G}$  and  $\tilde{g} \in \tilde{\mathbb{G}}$ . It then chooses a random scalar  $\alpha \in \mathbb{Z}_p$  and computes for all  $i \in [1, 2n] \setminus \{n+1\}$ , the values  $g_i = g^{\alpha^i}$  and  $\tilde{g}_i = \tilde{g}^{\alpha^i}$ . It also chooses at random  $r \in \mathbb{Z}_p$  and computes  $R = g^r$  and then, for all  $i \in [1, 2n] \setminus \{n+1\}$ ,  $h_i = g_i^r \in \mathbb{G}$ . Next, it picks random scalars  $\beta, \gamma \in \mathbb{Z}_p$  and sets  $B = g_n^\beta$ ,  $v = g^\gamma$  and  $V = v^r$ . It also picks additional random scalars  $x_1, x_2, \dots, x_n \in \mathbb{Z}_p$  and sets  $X_i = R^{x_i}$  for all  $i \in [1, n]$ . The public parameters are then

$$\text{param} = (g, \tilde{g}, B, R, V, g_n, \tilde{g}_1^r, h_1, \dots, h_n, h_{n+2}, \dots, h_{2n}, X_1, \dots, X_n)$$

The encryption key is **EK** =  $\{x_1, \dots, x_n\}$ .

To generate a decryption key  $d_u$ , let  $\mathcal{B}(u) = (A_{i_1}, \dots, A_{i_N})$  be the set of attributes of user  $u$  (among the set of all possible attributes). The algorithm first picks a random scalar  $s_u \in \mathbb{Z}_p$ , and computes  $\tilde{d}_{u_0} = \tilde{g}_1^{r(\beta + s_u)}$ , then  $\tilde{d}_{u_i} = \tilde{g}_i^{s_u}$  for all  $i \in [1, 2n] \setminus \{n+1\}$ , and finally  $\tilde{d}_j = \tilde{g}_j^{\gamma \cdot s_u}$  for all  $j \in \{i_1, \dots, i_N\}$ . The private decryption key for  $u$  is

$$d_u = (\tilde{d}_{u_0}, \tilde{d}_{u_1}, \dots, \tilde{d}_{u_n}, \tilde{d}_{u_{n+2}}, \dots, \tilde{d}_{u_{2n}}, \tilde{d}_{i_1}, \dots, \tilde{d}_{i_N}).$$

**Encrypt**( $\mathbb{A}, \text{EK}, \text{param}$ ) : assuming that the access policy is expressed in CNF  $\mathbb{A} = \beta_1 \wedge \dots \wedge \beta_m$ . The encryption phase works as follows. It first picks a random scalar  $t \in \mathbb{Z}_p$  and sets the session key as

$$K = e(B, \tilde{g}_1^r)^{m \cdot t + \sum_{k=1}^m \sum_{j \in \beta_k} x_j} = e(g_{n+1}, \tilde{g})^{r \cdot \beta (m \cdot t + \sum_{k=1}^m \sum_{j \in \beta_k} x_j)}.$$

It then computes the following values:

$$C_1 = R^t, C_2 = \prod_{k=1}^m (V \cdot \prod_{j \in \beta_k} h_{n+1-j})^{t + \sum_{j \in \beta_k} x_j}, C_3 = g_n^{m \cdot t + \sum_{k=1}^m \sum_{j \in \beta_k} x_j}.$$

The header is finally set to  $\mathbf{Hdr} = (\mathbb{A}, C_1, C_2, C_3)$ , and the pair  $(\mathbf{Hdr}, K)$  is the output.

**Decrypt**( $\mathbf{Hdr}, d_u, \mathcal{B}(u), \text{param}$ ) : this algorithm first parses  $\mathbf{Hdr} = (\mathbb{A}, C_1, C_2, C_3)$ .

Then, it computes a partial session key  $K_k$  for each clause  $\beta_k$  in  $\mathbb{A}$ ,  $k \in [1, m]$ . For that purpose, the user  $u$  chooses an attribute  $A_i \in (\beta_k \cap \mathcal{B}(u))$ , retrieves the corresponding private decryption key  $\tilde{d}_i$  and first computes

$$T_i = e(C_1 \cdot \prod_{j \in \beta_k} X_j, \tilde{d}_i \cdot \prod_{\substack{j \in \beta_k \\ j \neq i}} \tilde{d}_{u_{n+1-j+i}}).$$

The partial session key  $K_k$  is then computed as

$$K_k = \frac{e(C_2, \tilde{d}_{u_i})}{T_i \cdot \prod_{\substack{\ell=1 \\ \ell \neq k}}^m e(C_1 \cdot \prod_{j \in \beta_\ell} X_j, \tilde{d}_i \cdot \prod_{j \in \beta_\ell} \tilde{d}_{u_{n+1-j+i}})}.$$

We then remark that

$$\prod_{k=1}^m K_k = e(g_{n+1}, \tilde{g})^{(m \cdot t + \sum_{k=1}^m \sum_{j \in \beta_k} x_j) r \cdot s_u}$$

It follows that the session key can be computed as  $K = \frac{e(\tilde{d}_{u_0}, C_3)}{\prod_{k=1}^m K_k}$ .

