Combinatorial Subset Difference Public Key Broadcast Encryption Scheme for Secure Multicast*

Jihye Kim[1], Seunghwa Lee[1], Jiwon Lee[2], and Hyunok Oh[2(✉)]

[1] Kookmin University, Seoul, Korea,
{jihyek,ttyhgo}@kookmin.ac.kr
[2] Hanyang University, Seoul, Korea,
{jiwonlee,hoh}@hanyang.ac.kr

Abstract. Public key broadcast encryption is a cryptographic method to securely transmit a message from anyone to a group of receivers such that only privileged users can decrypt it. A secure multicast system allows a user to send a message to a dynamically changing group of users. The secure multicast can be realized by the broadcast encryption.
In this paper, we propose a novel *combinatorial subset difference* (CSD) public key broadcast encryption algorithm which allows a generalized subset different representation in which wildcards can be placed at any position. The proposed CSD is applicable to a secure multicast as well as minimizes the header size compared with existing public key broadcast encryption schemes without sacrificing key storage and encryption/decryption performance.
Experimental results show that the proposed CSD scheme not only reduces the ciphertext header size by 17% and 31% but also improves encryption performance (per subset) by 6 and 1.3 times, and decryption performance by 10 and 19 times compared with existing efficient subset difference (SD) and interval schemes, respectively. Furthermore, especially for subsets represented in a non-hierarchical manner, the proposed CSD reduces the number of subsets by a factor of 1000 times compared with SD and interval approaches.
We prove semantic security of our proposed CSD scheme under l-BDHE assumption without the random oracle model.

Keywords: Broadcast encryption, secure multicast, wildcard, subset difference, public key

1 Introduction

Broadcast encryption (BE) is a cryptographic method to securely transmit a message to a group of receivers such that only privileged users can decrypt

* This paper is a full version of the conference paper with the same title that was published in the ACM/SIGAPP Symposium on Applied Computing, 2018

them. Practical applications include Pay-TV and more generally digital rights management. In IoT infrastructures such as an enterprise control system, BE can also be used for a secure multicast.

Numerous BE schemes have been introduced with different features [25, 26, 9, 13, 7, 8, 17, 28, 11, 3, 6, 16, 18, 20, 24, 14, 27, 10, 12]. BE schemes can be constructed either in a symmetric key setting or a public key setting. For a public key BE scheme, anyone can broadcast a message to a changeful set of authorized users by using the public key of the scheme, while for a secret key BE scheme, only a special center, who knows the system secrets, can broadcast a message. BE schemes might be stateful or stateless. For a stateful BE scheme, the private key of a user can be updated from time to time, while the private keys of a user in a stateless BE scheme remain the same through the lifetime of the system. The security can be defined against fully or partially colluding adversaries. The schemes could be integrated with the revocation and traitor-tracing algorithms. They might support an ID based public key setting and/or forward security. For the purpose of this paper, we consider a *public-key* broadcast encryption system with *stateless* receivers, which is *fully* collusion-resistant.

In BE, a message format is generally represented as $(S,\ Hdr, C_M)$, where S represents a group of users, Hdr is a header, and C_M is a ciphertext of an original message. A sender generates a ciphertext C_M of a message M with a session key K_s, computes Hdr with S and K_s, and transmits (S, Hdr, C_M). To decrypt, a privileged user u in S computes K_s from its private key and the header Hdr, and decrypts the ciphertext. But any revoked user should not be able to deduce K_s with his/her private information. Furthermore, K_s should not be computable even if all (revoked) users not in S collude.

The important metrics to estimate BE schemes are the length of the header (as given by the number of encryptions of the session key), the private/public key size, and the encryption/decryption time. It is desirable to decrease them as far as possible, but, in most schemes it turns out that decreasing one increases the other. In broadcast encryption where a message is transmitted to a huge number of receivers, the main issue is to minimize the header length with practical computation cost and storage size of a user.

One of the most notable BE schemes is proposed by Naor et al. [24]. Considering that the method to represent the subset of users affects the transmission overhead, the scheme in [24] is a pioneer to provide an efficient subset cover algorithm called subset difference (SD). The SD algorithm is effectively applicable to any BE scheme for the representation of S. In many BE schemes [24, 18, 20, 14] the subset cover algorithm is crucial since the size of header is proportional to the number of subsets represented. So far, the SD method has become popular in most broadcast encryption applications and is already implemented in the DVD standard [1].

Secure Multicast. A secure multicast can be efficiently realized by BE since BE can determine any group of legitimate users as its destinations. In particular, we consider a special multicast setting where an ID is defined according to meanings (or attributes). In this setting, a certain group can be represented by

simply specifying the corresponding meanings. For instance, in an IP network IP address can be used to represent an organization, a physical region, and device types (or users).[3] In this setting, an administrator might need to transmit a message for firmware updates privately to every router device in a certain organization $a.b$.xxx.xxx, where all router devices are assigned as xxx.xxx.xxx.1 (e.g. $a.b.*.1$, where "$*$" denotes wildcard bits). In a more sophisticated case, the administrator might need to transmit a secure update message excluding routers in a certain compromised region R (e.g., $a.b.*.1 - a.b.R.1$). We can take another example of smart grid where a smart meter is identified by attributes of (location, model, company). Then a service provider might want to send critical software update to the smart meters of a certain model located at a certain area manufactured by a certain company. The smart meter might also want to deliver its collected date only to collectors in agreement. After the agreement is broken with certain collectors, the smart meter needs to exclude those collectors. Notice that each attribute can be mapped to binary bits and similar scenarios can be applied to a variety of secure multicast applications.

Unfortunately, although it seems intuitively convenient and efficient to express a group of legitimate users according to meanings or attributes, BE schemes should reconstruct it into subsets according to their own representation, which is not as compact as the original expression in general. Moreover, in most BE schemes, the size of ciphertexts grows in proportion to the number of subsets. As an illustration, to cover a group of gateway devices denoted as $a.b.*.1$ (i.e., $a.b.0.1, \cdots, a.b.255.1$), the SD method [24] requires 256 subset representations (i.e., $a.b.0.1-null, \cdots, a.b.255.1-null$) requiring large ciphertexts. This is mainly because the SD method is restricted in a hierarchical structure. To deal with this problem, we propose a new BE scheme that can embrace non-hierarchically structured secure multicast. We discuss more about the limitations of SD method and contributions of our proposed method in the following subsections.

Combinatorial Subset Difference. We propose a new combinatorial subset difference (CSD) representation which can express naturally a set of legitimate users in a non-hierarchical manner. In CSD, a set of users is represented as boolean expressions additionally with *wildcards* ("$*$") which can be either 0 or 1.

The CSD subset representation is analogous to SD in the sense that both denote a subset using a difference of two subsets. Briefly, existing SD algorithm expresses legitimate users with a subtraction of an inclusion complete subtree and an exclusion complete subtree. Each leaf node holds the key of co-path nodes in a tree so that the non-revoked users can decrypt the message unless they do not belong to the exclusion complete subtree. Similarly, CSD presents an expression of legitimate users with a subtraction of an inclusion subset and an exclusion subset. However, CSD represents each inclusion/exclusion subset using *non-hierarchical* representation (or label) in $\{0, 1, *\}^l$ where the maximum

[3] In fact it is also common in real protocols. First and second byte of an IP address usually represents a group hierarchy, and third and fourth byte usually represents local regions and device types (or users).

number of users is $n = 2^l$. Thus, a subset is denoted as $(\mathbf{c}, \mathbf{d})$, where labels c and d show an inclusion set and an exclusion set, respectively.