**For the correctness:** we first focus on the partial session key  $K_k$ . We use the relations  $\tilde{d}_i = \tilde{g}^{\gamma s_u \cdot \alpha^i}$ ,  $\tilde{d}_{u_i} = \tilde{g}_i^{s_u}$ ,  $\tilde{d}_{u_{n+1-j+i}} = \tilde{g}_{n+1-j+i}^{s_u}$ , and  $g_{n+1-j+i} = g_{n+1-j}^{\alpha^i}$ ,  $\tilde{g}_{n+1-j+i} = \tilde{g}_{n+1-j}^{\alpha^i}$ ,  $g_{n+1-i}^{\alpha^i} = g_{n+1-i}$ ,  $\tilde{g}_{n+1-i}^{\alpha^i} = \tilde{g}_{n+1-i}$ , and  $V = v^r$ ,  $h_i = g_i^r$ . It follows that  $K_k =$

$$\begin{aligned} &= \frac{e(\prod_{\ell=1}^{\ell=m} (v^r \cdot \prod_{j \in \beta_\ell} g_{n+1-j}^r)^{t + \sum_{j \in \beta_\ell} x_j}, \tilde{g}^{s_u \cdot \alpha^i})}{e(g^{r(t + \sum_{j \in \beta_k} x_j)}, \tilde{g}^{\gamma s_u \cdot \alpha^i} \cdot (\prod_{\substack{j \in \beta_k \\ j \neq i}} \tilde{g}_{n+1-j}^{s_u})^{\alpha^i}) \cdot \prod_{\substack{\ell=1 \\ \ell \neq k}}^{\ell=m} e(g^{r(t + \sum_{j \in \beta_\ell} x_j)}, \tilde{g}^{\gamma s_u \cdot \alpha^i} \cdot (\prod_{j \in \beta_\ell} \tilde{g}_{n+1-j}^{s_u})^{\alpha^i})} \\ &= \frac{e((g^\gamma \cdot \prod_{j \in \beta_k} g_{n+1-j})^{\alpha^i}, \tilde{g}^{t + \sum_{j \in \beta_k} x_j})^{r \cdot s_u}}{e(g^{t + \sum_{j \in \beta_k} x_j}, (\tilde{g}^\gamma \cdot \prod_{\substack{j \in \beta_k \\ j \neq i}} \tilde{g}_{n+1-j})^{\alpha^i})^{r \cdot s_u}} \cdot \prod_{\substack{\ell=1 \\ \ell \neq k}}^{\ell=m} \frac{e((g^\gamma \cdot \prod_{j \in \beta_\ell} g_{n+1-j})^{\alpha^i}, \tilde{g})^{r \cdot s_u \cdot (t + \sum_{j \in \beta_\ell} x_j)}}{e(g, (\tilde{g}^\gamma \cdot \prod_{j \in \beta_\ell} \tilde{g}_{n+1-j})^{\alpha^i})^{r \cdot s_u \cdot (t + \sum_{j \in \beta_\ell} x_j)}} \\ &= e(g_{n+1-i}^{\alpha^i}, \tilde{g}^{t + \sum_{j \in \beta_k} x_j})^{r \cdot s_u} = e(g_{n+1}, \tilde{g}^{t + \sum_{j \in \beta_k} x_j})^{r \cdot s_u} = e(g_{n+1}, \tilde{g})^{(t + \sum_{j \in \beta_k} x_j) r \cdot s_u} \end{aligned}$$

Now focusing on the session key  $K$ , we have

$$\frac{e(\tilde{d}_{u_0}, C_3)}{\prod_{k=1}^m K_k} = \frac{e(\tilde{g}_1^{r(\beta+s_u)}, g_n^{m.t+\sum_{k=1}^m \sum_{j \in \beta_k} x_j})}{e(g_{n+1}, \tilde{g})^{(m.t+\sum_{k=1}^m \sum_{j \in \beta_k} x_j)r.s_u}} = e(g_{n+1}, \tilde{g})^{r.\beta(m.t+\sum_{k=1}^m \sum_{j \in \beta_k} x_j)},$$

which exactly corresponds to the key  $K$  generated at the encryption step.

**Remark 2.** In the above construction, the attributes cannot be reused in the access policy since each  $\beta_k$  is a disjoint subset (following the proposed technique in *MCBE* scheme). To deal with this drawback, as in [43], we allow each attribute to have many copies of itself. If we assume that  $k_{\max}$  is the maximal number of times in which each attribute can appear in the access formula, then each attribute will have  $k_{\max}$  copies of itself. For example, the attribute *professor* can be represented by  $k_{\max}$  different attributes *professor*<sub>1</sub>, ..., *professor* <sub>$k_{\max}$</sub>  corresponding to  $k_{\max}$  different secret keys  $d_{i_1}, \dots, d_{i_{k_{\max}}}$ . A user possessing the attribute *professor* will receive  $k_{\max}$  corresponding secret keys  $d_{i_1}, \dots, d_{i_{k_{\max}}}$ . Therefore, the construction above can support CNF access policy with the cost that the key size is a factor of  $k_{\max}$  larger.

**Remark 3.** Following the work in [27], the construction above can easily be extended to support revocation. For that purpose, we consider the identity of each user as an additional attribute (without the need to have copies of this special attribute). Then, to do the revocation, the encryption procedure needs to add one more set  $\beta_{m+1}$  containing the identities of privileged (non revoked) users. The users outside the set  $\beta_{m+1}$  (revoked users) cannot decrypt because it lacks the partial session key corresponding to the set  $\beta_{m+1}$ . It follows that the key size in our scheme will be similar to the one in Junod-Karlov scheme [27], that is linear in the maximal number of users in the system.

This way, we obtain an *ABBE* scheme with constant-size ciphertext.

#### 5.2.4. Security

In this section, we first give a theorem to prove that our first CP-ABE scheme achieves basic selective security under a  $(P, Q, R, f)$ -GDDHE assumption. We then show that this assumption holds in the generic group model.