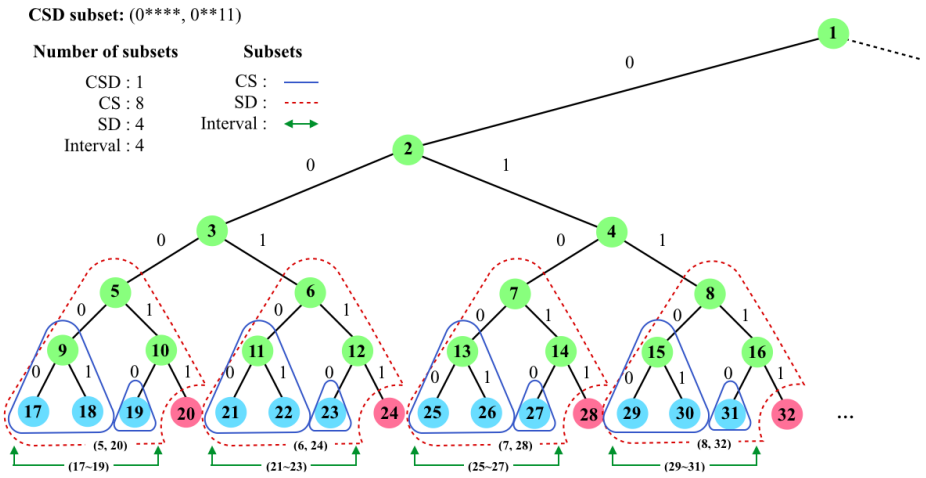


Fig. 1: Subset construction example in CS, SD, interval and CSD algorithms

In fact, the SD method is a special case of the CSD method. Figure 1 is a binary tree structure with labeled nodes, showing subsets in CSD, CS (complete subtree), SD, and interval algorithms. The leaf nodes $(17 - 32)$ indicate users where red nodes are revoked. A solid line indicates CS subsets, a dotted line denotes SD subsets, and an arrowed line presents intervals. Note that every node can be represented with the boolean expression using wilcards ("$*$"): node 4 can be expressed as $01 * **$, and node 11 can be expressed as $0010*$ for example. In this context, node 10 ($0001*$) defines a subtree with nodes 19 (00010) and 20 (00011). In CS and SD labels, no binary bit is followed after the wildcards due to the hierarchical tree structure. On the contrary, CSD can completely embrace general representation, meaning that wildcards can be placed anywhere. For instance, notation $** 00*$ is admitted in CSD, while it is not allowed in SD since there is no such node in the binary tree. Since the CSD approach employs more general labeling forms, the number of subsets to cover legitimate users in CSD is no more than in SD. As shown in Figure 1, consider that four users (nodes) 20, 24, 28, 32 (e.g. 00011, 00111, 01011, 01111) are revoked among all 32 users[4]. When using the CS approach, 8 subset labels 9, 19, 11, 23, 13, 27, 15, 31 are required to cover all non-revoked users. With the SD approach, 4 subset labels $(4, 19)$, $(5, 23)$, $(6, 27)$, $(7, 31)$ are required to exclude the four revoked users. In interval algorithm, 4 intervals (equivalent to subsets) $(17 - 19)$, $(21 - 23)$, $(25 - 27)$, $(29 - 31)$ are required to cover all non-revoked users. However, in

[4] Note that there are 16 omitted users which are revoked on the right subtree side.

contrast to CS, SD, and interval schemes, CSD approach covers all legitimate users with only a single subset ($\mathtt{0****}, \mathtt{0**11}$).

Contributions. We design a new public key broadcast encryption scheme that supports the non-hierarchical CSD representation. The scheme has a short key size: public key size of $O(\log n)$, and secret key size of $O(\log^2 n)$. The scheme performs encryption/decryption faster than the existing efficient BE schemes such as the subset difference scheme [14] and the interval encryption scheme [23]. More importantly, its header size is minimum: the number of subsets is r for r revoked users, and the number of group elements in a ciphertext (per subset) is only two.

Table 1: Comparison of public-key based BE schemes. *cf.* $\boldsymbol{e}$ = *point multiplication,* $\boldsymbol{p}$ = *pairing,* n = *the number of total users, and* r = *the number of revoked users*

	CSD	**SD [14](HIBE [3])**	**Interval [23]**
PK Size	$4\log n$	$\log n$	$2\log n$
SK Size	$3\log^2 n$	$\log^3 n$	$2\log^2 n$
Hdr Size	$2r$	$4r$	$3r$
# subsets	r	$2r$	r
# group elements	2	2	3
Enc Time	$3r\mathbf{e}$	$2r\mathbf{e}\log n$	$4r\mathbf{e}$
Dec Time	$2\mathbf{e}+2\mathbf{p}$	$\mathbf{e}\log n+2p$	$2\mathbf{e}\log n+4\mathbf{p}$
Assumption	$l-BDHE$	$l-BDHI$	$l-BDHE$

Table 1 summarizes general metrics that affect the efficiency of transmission time and device storage in the proposed CSD, SD, and interval schemes. In the table, SD is assumed as public key broadcast encryption [14] implemented with HIBE [3]. We calculate metrics without *Big-O* notation, to specify more concrete difference among the schemes. Note that constant values and point additions are ignored.

A header size is a main factor that determines the transmission cost, since transmission consists of header, along with set of groups or revocation list. There were many efforts to reduce the header size in a broadcast encryption area, and in the state of the art SD scheme had cut it down to $4r$ for r revoked users. Notice that the header size is calculated by multiplying the number of subsets and the number of group elements. Our proposed CSD scheme requires a header size of $2r$, while SD and interval schemes request $4r$ and $3r$ header size. In addition, the CSD scheme outperforms SD in encryption; the SD has point multiplications logarithmically proportional to the number of users while CSD has constant point multiplications. The CSD scheme outperforms both SD and interval schemes in decryption; the CSD has constant point multiplications while the SD and interval schemes require point multiplications logarithmically proportional to the number of users.

According to our experimental results, our scheme minimizes the ciphertext header size, and improves encryption and decryption performance compared with existing efficient public key broadcast SD and interval schemes. The proposed scheme reduces the header size by 17% and 31%, and accelerates encryption per subset by 6 and 1.3 times and decryption by 10 and 19 times compared with SD and interval schemes, respectively. Furthermore, especially for subsets represented in a non-hierarchical manner, the proposed CSD reduces the number of subsets by a factor of 1000 times compared with SD and interval approaches. We prove security of our CSD broadcast encryption scheme based on (decisional) l-BDHE assumption without the random oracle model.

Section 2 summarizes related works. Section 4 describes preliminaries of the proposed broadcast encryption scheme. We propose our secure broadcast scheme in section 5. Section 7 represents experiment results. In section 9 we draw a conclusion.

2 Related Work

There have been various broadcast encryption schemes with different flavors [25, 26, 9, 13, 7, 8, 17, 28, 11, 3, 6, 16, 18, 20, 24, 14, 27, 10, 12]. BE schemes can use either symmetric key cryptography or public key cryptography. They could be integrated with the revocation and traitor-tracing algorithms. They have stateful or stateless receivers. They are fully or partially collusion resistant. They might support an ID based public key setting and/or forward security. In particular, we focus on fully collusion resistant public key encryption with stateless receivers using a *new subset representation method*. In the following, we briefly describe some notable symmetric key and asymmetric key broadcast encryption schemes primarily considering this restricted context.

In symmetric key based BE, Fiat and Naor [16] presented the first formal model of the broadcast encryption in a symmetric key setting. They introduced the resiliency concept, and defined k-resilience, which means being resilient against a coalition of up to k revoked users. Their scheme required every receiver to store $O(k \log k \log n)$ keys and the center to broadcast $O(k^2 \log^2 k \log n)$ messages where n is the total number of users.

Naor, Naor and Lotspiech proposed two stateless schemes based on a symmetric key setting, called complete subtree (CS) and subset difference (SD) [24] depending on the subset cover method. The CS scheme has a transmission cost of $O(r \log(n/r))$, r denoting the number of revoked users. The SD scheme decreased the transmission overhead to $O(r)$, at the expense of increasing the key storage to $O(\log^2 n)$. The SD scheme was the most efficient scheme at the time of its proposal, and most of the recent schemes proposed since then are still based on the SD scheme in the matter of covering the subsets.

Variations of SD offering a tradeoff between user storage and bandwidth were proposed. Layered subset difference [20] and stratified subset difference [18] have worse bandwidth performance than the original SD scheme, but can decrease the user storage.

For an asymmetric key setting, Naor and Pinkas [25, 26] first presented a public key broadcast encryption scheme. Their scheme represented a threshold secret sharing method, and was not fully collusion resistant.

Dodis and Fazio [14] proposed a generic method to build subset difference (SD) public key broadcast encryption from any hierarchical identity based encryption. Their work can be efficiently constructed using HIBE [3] which has constant-size encryption.

Boneh, Gentry, and Waters (BGW) [6] introduced special and general public key broadcast encryption schemes which are fully collusion-resistant. Their special scheme requires both the broadcast ciphertext and a user storage of constant size with the cost of an n-sized public key. To provide a tradeoff between the ciphertext size and the public key size, they constructed a generalized scheme of the special BGW scheme by dividing a whole set into a number of subsets. Their general scheme achieves $O(\sqrt{n})$-sized public key, $O(\sqrt{n})$ transmission cost and $O(1)$ user storage cost. Since the public key is a part of input in decryption, additional $O(\sqrt{n})$ overhead should be considered either in the user storage or the message transmission.

Lee, et al. [22] proposed a revocation scheme based on SD algorithm with construction from a single revocation encryption scheme to a general public key revocation encryption scheme. They reduced the key size to $O(\log^2 n)$, compared with [14] which is $O(\log^3 n)$. The security of their scheme is based on a random oracle model.

Lin, et al. [23] introduced a new context of interval encryption, which can be constructed from binary tree encryption. The idea was to bind non-revoked users within intervals that is split by revoked users. Each user in an interval has two HIBE [3] keys that can make user keys for the left and the right sides, with a size of $O(\log^2 n)$. In the interval scheme, a valid user can have all the necessary keys within the interval, and decrypt incoming broadcast messages. The interval scheme requires r subsets for r revoked users. However, to present a subset, it requests three group elements in which one element denotes a random value, and the other two elements present left and right indices for an interval representation while the other schemes based on SD have two group elements for a subset.

Broadcast encryption can be constructed from attribute-based encryption (ABE) [19] with an access policy that expresses an arbitrary set of privileged users. However, existing ABE schemes suffer from a large ciphertext size problem and/or are not compatible with the standard SD method. The authors in [15] proposed a CP-ABE scheme with constant ciphertext size, but, without supporting wildcards in its access policy. Later, the ABE scheme in [29] was upgraded to support wildcards in its access policy so that it is applicable to a BE scheme with the CS method. But the CS method itself incurs the header communication overhead of $O(\log n/r)$.

3 Proposed Subset Construction Algorithm

In this section, we propose a heuristic algorithm to construct subsets for combinatorial subset difference (CSD). Although the proposed subset construction algorithm does not minimize the number of subsets optimally, it is guaranteed that it generates no more subsets than the existing subset difference (SD) algorithm. In addition, the proposed heuristic algorithm generates no more than r subsets for r revoked users even in the worst case while $2r - 1$ subsets are generated in the SD algorithm.

Since SD is a special case of CSD, we will first explain how to minimize the number of subsets in the existing SD algorithm in our frame and then show how to extend the SD algorithm to the CSD algorithm focusing on the difference between the SD algorithm and the proposed CSD algorithm.

Recall that a subset in SD is represented as $S = (c, d)$, where label c indicates the nodes to include, and label d presents the nodes to exclude. For example, in a tree structure, S includes all descendants of c, except all descendants of d.

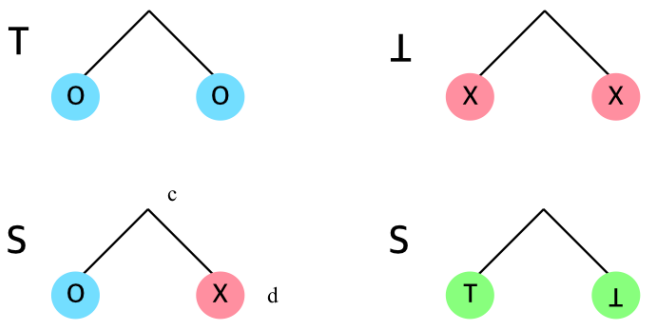


Fig. 2: Node types where node O is a legitimate node and node X is a revoked node.

In the SD (CSD) algorithm, an internal node c is categorized as one of following three types depending on its children nodes, as illustrated in Figure 2.

- Type $\top$: There is no revoked descendant node of c.
- Type $\bot$: Node c should be excluded by its parent node since either it is already covered by other subsets or its all descendant nodes are revoked.
- Type S : Node c becomes a subset (c, d) where all descendants of node d are already covered by some other subsets or revoked.