More precisely, following the security model we define in section 5.1.2 the adversary first outputs the target access policy  $\mathbb{A}^*$  as well as a repartition  $\mathcal{B}(u_i)_{1 \leq i \leq \vartheta}$  which he intends to attack. The challenger then runs the setup algorithm and returns the **param**, decryption keys of all user  $u_i$  where  $\mathcal{B}(u_i)$  does not satisfy the target access policy  $\mathbb{A}^*$  to the adversary, he also computes and returns the challenge header to the adversary. The adversary finally needs to make his guess on bit  $b$ . According to the framework of GDDHE assumption, we can describe this fact as a  $(P, Q, R, f)$ -GDDHE assumption as follows. Let  $P, Q, R$  be the list of polynomials consisting of all elements corresponding to the public global parameters,

the private decryption keys of corrupted users, and the challenge header.

$$\begin{aligned}
P &= \left\{ 1, r, \alpha^n \beta, r\gamma, \alpha^n, r\alpha, \dots, r\alpha^n, r\alpha^{n+2}, \dots, r\alpha^{2n}, x_1 r, \dots, x_n r, rt, \alpha^n (mt + \sum_{k=1}^m \sum_{j \in \beta_k} x_j), \right. \\
&\quad \left. \sum_{k=1}^m (r\gamma + \sum_{j \in \beta_k} \alpha^{n+1-j} r) (t + \sum_{j \in \beta_k} x_j) \right\} \\
Q &= \{1, \alpha r, r(\beta + s_u) \alpha, \alpha s_u, \dots, \alpha^n s_u, \alpha^{n+2} s_u, \dots, \alpha^{2n} s_u, \alpha^{i_1} \gamma s_u, \dots, \alpha^{i_N} \gamma s_u\} \\
R &= \{1\}, \quad f = \alpha^{n+1} r \beta (mt + \sum_{k=1}^m \sum_{j \in \beta_k} x_j)
\end{aligned}$$

For all corrupted user  $u$ ,  $1 \leq N = |\mathcal{B}(u)| \leq n$ .

**Theorem 13.** *If there exists an adversary  $\mathcal{A}$  that solves the basic selective security of our first CP-ABE scheme with advantage  $\varepsilon$ , then we can construct a simulator to solve the  $(P, Q, R, f)$ -GDDHE assumption above with the same advantage  $\varepsilon$  in polynomial time.*

*Proof.* Assume that  $\mathcal{B}$  is a simulator that solves the  $(P, Q, R, f)$ -GDDHE assumption above. At the beginning,  $\mathcal{B}$  is given an instance of the  $(P, Q, R, f)$ -GDDHE assumption, i.e., all elements corresponding to the public global parameters, the private decryption keys of corrupted users, and the challenge header (denoted  $g^{P(\dots)}, \tilde{g}^{Q(\dots)}, g_T^{R(\dots)}$ ), as well as an element  $K$  such that  $K = e(g, \tilde{g})^f$  if bit  $b = 0$ , and  $K$  is a random element in  $\mathbb{G}_T$  if  $b = 1$ .  $\mathcal{B}$  will use this instance to simulate  $\mathcal{A}$  and use the output of  $\mathcal{A}$  to guess bit  $b$ . To do that, in the setup phase  $\mathcal{B}$  gives  $\mathcal{A}$  the public global parameters, the private decryption keys of corrupted users. Finally in the challenge phase,  $\mathcal{B}$  gives  $\mathcal{A}$  the challenge header as well as  $K$ . We note that all of these information are in  $g^{P(\dots)}, \tilde{g}^{Q(\dots)}$ . When  $\mathcal{A}$  outputs its guess for  $b$ ,  $\mathcal{B}$  uses this guess to break the security of the  $(P, Q, R, f)$ -GDDHE assumption. Since the simulation is perfect and  $\mathcal{A}$  has advantage  $\varepsilon$ ,  $\mathcal{B}$  also has the same advantage  $\varepsilon$  in solving the  $(P, Q, R, f)$ -GDDHE assumption.  $\square$

We are now going to prove in Appendix A that  $(P, Q, R)$  and  $f$  are independent, so that the  $(P, Q, R, f)$ -GDDHE assumption holds in our case.

**Lemma 14.** *In the  $(P, Q, R, f)$  – GDDHE assumption above,  $(P, Q, R)$  and  $f$  are linearly independent.*

### 5.3. Second Construction – CP-ABE<sub>2</sub>

We now give the details of our second CP-ABE scheme, which aims at improving the first CP-ABE scheme regarding the security. More precisely, it achieves selective CCA security under again a similar GDDHE assumption but needs to use the random oracle.

### 5.3.1. Description

In this construction, instead of generating the terms  $X_i$ , we use a random oracle to generate them at the time of encryption. In addition, we add a dummy clause containing only one attribute  $A_n$  to any access formula, and allow all users in the system to possess this attribute. This way, we are able to reach the selective CCA security.

**Setup**( $1^\lambda, \vartheta, \mathcal{B}(u_i)_{1 \leq i \leq \vartheta}$ ) : similar to the one in the first construction, except that the algorithm here uses an additional random oracle  $\mathcal{H}$  on to  $\mathbb{G}$  and  $\tilde{h} = \tilde{g}^r$ . The public parameters<sup>1</sup> are then

$$\text{param} = (g, \tilde{g}, h, \tilde{h}, V, g_n, h_1, \dots, h_n, h_{n+2}, \dots, h_{2n}, \mathcal{H})$$

The encryption key is  $\text{EK} = (r, \beta, \gamma, \alpha) \cup \text{param}$ .

To generate the decryption key for user  $u$ , similar to the one in the first construction, let  $\mathcal{B}(u) = (A_{i_1}, \dots, A_{i_N}, A_n)$  be the set of attributes of user  $u$ . The private decryption key for  $u$  is

$$d_u = (\tilde{d}_{u_0}, \tilde{d}_{u_1}, \dots, \tilde{d}_{u_n}, \tilde{d}_{u_{n+2}}, \dots, \tilde{d}_{u_{2n}}, \tilde{d}_{i_1}, \dots, \tilde{d}_{i_N}, \tilde{d}_n).$$

**Encrypt**( $\mathbb{A}, \text{EK}, \text{param}$ ): assume that the access policy is expressed in CNF  $\mathbb{A} = \beta_1 \wedge \beta_2 \wedge \dots \wedge \beta_m$ , where  $\beta_m$  is a dummy clause that only contains the attribute  $A_n$ . The encryption phase works as follows: it first picks a random scalar  $t \xleftarrow{\$} \mathbb{Z}_p$ , and then computes  $Y_i = \mathcal{H}(i, h^t) = h^{y_i}$  for  $i = 1, \dots, m$  with unknown scalars  $y_i$ . The session key is then computed as:

$$K = e(g_{n+1}, \tilde{g})^{r \cdot \beta \cdot m \cdot t} \prod_{k=1}^m e(Y_k, \tilde{g}_{n+1}^\beta) = e(g_{n+1}, \tilde{g})^{r \cdot \beta (m \cdot t + \sum_{k=1}^m y_k)}.$$

Next, one computes:

$$C_1 = h^t, \quad \tilde{C}_1 = \tilde{h}^t, \quad C_2 = \prod_{k=1}^{k=m} Y_k^\gamma V^t \prod_{j \in \beta_k} Y_k^{\alpha^{n+1-j}} h_{n+1-j}^t = \prod_{k=1}^{k=m} (V \cdot \prod_{j \in \beta_k} h_{n+1-j})^{t+y_k}$$

$$C_3 = g_n^{m \cdot t} \cdot \prod_{k=1}^m ((Y_k)^{r^{-1}})^{\alpha^n} = g_n^{m \cdot t + \sum_{k=1}^m y_k}, \quad C_4 = \mathcal{H}(C_1, C_2, C_3)^t$$

The broadcaster can easily compute  $K$  and  $\text{Hdr}$  because it knows the values  $r, \beta, \alpha, \gamma, g, \tilde{g}$  from  $\text{EK}$ . The header is set to  $\text{Hdr} = (\mathbb{A}, C_1, \tilde{C}_1, C_2, C_3, C_4)$ , and the pair  $(\text{Hdr}, K)$  is the output.

---

<sup>1</sup>We make the choice of putting all these values into  $\text{param}$ , so that the encryptor doesn't need to re-compute these values when encrypting. Another possibility is to set  $\text{param} = \{g, \tilde{g}, h, \tilde{h}, \mathcal{H}\}$  and re-compute all others values when encrypting.

**Decrypt**(Hdr,  $d_u$ ,  $\mathcal{B}(u)$ , param): the user  $u$  first parses the header Hdr as above:  $(\mathbb{A}, C_1, \tilde{C}_1, C_2, C_3, C_4)$ . It then checks whether the equations

$$e(C_1, \tilde{h}) = e(h, \tilde{C}_1) \text{ and } e(\mathcal{H}(C_1, C_2, C_3), \tilde{C}_1) = e(\tilde{h}, C_4)$$

hold. It then computes  $Y_i = \mathcal{H}(i, C_1)$  for  $i = 1, \dots, m$ . For each clause  $\beta_k$  in  $\mathbb{A}$ , the user  $u$  chooses an attribute  $A_i \in (\beta_k \cap \mathcal{B}(u))$  and computes, as in the previous scheme, for each  $k \in [1, m]$ :

$$\begin{aligned} K_k &= \frac{e(C_2, \tilde{d}_{u_i})}{e(C_1 \cdot Y_k, \tilde{d}_i \cdot \prod_{\substack{j \in \beta_k \\ j \neq i}} \tilde{d}_{u_{n+1-j+i}}) \cdot \prod_{\substack{\ell=1 \\ \ell \neq k}}^m e(C_1 \cdot Y_\ell, \tilde{d}_i \cdot \prod_{j \in \beta_\ell} \tilde{d}_{u_{n+1-j+i}})} \\ &= e(g_{n+1}, \tilde{g})^{(t+y_k)r \cdot s_u}. \end{aligned}$$

We remark that  $\prod_{k=1}^m K_k = e(g_{n+1}, \tilde{g})^{(m \cdot t + \sum_{k=1}^m y_k)r \cdot s_u}$ . The session key is then computed as:

$$K = \frac{e(C_3, \tilde{d}_{u_0})}{\prod_{k=1}^m K_k} = \frac{e(g_n^{m \cdot t + \sum_{k=1}^m y_k}, \tilde{g}_1^{r(\beta + s_u)})}{e(g_{n+1}, \tilde{g})^{(m \cdot t + \sum_{k=1}^m y_k)r \cdot s_u}} = e(g_{n+1}, \tilde{g})^{r \cdot \beta(m \cdot t + \sum_{k=1}^m y_k)}.$$

### 5.3.2. Security

In this section, we first give a theorem to prove that our second scheme is selective CCA secure under a  $(P, Q, R, f)$ -GDDHE assumption. We then show that this assumption holds in the generic group model.

The  $(P, Q, R, f)$ -GDDHE assumption that we need is, in fact, similar to the one given in Section 2.3, except that the terms  $rx_1, \dots, rx_n$  are now replaced by the terms  $ry_1, \dots, ry_m, z, zt$ . More precisely, let  $P, Q, R$  be the list of polynomials consisting of all elements corresponding to the public global parameters, the private decryption keys of revoked users, and the challenge header.

$$\begin{aligned} P &= \left\{ 1, r, r\gamma, \alpha^n, r\alpha, \dots, r\alpha^n, r\alpha^{n+2}, \dots, r\alpha^{2n}, ry_1, \dots, \right. \\ &\quad \left. ry_m, z, zt, rt, \alpha^n(mt + \sum_{k=1}^m y_k), \sum_{k=1}^m (r\gamma + \sum_{j \in \beta_k} \alpha^{n+1-j}r)(t + y_k) \right\} \\ Q &= \left\{ 1, rt, r(\beta + s_u)\alpha, \alpha s_u, \dots, \alpha^n s_u, \alpha^{n+2} s_u, \dots, \alpha^{2n} s_u, \alpha^{i_1} \gamma s_u, \dots, \alpha^{i_N} \gamma s_u, \alpha^n \gamma s_u \right\} \\ R &= \{1\}, \text{ and } f = \alpha^{n+1} r \beta (mt + \sum_{k=1}^m y_k). \end{aligned}$$

For each user  $u$  belonging to the set of corrupted users, we have  $1 \leq N = |\mathcal{B}(u)| < n$ . This assumption can now be re-written as follows. Given

$$g, \tilde{g}, \tilde{g}_1^{r(\beta+s_u)}, \tilde{g}_1^{s_u}, \dots, \tilde{g}_n^{s_u}, \tilde{g}_{n+2}^{s_u}, \dots, \tilde{g}_{2n}^{s_u}, \tilde{g}_{i_1}^{\gamma s_u}, \dots, \tilde{g}_{i_N}^{\gamma s_u}, \tilde{g}_n^{\gamma s_u}, h, V, g_n, h_1, \dots, h_n, h_{n+2}, \\ \dots, h_{2n}, h^{y_1}, \dots, h^{y_m}, g^z, g^{z^t}, h^t, \tilde{h}^t, \prod_{k=1}^{k=m} (V \cdot \prod_{j \in \beta_k} h_{n+1-j})^{t+y_k}, g_n^{m \cdot t + \sum_{k=1}^m y_k}$$

for all corrupted user  $u$ , distinguish between the value  $e(g_{n+1}, \tilde{g})^{r \cdot \beta(m \cdot t + \sum_{k=1}^m y_k)}$  and a random  $T \in \mathbb{G}_T$ .