Since the node type is determined by the node types of its children, the internal node type is determined recursively from the bottom to the top. Considering

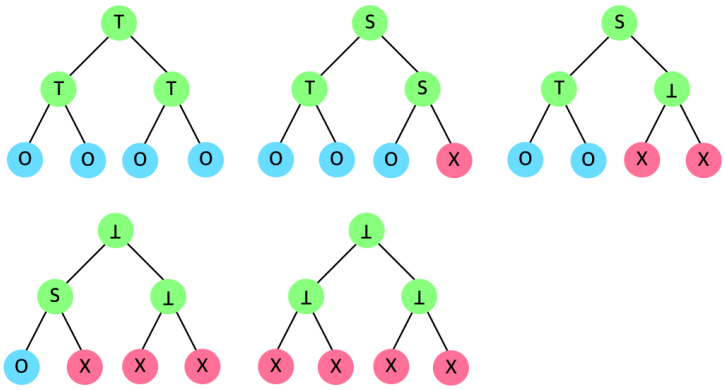


Fig. 3: Node type examples in SD (and CSD)

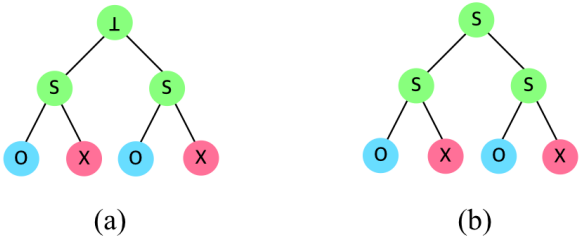


Fig. 4: Category difference between (a) SD and (b) CSD

the node type definition in Figure 2, combining of two internal nodes identifies their parent node type as in Figure 3. In particular, the examples in Figure 3 are also valid for CSD as well. However, SD and CSD show difference when they combine two S type nodes as shown in Figure 4; a parent node of two S nodes is represented as $\perp$ in SD, but CSD represents it as another S. In the SD algorithm, a set can be bound only if lower level nodes are all $\top$ or $\perp$. Thus two S type nodes cannot be merged as a single set and the type of the parent node becomes $\perp$ since the two S type nodes should be represented as individual subsets in SD. However in CSD, the type of the parent node becomes S without generating two subsets for the two S type nodes by merging them.

Consider an example in which the number of users is 8, and users 1 and 3 are revoked, as illustrated in Figure 5. In a binary format, users 001 and 011 are revoked. Using $*$, we denote parent nodes. For example, node $00*$ becomes a parent of nodes 000 and 001, and node $01*$ becomes a parent of nodes 010 and 011. Node 000 has type $\top$ since nothing is revoked and node 001 has type $\perp$ since it is revoked. Node $00*$ has type S since one children node 000 is included while

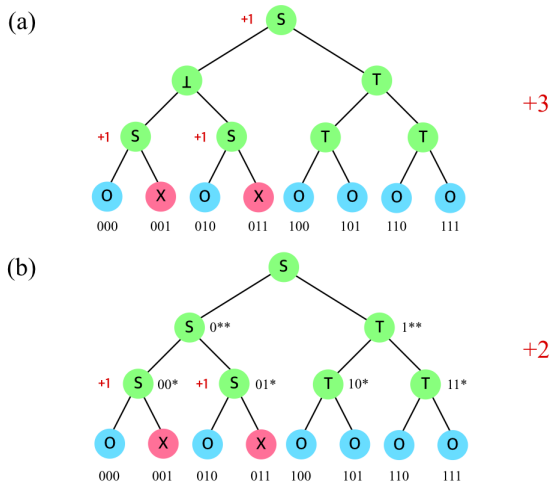


Fig. 5: (a) Subset construction in SD, and (b) subset construction in CSD.

the other one 001 is excluded, which requires to build a subset $(00*, 001)$ to be covered. Similarly, node $01*$ has type S. Consider node $0**$ of which children have type S. Node $0**$ cannot be a inclusion label in a subset since both its children nodes are labeled as two different subsets, $(00*, 001)$ and $(01*, 011)$, which cannot be merged. Therefore, node $0**$ should be excluded and it has type $\perp$. The root type is S since the left child node has $\perp$ and the right child node has $\top$ as type. Consequently, three subsets of $(***, 0**)$, $(00*, 001)$, and $(01*, 011)$ are generated.

On the contrary, in the proposed CSD, node $0**$ is treated specially. Even if two children nodes have type S, it is not required to exclude the node in subset difference presentation like SD. Indeed, if the node is merged with its sibling node of which type is $\top$, it is possible to represent the node without creating a new subset. For instance, for the above example, consider node $0**$ which has two subsets of $(00*, 000)$ and $(01*, 010)$, and its sibling node $1**$ of which type is $\top$ meaning that all descendant nodes are included. The merge of subsets $(00*, 000)$ and $(01*, 010)$, and subset $(1**, \perp)$ generates subsets $(*0*, 000)$ and $(*1*, 010)$ in which inclusion labels $*0*$ and $*1*$ include all eight users, and exclusion labels 000 and 010 exclude users 000 and 010 since the proposed CSD allows $*$ in the middle of subset representation. Finally, the CSD algorithm generates two subsets of $(*0*, 001)$ and $(*1*, 011)$ while the SD requires three subsets. Indeed, the proposed CSD algorithm does not create subsets more than the number of revoked users.

Table 2 summarizes the resolved type of a node when its children nodes' types T_0 and T_1 are given in the first and second columns. The resolved types in SD and CSD are shown in the third and fourth columns, respectively. The

last column illustrates whether subset generation is required or not where 0 does not create a subset and 1 does. Note that a new subset is required if one child node is included and the other one is excluded. The only difference between the existing SD and the proposed CSD is that when both children nodes have type S the parent type is $\perp$ in SD while it is S in CSD.

Table 2: Type solution in SD and CSD algorithms

T_0	T_1	Type in SD	Type in CSD	subset generation
$\perp$	$\perp$	$\perp$	$\perp$	0
$\perp$	$\top$	S	S	1
$\perp$	S	$\perp$	$\perp$	0
$\top$	$\top$	$\top$	$\top$	0
$\top$	S	S	S	0
S	S	$\perp$	S	0

Algorithm 1 represents how to construct the subsets in both SD algorithm and CSD algorithm. The only difference exists at line 27. While the algorithm for CSD executes line 27, it does not for SD. In addition, $resolveType(T_0, T_1)$ at line 29 returns different values as shown in table 2.

The subset construction algorithm starts with a revoked user ID set. Function *sd* examines the IDs of the revoked users from the most significant bit to the least significant bit. The current watching index is denoted as i. When index i reaches the least significant bit(LSB), the user is either revoked or legitimate. Hence it returns $\perp$ or $\top$ as shown in lines 2 to 7. Even if index i does not reach LSB, if there is no revoked user then it returns $\top$. At lines 11 and 12, the revoked users are divided into two sets depending on user's ID at index i, and then function *sd* is recursively called for each set, which denotes tree traverse at lines 13 and 14. If the types of children nodes are $\perp$ and $\top$ respectively, a new subset is created as shown at lines 16 and 19. Function $createSubset(prefix, i, b)$ creates a subset (c, d) where c is an inclusion label and d is an exclusion label. Label c covers all nodes with $prefix_1, \ldots, prefix_i, *, \ldots$ and label d covers all nodes with $prefix_1, \ldots, prefix_i, b, *, \ldots$ where $prefix_k$ denotes a $k-th$ bit of $prefix$. When one child has type S and the other child has type $\top$, each inclusion label in an unresolved subset is revised to include the child node with $\top$ as shown at lines 22 and 24. In the proposed CSD algorithm, the unresolved subsets are preserved when children types are S since inclusion labels in the unresolved subsets will be revised if they are required to include a node with $\top$ at line 27.