**Theorem 15.** *Our second scheme is selectively-CCA secure under CDH assumption and the  $(P, Q, R, f)$ -GDDHE assumption above.*

*Proof.* Let  $\text{Hdr} = (\mathbb{A}, C_1, \tilde{C}_1, C_2, C_3, C_4)$  be the challenge header. Similar to the proof of  $\text{MCBE}_2$  scheme, we will prove the security of  $\text{CP-ABE}_2$  scheme in two steps. First, we prove that the adversary cannot produce any decryption query of the form  $\text{Hdr}' = (\mathbb{A}, C_1, \tilde{C}'_1, C'_2, C'_3, C'_4)$  under the CDH assumption. In the second step, we prove that our second scheme is selectively-CCA secure under  $(P, Q, R, f)$ -GDDHE assumption with the requirement that the adversary doesn't ask any query  $\text{Hdr}' = (\mathbb{A}, C_1, \tilde{C}'_1, C'_2, C'_3, C'_4)$ .

**First step.** This step is similar to the first step in the proof of  $\text{MCBE}_2$  scheme, we thus refer the reader to the one in the proof of  $\text{MCBE}_2$  scheme.

**Second step.** First, the simulator is given the instance of aforementioned  $(P, Q, R, f)$ -GDDHE assumption. Let  $\mathcal{A}$  be an adversary against the security of our second scheme. The simulator will use the guess of  $\mathcal{A}$  to break the instance of  $(P, Q, R, f)$ -GDDHE assumption. For that purpose, the simulator first from the instance of  $(P, Q, R, f)$ -GDDHE assumption gives  $\mathcal{A}$  the public parameters, and the decryption keys of all corrupted users. The simulator also needs to answer the following types of queries from  $\mathcal{A}$ .

1. *Hash query:* There are two types of hash queries,  $(j, h^*) \in \mathbb{Z}_p \times \mathbb{G}$  or  $(h_1^*, h_2^*, h_3^*) \in \mathbb{G}^3$ . For any query  $q$ , if it has been asked before, the same answer is sent back. Otherwise, for the  $(j, h^*)$  queries the simulator randomly chooses  $y \in \mathbb{Z}_p$  and sets  $\mathcal{H}(q) = h^y$ , and appends the tuple  $(q, h^y, y)$  to the hash list. If the value  $y$  is unknown, it is replaced by  $\perp$ . For the  $(h_1^*, h_2^*, h_3^*)$  query, the simulator randomly chooses  $z^* \in \mathbb{Z}_p$  and set  $\mathcal{H}(q) = g^{z^*}$ , and appends the tuple  $(q, g^{z^*}, z^*)$  to the hash list. If the value  $z^*$  is unknown, it is replaced by  $\perp$ .
2. *Encryption query:*  $\mathcal{A}$  sends an access policy  $\mathbb{A} = \beta'_1 \wedge \beta'_2 \wedge \dots \wedge \beta'_\ell$  to simulator where  $\beta'_\ell = A_n$ . The simulator first randomly chooses  $t', z', y'_1, \dots, y'_\ell \in \mathbb{Z}_p$  and appends to the hash list the tuple  $(q'_{z'}, g^{z'}, z')$  and for all  $i = 1, \dots, \ell$ , the tuples  $(q'_i, h^{y'_i}, y'_i)$ . It

takes the private decryption key of a user  $u$  and then computes:

$$K = \left( \frac{e(\tilde{g}_1^{r(\beta+s_u)}, g_n)}{e(\tilde{g}_n^{s_u}, g_1^r)} \right)^{t'\ell + \sum_{k=1}^{\ell} y'_k} = e(g_{n+1}, \tilde{g})^{r \cdot \beta(t'\ell + \sum_{k=1}^{\ell} y'_k)}$$

$$C_1 = h^{t'}, \tilde{C}_1 = \tilde{h}^{t'}, C_2 = \prod_{k=1}^{k=\ell} (V \cdot \prod_{j \in \beta_k} h_{n+1-j})^{t'+y'_k}, C_3 = g_n^{t'\ell + \sum_{k=1}^{\ell} y'_k}, C_4 = g^{z't'}.$$

3. *Decryption query*: we assume that  $\mathcal{A}$  sends the following ciphertext to the simulator (note that  $t' \neq t$  since one cannot reuse the  $C_1$  in the challenge header):

$$C_1 = h^{t'}, \tilde{C}_1 = \tilde{h}^{t'}, C_2 = \prod_{k=1}^{k=m'} (V \cdot \prod_{j \in \beta_k} h_{n+1-j})^{t'+y'_k},$$

$$C_3 = g_n^{m' \cdot t' + \sum_{k=1}^{m'} y'_k}, C_4 = \mathcal{H}(C_1, C_2, C_3)^{t'}$$

The simulator first checks whether the equations  $e(C_1, \tilde{h}) = e(h, \tilde{C}_1)$  and  $e(\mathcal{H}(C_1, C_2, C_3), \tilde{C}_1) = e(\tilde{h}, C_4)$  hold, takes the private decryption key of a corrupted user  $u$  and then uses the secret key  $\tilde{d}_n$  corresponding to attribute  $A_n$  in the clause  $\beta_{m'}$  to compute the value  $e(g_{n+1}, \tilde{g})^{(t'+y'_{m'})r \cdot s_u}$ . It extracts the value  $y'_{m'}$  from the hash list (since  $t' \neq t$ ) and compute  $e(\tilde{g}_n^{s_u}, g_1^r)^{y'_{m'}}$ . This permits to obtain the value

$$\frac{e(g_{n+1}, \tilde{g})^{(t'+y'_{m'})r \cdot s_u}}{e(\tilde{g}_n^{s_u}, g_1^r)^{y'_{m'}}} = e(g_{n+1}, \tilde{g})^{t' \cdot r \cdot s_u}.$$

Next, it extracts all the values from  $y'_1$  to  $y'_{m'-1}$  from the hash list (since  $t' \neq t$ ) and computes the partial session keys related to each clause  $\beta_i, i = 1, \dots, m' - 1$

$$K_i = e(g_{n+1}, \tilde{g})^{t' \cdot r \cdot s_u} \cdot e(\tilde{g}_n^{s_u}, g_1^r)^{y'_i} = e(g_{n+1}, \tilde{g})^{(t'+y'_i)r \cdot s_u}.$$

The simulator can finally recover the following session key and forwards the result to  $\mathcal{A}$ .

$$K = e(g_{n+1}, \tilde{g})^{r \cdot \beta(t'm' + \sum_{k=1}^{m'} y'_k)}.$$

Next, during the challenge phase, the simulator first appends to the hash list the values  $\mathcal{H}(i, h^t) = (q_i, h^{y_i}, \perp)$ , for all  $i = 1, \dots, m$  and the values  $\mathcal{H}(C_1, C_2, C_3) = (q_z, g^z, \perp)$ . It then sends the following challenge ciphertext to  $\mathcal{A}$ :

$$C_1 = h^t, \tilde{C}_1 = \tilde{h}^t, C_2 = \prod_{k=1}^{k=m} (V \cdot \prod_{j \in \beta_k} h_{n+1-j})^{t+y_k}, C_3 = g_n^{m \cdot t + \sum_{k=1}^m y_k}, C_4 = g^{zt}.$$

If  $\mathcal{A}$  make new requests to the different oracles, the simulator can use again the above strategy. Finally, when  $\mathcal{A}$  outputs its guess for  $b$ , the simulator uses this guess to break the security of the  $(P, Q, R, f)$ -GDDHE assumption.  $\square$

The following lemma finally shows that in the aforementioned  $(P, Q, R, f)$ -GDDHE assumption,  $(P, Q, R)$  and  $f$  are linearly independent. The proof of this lemma is similar to the one given for Lemma 14 and, therefore, we do not repeat it again.

**Lemma 16.** *In the  $(P, Q, R, f)$ -GDDHE assumption above,  $(P, Q, R)$  and  $f$  are linearly independent.*

## 6. Conclusion

In this paper, we introduced a new technique which can be used to optimize the ciphertext-size for both the multi-channel broadcast encryption and the ciphertext-policy attribute-based encryption. We leave some following challenging open problems for our proposed technique:

- This technique applies well to the context of multi-channel broadcast encryption and ciphertext-policy attribute-based encryption, but we do not know whether it can be applied to more general contexts such as functional encryption;
- This technique suffers the drawback of private-key only, and it also inherits the weakness of the BGW scheme, with an inefficient decryption procedure;
- This technique is partially based on the technique from the BGW scheme. It would be interesting to study whether we can rely on other techniques from other Broadcast encryption schemes with constant-size ciphertext, such as [16]. Note that in [11], the authors already made use of the technique from [16] to propose a CP-ABE scheme with constant-size secret key.

## References

- [1] Masayuki Abe, Jens Groth, Miyako Ohkubo, and Takeya Tango. Converting cryptographic schemes from symmetric to asymmetric bilinear groups. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 241–260, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.
- [2] Shashank Agrawal and Melissa Chase. A study of pair encodings: Predicate encryption in prime order groups. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 259–288, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

- [3] Shashank Agrawal and Melissa Chase. Simplifying design and analysis of complex predicate encryption schemes. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 627–656, Paris, France, May 8–12, 2017. Springer, Heidelberg, Germany.
- [4] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 557–577, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.
- [5] Nuttapong Attrapadung, Goichiro Hanaoka, and Shota Yamada. Conversions among several classes of predicate encryption and applications to ABE with various compactness tradeoffs. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 575–601, Auckland, New Zealand, November 30 – December 3, 2015. Springer, Heidelberg, Germany.
- [6] Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 90–108, Taormina, Italy, March 6–9, 2011. Springer, Heidelberg, Germany.
- [7] Mihir Bellare and Phillip 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, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.
- [8] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.
- [9] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany.
- [10] Dan Boneh, Brent Waters, and Mark Zhandry. Low overhead broadcast encryption from multilinear maps. In Juan A. Garay and Rosario Gennaro, editors, *Advances*

in *Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 206–223, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