Theorem 1. *Algorithm 1 generates r subsets at most for r revoked users in a combinatorial subset difference algorithm.*

Proof. As shown in table 2, a subset is generated when the type of a child node is $\perp$. The resolved type becomes $\perp$ only if the type of a child node is $\perp$ at least. A leaf node (user) has type $\perp$ only if it is a revoked user. If there are r revoked

Algorithm 1 Subset construction for SD and CSD

function $sd(Set, Unresolved, i, prefix, revSet)$
  if $i = depth$ **then**
    if $|revSet| > 0$ **then**
      return $\perp$
    else
      return $\top$
    end if
  else if $|revSet| = 0$ **then**
    return $\top$
  end if
  $S_0 = \{r | r \in revSet, bit(r, i+1) = 0\}$
  $S_1 = \{r | r \in revSet, bit(r, i+1) = 1\}$
  $T_0 \leftarrow sd(Set, U_0, i+1, prefix||0, S_0)$
  $T_1 \leftarrow sd(Set, U_1, i+1, prefix||1, S_1)$
  if $(T_0 = \perp) \wedge (T_1 = \top)$ **then**
    $s \leftarrow createSubset(prefix, i, 0)$
    $Unresolved \leftarrow s$; $Set \leftarrow Set \cup s$
  else if $(T_0 = \top) \wedge (T_1 = \perp)$ **then**
    $s \leftarrow createSubset(prefix, i, 1)$
    $Unresolved \leftarrow s$; $Set \leftarrow Set \cup s$
  else if $(T_0 = S) \wedge (T_1 = \top)$ **then**
    $\forall (c, d) \in U_0, c_i \leftarrow *$; $Unresolved \leftarrow U_0$
  else if $(T_0 = \top) \wedge (T_1 = S)$ **then**
    $\forall (c, d) \in U_1, c_i \leftarrow *$; $Unresolved \leftarrow U_1$
  else if $Combinatorial \wedge (T_0 = S) \wedge (T_1 = S)$ **then**
    // Only used for CSD
    $Unresolved \leftarrow U_0 \cup U_1$
  end if
  return $resolveType(T_0, T_1)$
end function
procedure $main$
  $revSet \leftarrow \{revoked\ user\ IDs'\}$
  $header \leftarrow sd(\perp, \perp, 0, \perp, revSet)$
end procedure

nodes, r leaf nodes have type $\perp$ at most. Consequently, the number of created subsets is no more than r.

In the SD algorithm, since the resolved type of a node can be $\perp$ even if both children nodes have type S, the number of subsets can be more than r for r revoked users.

It is easy to deduce that the number of generated subsets in CSD is no larger than SD. A subset is generated when a child node has type $\perp$ and the other child has type $\top$ in both SD and CSD. In both schemes, a node can have type $\top$ when its children nodes have type $\top$, and it has type $\perp$ when a child has type $\perp$ and the other one has type $\perp$ or S. Additionally, in SD, a node has type $\perp$ when its children have type S. Hence, the number of subsets in CSD is no more than SD.

4 Preliminaries

4.1 Public Key Broadcast Encryption

A public key broadcast encryption (PKBE) system Π consists of three algorithms:

$\mathsf{Setup}(l, m)$ Takes as input user's ID length l and session key length m. It outputs a public key PK and 2^l private key sets $\{SK_{ID}\}_{ID\in\{0,1\}^l}$, one for each user with l-bit ID.

$\mathsf{Encrypt}(PK, S)$ Takes as input a subset $S \subseteq \{0,1\}^l$, and a public key PK. It outputs a pair (Hdr, K) where Hdr is called the header and $K \in \mathcal{K}$ is a message encryption key. Let M be a message to be broadcast to the set S and let C_M be the encryption of M under the symmetric key K. The broadcast to users in S is composed of (S, Hdr, C_M). The pair (S, Hdr) is often called the full header, Hdr is often called a broadcast ciphertext, and C_M is often called the broadcast body.

$\mathsf{Decrypt}(S, ID, SK_{ID}, Hdr)$ Takes as input a subset $S \subseteq \{0,1\}^l$, a user id $ID \in \{0,1\}^l$, the private key SK_{ID} for user ID, and a header Hdr. If $ID \in S$, then the algorithm outputs the message encryption key $K \in \mathcal{K}$. The key K can then be used to decrypt the broadcast body C_M and obtain the message body M.

The system is correct if all users in S can get the broadcast message M. Namely, for all $S \subseteq \{0,1\}^l$ and all $ID \in S$, if $(PK, (SK_{ID_{0^l}}, \ldots, SK_{ID_{1^l}})) \xleftarrow{\$} \mathsf{Setup}(\mathsf{l}, \mathsf{m})$ and $(Hdr, K) \xleftarrow{\$} \mathsf{Encrypt}(\mathsf{PK}, \mathsf{S})$ then $\mathsf{Decrypt}(\mathsf{S}, \mathsf{ID}, \mathsf{SK_{ID}}, \mathsf{Hdr}) = \mathsf{K}$.

In the following section, we describe selective chosen ciphertext security against adaptive chosen ciphertext attacks for broadcast encryption as in [6]. Depending on whether the number of the challenged set is represented as a single subset or multiple subsets, we separate security notions as single-set security and multi-set security. Finally, we show the single-set security implies a multi-set security.

4.2 Security

The single-set security is defined by an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ via following game. Both the $\mathcal{C}$ and $\mathcal{A}$ are given l and m, the user ID length in bits and the key length in bits, as an input.

Init: Algorithm $\mathcal{A}$ begins outputting a set S^* of users that it intends to attack.
Setup: The challenger $\mathcal{C}$ runs $\mathsf{Setup}(\mathsf{l}, \mathsf{m})$ to obtain a public key PK and private keys $SK_{0^l}, \cdots, SK_{1^l}$. $\mathcal{C}$ gives $\mathcal{A}$ the public key PK and all private keys SK_{ID} for $ID \notin S^*$.
Query phase1: Attacker $\mathcal{A}$ adaptively issues decryption queries $q_1, \ldots, q_m$ where a decryption query consists of the triple (ID, S, Hdr) with $S \subseteq S^*$ and $ID \in S$. $\mathcal{C}$ responds with $\mathsf{Decrypt}(\mathsf{S}, \mathsf{ID}, \mathsf{SK_{ID}}, \mathsf{Hdr})$.
Challenge: $\mathcal{C}$ runs algorithm $\mathsf{Encrypt}(\mathsf{S}^*, \mathsf{PK})$ to obtain (Hdr^*, K) where $K \in \mathcal{K}$. Next, it picks $b \in \{0, 1\}$. If $b = 1$, it sets $K^* = K$, otherwise it picks a random $R \in \mathcal{K}$ and sets $K^* = R$. It gives (Hdr^*, K^*) to attacker $\mathcal{A}$.
Query phase2: Attacker $\mathcal{A}$ continues to adaptively issue decryption queries q_{m+1}, $\ldots$, q_{q_D} where a decryption query consists of (ID, S, Hdr) with $S \subseteq S^*$ and $ID \in S$. The only constraint is that $Hdr \neq Hdr^*$. $\mathcal{C}$ responds as in query phase 1.
Guess: Attacker $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ for b and wins the game if $b = b'$.

Let $\mathsf{AdvSSBr}_{\mathcal{A},\Pi}(\mathsf{l}, \mathsf{m})$ be the advantage that $\mathcal{A}$ wins the above game.