- [11] Sébastien Canard, Duong Hieu Phan, and Viet Cuong Trinh. A new technique for compacting secret key in attribute-based broadcast encryption. In Sara Foresti and Giuseppe Persiano, editors, *CANS 16: 15th International Conference on Cryptology and Network Security*, volume 10052 of *Lecture Notes in Computer Science*, pages 594–603, Milan, Italy, November 14–16, 2016. Springer, Heidelberg, Germany.
- [12] Cheng Chen, Jie Chen, Hoon Wei Lim, Zhenfeng Zhang, Dengguo Feng, San Ling, and Huaxiong Wang. Fully secure attribute-based systems with short ciphertexts/signatures and threshold access structures. In Ed Dawson, editor, *Topics in Cryptology – CT-RSA 2013*, volume 7779 of *Lecture Notes in Computer Science*, pages 50–67, San Francisco, CA, USA, February 25 – March 1, 2013. Springer, Heidelberg, Germany.
- [13] Cheng Chen, Zhenfeng Zhang, and Dengguo Feng. Efficient ciphertext policy attribute-based encryption with constant-size ciphertext and constant computation-cost. In Xavier Boyen and Xiaofeng Chen, editors, *ProvSec 2011: 5th International Conference on Provable Security*, volume 6980 of *Lecture Notes in Computer Science*, pages 84–101, Xi’an, China, October 16–18, 2011. Springer, Heidelberg, Germany.
- [14] Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system ABE in prime-order groups via predicate encodings. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 595–624, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.
- [15] Jung Hee Cheon, KyooHyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 3–12, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.
- [16] Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 200–215, Kuching, Malaysia, December 2–6, 2007. Springer, Heidelberg, Germany.
- [17] Yevgeniy Dodis and Nelly Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In Yvo 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, Miami, FL, USA, January 6–8, 2003. Springer, Heidelberg, Germany.
- [18] Keita Emura, Atsuko Miyaji, Akito Nomura, Kazumasa Omote, and Masakazu Soshi. A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In Feng Bao, Hui Li, and Guilin Wang, editors, *ISPEC 2009: 5th International Conference on Information Security Practice and Experience*, volume 5451 of *Lecture Notes in Computer Science*, pages 13–23, Xi'an, China, April 13–15 2009. Springer, Heidelberg, Germany.
  - [19] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Heidelberg, Germany.
  - [20] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 479–499, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.
  - [21] Aijun Ge, Rui Zhang, Cheng Chen, Chuanguai Ma, and Zhenfeng Zhang. Threshold ciphertext policy attribute-based encryption with constant size ciphertexts. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *ACISP 12: 17th Australasian Conference on Information Security and Privacy*, volume 7372 of *Lecture Notes in Computer Science*, pages 336–349, Wollongong, NSW, Australia, July 9–11, 2012. Springer, Heidelberg, Germany.
  - [22] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In Antoine Joux, editor, *Advances in Cryptology – EURO-CRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 171–188, Cologne, Germany, April 26–30, 2009. Springer, Heidelberg, Germany.
  - [23] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06: 13th Conference on Computer and Communications Security*, pages 89–98, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press. Available as Cryptology ePrint Archive Report 2006/309.
  - [24] Mike Hamburg. Spatial encryption. Cryptology ePrint Archive, Report 2011/389, 2011. <http://eprint.iacr.org/2011/389>.

- [25] Javier Herranz, Fabien Laguillaumie, and Carla Ràfols. Constant size ciphertexts in threshold attribute-based encryption. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 19–34, Paris, France, May 26–28, 2010. Springer, Heidelberg, Germany.
- [26] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. Cryptology ePrint Archive, Report 2015/301, 2015. <http://eprint.iacr.org/2015/301>.
- [27] Pascal Junod and Alexandre Karlov. An efficient public-key attribute-based broadcast encryption scheme allowing arbitrary access policies. In Ehab Al-Shaer, Hongxia Jin, and Marc Joye, editors, *Proceedings of the 10th ACM Workshop on Digital Rights Management*, pages 13–24, Chicago, Illinois, USA, October 4 2010. ACM.
- [28] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.
- [29] Allison B. Lewko, Amit Sahai, and Brent Waters. Revocation systems with very small private keys. In *2010 IEEE Symposium on Security and Privacy*, pages 273–285, Berkeley/Oakland, CA, USA, May 16–19, 2010. IEEE Computer Society Press.
- [30] Allison B. Lewko and Brent Waters. Unbounded HIBE and attribute-based encryption. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 547–567, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.
- [31] David Lubicz and Thomas Sirvent. Attribute-based broadcast encryption scheme made efficient. In Serge Vaudenay, editor, *AFRICACRYPT 08: 1st International Conference on Cryptology in Africa*, volume 5023 of *Lecture Notes in Computer Science*, pages 325–342, Casablanca, Morocco, June 11–14, 2008. Springer, Heidelberg, Germany.
- [32] Qutaibah M. Malluhi, Abdullatif Shikfa, and Viet Cuong Trinh. A ciphertext-policy attribute-based encryption scheme with optimized ciphertext size and fast decryption. In Ramesh Karri, Ozgur Sinanoglu, Ahmad-Reza Sadeghi, and Xun Yi, editors, *AsiaCCS 2017: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 230–240, Abu Dhabi, United Arab Emirates, April 2–6 2017. ACM.
- [33] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume

9815 of *Lecture Notes in Computer Science*, pages 629–658, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

- [34] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.
- [35] Tatsuoaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 349–366, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.
- [36] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 07: 14th Conference on Computer and Communications Security*, pages 195–203, Alexandria, Virginia, USA, October 28–31, 2007. ACM Press.
- [37] Duong Hieu Phan, David Pointcheval, Siamak Fayyaz Shahandashti, and Mario Strefer. Adaptive CCA broadcast encryption with constant-size secret keys and ciphertexts. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *ACISP 12: 17th Australasian Conference on Information Security and Privacy*, volume 7372 of *Lecture Notes in Computer Science*, pages 308–321, Wollongong, NSW, Australia, July 9–11, 2012. Springer, Heidelberg, Germany.
- [38] Duong Hieu Phan, David Pointcheval, and Mario Strefer. Security notions for broadcast encryption. In Javier Lopez and Gene Tsudik, editors, *ACNS 11: 9th International Conference on Applied Cryptography and Network Security*, volume 6715 of *Lecture Notes in Computer Science*, pages 377–394, Nerja, Spain, June 7–10, 2011. Springer, Heidelberg, Germany.
- [39] Tran Viet Xuan Phuong, Guomin Yang, Willy Susilo, and Xiaofeng Chen. Attribute based broadcast encryption with short ciphertext and decryption key. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *ESORICS 2015: 20th European Symposium on Research in Computer Security, Part II*, volume 9327 of *Lecture Notes in Computer Science*, pages 252–269, Vienna, Austria, September 21–25, 2015. Springer, Heidelberg, Germany.
- [40] Yannis Rouselakis and Brent Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13: 20th Conference on Computer and Communications Security*, pages 463–474, Berlin, Germany, November 4–8, 2013. ACM Press.

- [41] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.
- [42] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany.
- [43] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 53–70, Taormina, Italy, March 6–9, 2011. Springer, Heidelberg, Germany.
- [44] Hoeteck Wee. Dual system encryption via predicate encodings. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 616–637, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.
- [45] Shota Yamada, Nuttapong Attrapadung, Goichiro Hanaoka, and Noboru Kunihiro. A framework and compact constructions for non-monotonic attribute-based encryption. In Hugo Krawczyk, editor, *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 275–292, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany.

## Appendix A. Proof of Lemma 14

*Proof.* We prove for the general case where we allow all polynomials in  $P, Q$  to multiply with each other, which is exactly the symmetric pairing when  $P, Q$  are in the same group. For notational simplicity, we denote  $P = P \cup Q$ .

Suppose that  $f$  is not independent to  $(P, Q, R)$ , i.e., one can find  $a_{i,j}, c_i$  such that the following equation holds

$$f = \sum_{\{p_i, p_j\} \subset P} a_{i,j} \cdot p_i \cdot p_j + c_i$$

Assume that  $\Lambda_C$  is the list of corrupted users. We will use  $\beta$  to analyze  $f$ , set  $q_u = \text{ar}(\beta + s_u), u \in \Lambda_C, P' = P \setminus \{q_u\}_{u \in \Lambda_C}$ . We rewrite  $f$  as follows:

$$f = \sum_{\{u,v\} \subset \Lambda_C} a_{u,v} q_u q_v + \sum_{u \in \Lambda_C, p_i \in P'} a_{u,i} p_i q_u +$$

$$\sum_{\{p_i, p_j\} \subset P'} a_{i,j} p_i p_j + c_i = f_1 + f_2 + f_3$$

Consider  $f_1$ , we rewrite it as follows:

$$\begin{aligned} f_1 &= \sum_{\{u,v\} \subset \Lambda_C} a_{u,v} q_u q_v = \\ &= \sum_{\{u,v\} \subset \Lambda_C} a_{u,v} \alpha^2 r^2 (\beta^2 + \beta s_u + \beta s_v + s_u s_v) \end{aligned}$$

Since  $s_u, s_v$  are random elements thus the value  $a_{u,v} \alpha^2 r^2 s_u s_v$  is unique. On the other hand, this value doesn't appear in  $f = \alpha^{n+1} r \beta (mt + \sum_{k=1}^m \sum_{j \in \beta_k} x_j)$ , this leads to the fact that  $a_{u,v} = 0$  for any  $\{u, v\} \subset \Lambda_C$ , or we have  $f_1 = 0$ .

Consider  $f_2 = \sum_{u \in \Lambda_C, p_i \in P'} a_{u,i} p_i q_u$ , to let it appear the needed term  $\alpha^{n+1} r \beta$  we divide the polynomials  $p_i \in P'$  into two subsets, one containing the term  $\alpha^n$  denoted  $P'_1$ , and one doesn't denoted  $P'_2$ . We now rewrite  $f_2$  as follows:

$$\begin{aligned} f_2 &= \sum_{u \in \Lambda_C, p_i \in P'_1} a_{u,i} p_i q_u + \sum_{u \in \Lambda_C, p_i \in P'_2} a_{u,i} p_i q_u \\ &= \sum_{u \in \Lambda_C, p_i \in P'_1} a_{u,i} p_i \alpha r (\beta + s_u) + \sum_{u \in \Lambda_C, p_i \in P'_2} a_{u,i} p_i q_u. \end{aligned}$$

We therefore obtain the equation

$$f = \alpha^{n+1} r \beta (mt + \sum_{k=1}^m \sum_{j \in \beta_k} x_j) = \sum_{u \in \Lambda_C, p_i \in P'_1} a_{u,i} p_i \alpha r (\beta + s_u) + \sum_{u \in \Lambda_C, p_i \in P'_2} a_{u,i} p_i q_u + f_3 \quad (1)$$

Since the term  $\alpha^{n+1} r \beta$  only appear in  $\sum_{u \in \Lambda_C, p_i \in P'_1} a_{u,i} p_i \alpha r (\beta + s_u)$ , to make the equation (1) hold one needs to remove the term related to  $s_u$  in  $\sum_{u \in \Lambda_C, p_i \in P'_1} a_{u,i} p_i \alpha r (\beta + s_u)$ , and the only way to do that is to produce the term  $\sum_{u \in \Lambda_C, p_i \in P'_1} a_{u,i} p_i \alpha r s_u$  for each  $u \in \Lambda_C$ . On the other hand, to make the term

$$f = \alpha^{n+1} r \beta (mt + \sum_{k=1}^m \sum_{j \in \beta_k} x_j)$$

appear, the polynomial  $p_i, p_i \in P'_1$ , cannot have the form containing  $\alpha^n \beta$  or  $\alpha^n r$ , or  $\alpha^n s_u$  (if not, it will make the redundancy when multiplying with  $q_u = \alpha r (\beta + s_u)$ ). The only one such  $p_i$  comes from  $p_i = \alpha^n (mt + \sum_{k=1}^m \sum_{j \in \beta_k} x_j)$ . This leads to the fact that one only can produce the term

$$\sum_{u \in \Lambda_C} a_{u,i} \alpha^{n+1} r (\beta + s_u) (mt + \sum_{k=1}^m \sum_{j \in \beta_k} x_j)$$

That means one needs to produce the term related to  $s_u$ :

$$f' = \sum_{u \in \Lambda_C} a_{u,i} \alpha^{n+1} r s_u (mt + \sum_{k=1}^m \sum_{j \in \beta_k} x_j)$$

Since each user  $u \in \Lambda_C$  lacks at least one term  $\alpha^{n+1} r s_u (t + \sum_{j \in \beta_k} x_j)$  for some  $\beta_k$  and no one can help because of the unique value  $s_u$ , therefore one cannot reach to  $f'$ . That means the equation (1) cannot hold or  $f$  is independent to  $(P, Q, R)$ .

□