Definition 1. *A single-set public key broadcast encryption system Π is (t, ϵ, l, m, q_D)-CCA secure if for every t-time adversary $\mathcal{A}'$ that makes at most q_D decryption queries, we have that $|\mathsf{AdvSSBr}_{\mathcal{A},\Pi}(\mathsf{l}, \mathsf{m}) - 1/2| < \epsilon$.*

The multi-set security is defined by an adversary $\mathcal{A}'$ and a challenger $\mathcal{C}$ as in the above the game for the single-set security. Only difference is that the challenged set is represented with multiple subsets.

Init: Algorithm $\mathcal{A}'$ begins outputting a set $S^* = (S_1^*, \cdots, S_w^*)$ of users that it intends to attack.
Setup: The challenger $\mathcal{C}$ runs $\mathsf{Setup}(\mathsf{l}, \mathsf{m})$ to obtain a public key PK and private keys $SK_{0^l}, \cdots, SK_{1^l}$. $\mathcal{C}$ gives $\mathcal{A}'$ the public key PK and all private keys SK_{ID} for $ID \notin S^*$.
Query phase1: Attacker $\mathcal{A}'$ adaptively issues decryption queries $q_1, \ldots, q_m$ where a decryption query consists of the triple (ID, S, Hdr) with $S \subseteq S_i^*$ for all i and $ID \in S$. $\mathcal{C}$ responds with $\mathsf{Decrypt}(\mathsf{S}, \mathsf{ID}, \mathsf{SK_{ID}}, \mathsf{Hdr})$.
Challenge: $\mathcal{C}$ runs algorithm $\mathsf{Encrypt}(\mathsf{S_i^*}, \mathsf{PK})$ where $i = 1, \cdots, w$ to obtain (Hdr_i^*, K_i) where $K \in \mathcal{K}$. Next, it picks $b \in \{0, 1\}$. If $b = 1$, it sets $K^* = (K_1, \cdots, K_w)$, otherwise it picks a random $R_i \in \mathcal{K}$ where $i = 1, \cdots, w$ and sets $K^* = (R_1, \cdots, R_w)$. It gives (Hdr^*, K^*) to attacker $\mathcal{A}'$.
Query phase2: Attacker $\mathcal{A}'$ continues to adaptively issue decryption queries q_{m+1}, $\ldots$, q_{q_D} where a decryption query consists of (ID, S, Hdr) with $S \subseteq S_i^*$ for all i and $ID \in S$. The only constraint is that $Hdr \neq Hdr^*$. $\mathcal{C}$ responds as in query phase 1.

Guess: Attacker $\mathcal{A}'$ outputs its guess $b' \in \{0,1\}$ for b and wins the game if $b = b'$.

Let $\mathsf{AdvMSBr}_{\mathcal{A}',\Pi}(\mathsf{l},\mathsf{m})$ be the advantage that $\mathcal{A}'$ wins the above game.

Definition 2. *A multi-set public key broadcast encryption system Π' is (t,ϵ',l,m,q_D) CCA secure if for every t-time adversary $\mathcal{A}'$ that makes at most q_D decryption queries, we have that $|\mathsf{AdvMSBr}_{\mathcal{A}',\Pi'}(\mathsf{l},\mathsf{m}) - 1/2| < \epsilon$.*

Finally, we show that the single-set security implies the multi-set security.

Theorem 2. *Suppose the single-set public key broadcast encryption system Π is (t,ϵ,l,m,q_D)-CCA secure. Then multi-set public key broadcast encryption system Π' is (t,ϵ',l,m,q_D)-CCA secure for arbitrary $\epsilon' < \epsilon * w$, where w is the number of subsets [22].*

Proof. The basic idea of the proof is to convert the challenge key of the multi-set scheme from a real key set K^* to a random key set $\overline{K}^*$ by using hybrid games that change each key of the single-set scheme from a real key to a random key. If an adversary cannot distinguish the changes of each key of the single-set scheme with more than a non-negligible probability, then it also cannot distinguish the changes of the challenge key of the multi-set scheme with more than a non-negligible probability since the number of hybrid games is just polynomial. Suppose that a challenge set is given as $S^* = (S_1, S_2, \cdots, S_w)$ for polynomial w and the corresponding key set is described as $K^* = (K_1, \cdots, K_w)$ s.t. K_i is the key for S_i. The hybrid games $G_0, \cdots, G_h, \cdots, G_w$ for the proof are defined as follows:

Game G_0 In this game, all keys K_j are real key from an encryption on the set S_j. That is, the challenge key K^* is a set of real keys.

Game G_h This game is almost identical to the game G_{h-1} except the key K_h since K_h in this game is a random key. Specifically, in this game, the key K_j for $j \leq h$ is a random key and the key K_j for $h < j$ is a real key.

Game G_w In this game, all keys K_j are random keys. That is, the challenge key K^* is a set of random keys $\overline{K^*}$.

Let $S_{\mathcal{A}}^{G_h}$ be the event that $\mathcal{A}$ outputs 1 in G_h. $\mathcal{A}$ distinguishes G_{h-1} from G_h by the advantage of the single-set security. Thus, we have that

$$\begin{aligned} & Pr[S_{\mathcal{A}}^{G_0}] - Pr[S_{\mathcal{A}}^{G_w}] \\ & \quad = Pr[S_{\mathcal{A}}^{G_0}] + \sum_{h=1}^{w-1} (Pr[S_{\mathcal{A}}^{G_h}] - Pr[S_{\mathcal{A}}^{G_h}]) - Pr[S_{\mathcal{A}}^{G_w}] \\ & \quad \leq \sum_{h=1}^{w} |Pr[S_{\mathcal{A}}^{G_{h-1}}] - Pr[S_{\mathcal{A}}^{G_h}]| \leq 2w \cdot \mathsf{AdvSSBr}_{\mathcal{A}}(\lambda) \end{aligned}$$

Finally, we obtain the following inequality relation as

$$\begin{aligned}
&\mathsf{AdvMSBr}_{\mathsf{A}'}(\lambda) = |\mathsf{Pr}[\mathsf{b}=1]\cdot \mathsf{Pr}[\mathsf{b}=\mathsf{b}'|\mathsf{b}=1] \\
&\quad + Pr[b=0]\cdot Pr[b=b'|b=0] - \frac{1}{2}| \\
&= |\frac{1}{2}\cdot Pr[b'=1|b=1] + \frac{1}{2}\cdot(1-Pr[b'=1|b=0]) - \frac{1}{2}]| \\
&= \frac{1}{2}\cdot |Pr[b'=1|b=1] - Pr[b'=1|b=0]| \\
&\leq \frac{1}{2}\cdot |Pr[S^{G_0}_{\mathcal{A}}] - Pr[S^{G_w}_{\mathcal{A}}]| \leq w\cdot \mathsf{AdvSSBr}_{\mathcal{A}}(\lambda)
\end{aligned}$$

The above game can be transformed to define semantic security for a public key broadcast encryption system if the attacker is not allowed to issue decryption queries.

Definition 3. *A public key broadcast encryption system is* (t, ϵ, l, m) *semantically secure if it is* $(t, \epsilon, l, m, 0)$*-CCA secure.*

In this paper we first provide a semantically secure scheme and then convert it to have CCA security.

4.3 Bilinear Groups and Pairings

We briefly review the necessary facts about bilinear maps and bilinear map groups. We use the following standard notation. [21, 5]

1. $\mathbb{G}$ and $\mathbb{G}_1$ are (multiplicative) cyclic groups of prime order p.
2. g is a generator of $\mathbb{G}$.
3. $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a bilinear map.

Let $\mathbb{G}$, and $\mathbb{G}_1$ be groups as above. A bilinear map is a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ with the following properties:

1. Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$
2. Non-degenerate: $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is bilinear group if the group action in $\mathbb{G}$ can be computed efficiently and there exist a group $\mathbb{G}_1$ and an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ as above.

4.4 Computational Complexity Assumptions

Security of our system is based on a complexity assumption called the bilinear Diffie-Hellman Exponent assumption (BDHE) used in [3, 6], which is a natural extension of bilinear-DHI assumption previously used in [2]. Let $\mathbb{G}$ be bilinear

group of prime order p. The l-BDHE problem in $\mathbb{G}$ are stated as follows: given a vector of $2l+1$ elements

$$(h, g, g^{\alpha}, g^{(\alpha^2)}, \ldots, g^{(\alpha^l)}, g^{(\alpha^{l+2})}, \ldots, g^{(\alpha^{2l})}) \in \mathbb{G}^{2l+1}$$

as input, output $e(h,g)^{\alpha^{l+1}} \in \mathbb{G}_1$. As shorthand, once g and α are specified, we use y_i to denote $y_i = g^{\alpha^i} \in \mathbb{G}$. An algorithm $\mathcal{A}$ has advantage ϵ in solving l-BDHE in $\mathbb{G}$ if

$$Pr[\mathcal{A}(h, g, y_1, \ldots, y_l, y_{l+2}, \ldots, y_{2l}) = e(h, y_{l+1})] \geq \epsilon$$

where the probability is over the random choice of generators $h, g \in \mathbb{G}$, the random choice of α in $\mathbb{Z}_p$, and the random bits used by $\mathcal{A}$. The decisional version of the l-BDHE problem is defined analogously. Let $\boldsymbol{y}_{g,\alpha,l} = (y_1, \ldots, y_l, y_{l+2}, \ldots, y_{2l})$. An algorithm $\mathcal{B}$ that outputs $b \in \{0,1\}$ has advantage ϵ in solving decisional l-BDHE if

$$|Pr[\mathcal{B}(h, g, \boldsymbol{y}_{g,\alpha,l}, e(\hat{g}, y_{l+1})) = 0] - Pr[\mathcal{B}(h, g, \boldsymbol{y}_{g,\alpha,l}, T) = 0]| \geq \epsilon$$

where the probability is over the random choice of generators $h, g \in \mathbb{G}$, the random choice of α in $\mathbb{Z}_p$, the random choice of $T \in \mathbb{G}_1$, and the random bits assumed by $\mathcal{B}$.

Definition 4. *We say that the (decisional) (t, ϵ, l)-BDHE assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the (decisional) l-BDHE problem in $\mathbb{G}$.*

Occasionally we omit the t and ϵ, and refer to the (decisional) l-BDHE in $\mathbb{G}$.

5 Proposed Broadcast Encryption Scheme

In this section, we propose a combinatorial subset difference broadcast encryption (CSD) scheme which provides a generalized subset difference representation allowing wildcards in any bit position.

5.1 Main Scheme

Setup(l,m): Let 2^l be the maximum number of users, and $\{0,1\}^m$ the message space. Note that it denotes an l-level public key broadcast encryption system . The generation of a random initial set of keys proceeds as follows. Select a random integer $\alpha, \beta \in \mathbb{Z}_p$, and $O(l)$ random group elements $g \in \mathbb{G}$, $g_2, g_3, h_{1,0}, h_{1,1}, \ldots, h_{l,0}, h_{l,1}, k_{1,0}, k_{1,1}, \ldots, k_{l,0}, k_{l,1} \in \mathbb{G}$, and compute $g_1 = g^{\alpha} \in \mathbb{G}$.

The public key is given by

$$PK \leftarrow (\hat{g}, \hat{g_1}, g, g_2, g_3, h_{1,0}, h_{1,1}, \ldots, h_{l,0}, h_{l,1}, k_{1,0}, k_{1,1}, \ldots, k_{l,0}, k_{l,1}).$$

A master secret key is defined as $master - key = g_2^{\alpha}$.

Next, to generate a private key SK_{ID} for an identity $ID = b_1 \cdots b_l$ using the master secret, pick random $r_1, \ldots, r_l \in \mathbb{Z}_p$ and output $SK_{ID} = (SK_{ID,1}, \ldots, SK_{ID,l})$ where

$$SK_{ID,i} = (\hat{g}^{r_i}, g_2^{\alpha}(h_{1,b_1} \cdots h_{l,b_l} k_{i,\bar{b}_i} g_3)^{r_i}, h_{1,\bar{b}_1}^{r_i}, \ldots, h_{l,\bar{b}_l}^{r_i}, \\ k_{1,0}^{r_i}, k_{1,1}^{r_i}, \ldots, k_{i-1,0}^{r_i}, k_{i-1,1}^{r_i}, k_{i,b_i}^{r_i}, k_{i+1,0}^{r_i}, k_{i+1,1}^{r_i}, \ldots, k_{l,0}^{r_i}, k_{l,1}^{r_i}) \in \mathbb{G}^{3l+1} \tag{1}$$

where $\bar{b}_i$ represents a bit NOT of b_i or $1 - b_i$.

Example : For a user of which $ID = 010$,

$$SK_{010} = (g^{r_1}, g_2^{\alpha}(h_{1,0}h_{2,1}h_{3,0}k_{1,1}g_3)^{r_1}, h_{1,1}^{r_1}, h_{2,0}^{r_1}, h_{3,1}^{r_1}, k_{1,0}^{r_1}, k_{2,0}^{r_1}, k_{2,1}^{r_1}, k_{3,0}^{r_1}, k_{3,1}^{r_1}, \\ g^{r_2}, g_2^{\alpha}(h_{1,0}h_{2,1}h_{3,0}k_{2,0}g_3)^{r_2}, h_{1,1}^{r_2}, h_{2,0}^{r_2}, h_{3,1}^{r_2}, k_{1,0}^{r_2}, k_{1,1}^{r_2}, k_{2,1}^{r_2}, k_{3,0}^{r_2}, k_{3,1}^{r_2}, \\ g^{r_3}, g_2^{\alpha}(h_{1,0}h_{2,1}h_{3,0}k_{3,1}g_3)^{r_3}, h_{1,1}^{r_3}, h_{2,0}^{r_3}, h_{3,1}^{r_3}, k_{1,0}^{r_3}, k_{1,1}^{r_3}, k_{2,0}^{r_3}, k_{2,1}^{r_3}, k_{3,0}^{r_3})$$

$\mathsf{Encrypt}(PK, S)$: Assume that subset difference sets are computed for the broadcast message. In the proposed combinatorial subset difference, a subset is represented as $S = (c, d)$, where c and d are l bit strings, or $c, d \in \{0, 1, *\}^l$. Label c indicates the covered nodes to include, and label d presents the nodes to be excluded. Using wildcards $(*)$, the label can denote the multiple nodes. For instance, for a subset $(**0*, 0*01)$, $**0*$ nodes except $0*01$ are included. Since label $**0*$ includes $0000, 0001, 0100, 0101, 1000, 1001, 1100$, and 1101, and label $0*01$ does 0001 and 0101, label $(**0*, 0*01)$ covers 0000, 0100, 1000, 1001, 1100, and 1101.

For each given $S = (c, d)$ where $c = c_1 \cdots c_l$, and $d_1 \cdots d_l$, to generate header Hdr and encryption key K, pick a random $s \in \mathbb{Z}_p$ and output

$$Hdr \leftarrow (g^s, (h_{1,c_1} \cdots h_{l,c_l} k_{1,d_1} \cdots k_{l,d_l} g_3)^s) \tag{2}$$

$$K \leftarrow e(g_1, g_2)^s \tag{3}$$

where $Hdr \in \mathbb{G}^2$ and $K \in \mathbb{G}_1$, $h_{i,*} = h_{i,0}h_{i,1}$, and $k_{i,*} = 1$.

Example : For the above subset $(**0*, 0*01)$,

$$Hdr \leftarrow (g^s, (h_{1,0}h_{1,1}h_{2,0}h_{2,1}h_{3,0}h_{4,0}h_{4,1}k_{1,0}k_{3,0}k_{4,1}g_3)^s)$$

$$K \leftarrow (g_1, g_2)^s$$

$\mathsf{Decrypt}(S, ID, SK_{ID}, Hdr)$: Consider an identity $ID = b_1 \cdots b_l$ and a subset $S = (c, d)$. Let j indicate an index such as $d_j = 0$ and $b_j = 1$, or $d_j = 1$ and $b_j = 0$. For instance, if $d = 0*01$ and $ID = 0000$ then $j = 4$ since $d_4 = 1$ and $b_4 = 0$.

Among $SK_{ID} = (SK_{ID,1}, \ldots, SK_{ID,l})$, we choose $SK_{ID,j}$ such that index j satisfies the above condition. To regenerate decrypt key K using the given header $Hdr = (A_0, A_1)$ and the private key $SK_{ID,j} = (a_0, a_1, h_{1,\bar{b}_1}^{r_j}, \ldots, h_{l,\bar{b}_l}^{r_j},$

$k_{1,0}^{r_j}, k_{1,1}^{r_j}, \ldots, k_{j-1,0}^{r_j}, k_{j-1,1}^{r_j}, k_{j,b_j}^{r_j}, k_{j+1,0}^{r_j}, k_{j+1,1}^{r_j}, \ldots, k_{l,0}^{r_j}, k_{l,1}^{r_j})$, compute $B = a_1 \cdot \prod_{i=1,c_i=*}^{l} h_{i,\bar{b}_i}^{r_j} \cdot \prod_{i=1,i\neq j,d_i\neq *}^{l} k_{i,d_i}^{r_j}$ and output

$$\frac{e(A_0, B)}{e(a_0, A_1)} = K \tag{4}$$

Indeed, for a valid ciphertext, we have

$$\begin{aligned}\frac{e(A_0, B)}{e(a_0, A_1)} &= \frac{e(g^s, g_2^\alpha (h_{1,c_1} \cdots h_{l,c_l} k_{1,d_1} \cdots k_{l,d_l} g_3)^{r_j})}{e(g^{r_j}, (h_{1,c_1} \cdots h_{l,c_l} k_{1,d_1} \cdots k_{l,d_l} g_3)^s)} \\ &= e(g, g_2)^{s\alpha} = e(g_1, g_2)^s\end{aligned}$$

Consider the above example with $ID = 0000$, $(c, d) = (**0*, 0*01)$, and $Hdr = (g^s, (h_{1,0}h_{1,1}h_{2,0}h_{2,1}h_{3,0}h_{4,0}h_{4,1}\ k_{1,0}k_{3,0}k_{4,1}g_3)^s)$. Since $b_4 = 0$, $d_4 = 1$, and $j = 4$, using $SK_{0000,4} = (g^{r_4}, g_2^\alpha (h_{1,0}h_{2,0}h_{3,0}h_{4,0}k_{4,1}g_3)^{r_4}, \ldots)$, we compute B.

$$\begin{aligned}B =& a_1 \cdot (h_{1,1}h_{2,1}h_{4,1})^{r_4} \cdot (k_{1,0}k_{3,0})^{r_4} \\ =& g_2^\alpha (h_{1,0}h_{2,0}h_{3,0}h_{4,0}k_{4,1}g_3)^{r_4} \cdot (h_{1,1}h_{2,1}h_{4,1})^{r_4} \cdot (k_{1,0}k_{3,0})^{r_4} \\ =& g_2^\alpha (h_{1,0}h_{1,1}h_{2,0}h_{2,1}h_{3,0}h_{4,0}h_{4,1}k_{1,0}k_{3,0}k_{4,1}g_3)^{r_4}.\end{aligned}$$

$$\begin{aligned}\frac{e(A_0, B)}{e(a_0, A_1)} &= \frac{e(g^s, g_2^\alpha (h_{1,0}h_{1,1}h_{2,0}h_{2,1}h_{3,0}h_{4,0}h_{4,1}k_{1,0}k_{3,0}k_{4,1}g_3)^{r_4})}{e(g^{r_4}, (h_{1,0}h_{1,1}h_{2,0}h_{2,1}h_{3,0}h_{4,0}h_{4,1}k_{1,0}k_{3,0}k_{4,1}g_3)^s)} \\ &= \frac{e(g, g_2)^{s\alpha} e(g, h_{1,0}h_{1,1}h_{2,0}h_{2,1}h_{3,0}h_{4,0}h_{4,1}k_{1,0}k_{3,0}k_{4,1}g_3)^{sr_4}}{e(g, h_{1,0}h_{1,1}h_{2,0}h_{2,1}h_{3,0}h_{4,0}h_{4,1}k_{1,0}k_{3,0}k_{4,1}g_3)^{sr_4}} \\ &= e(g, g_2)^{s\alpha}.\end{aligned}$$

Consider an $ID = 0010$ which is not included by an inclusion label $**0*$ in subset $(**0*, 0*01)$. Since $SK_{0010,i} = (g^{r_i}, g_2^\alpha (h_{1,0}h_{2,0}h_{3,1}h_{4,0}k_{i,\bar{b}_i})^{r_i}, \ldots)$ and $A_1 = (h_{1,0}h_{1,1}h_{2,0}h_{2,1}\ \mathbf{h_{3,0}}h_{4,0}h_{4,1}\cdots)^s$, it is necessary to eliminate $h_{3,1}^{r_i}$ from $SK_{0010,i}$, which is not possible. Hence, it is impossible to compute B using $SK_{0010,i}$.

Examine an $ID = 0001$ which is included by an exclusion label $0*01$ in subset $(**0*, 0*01)$. Since $SK_{0001,1} = (g^{r_1}, g_2^\alpha(\cdots k_{1,1})^{r_1}, \ldots)$, $SK_{0001,2} = (g^{r_2}, g_2^\alpha(\cdots k_{2,1})^{r_2}, \ldots)$, $SK_{0001,3} = (g^{r_3}, g_2^\alpha(\cdots k_{3,1})^{r_3}, \ldots)$, $SK_{0001,4} = (g^{r_4}, g_2^\alpha(\cdots k_{4,0})^{r_4}, \ldots)$, and $A_1 = (\cdots k_{1,0}k_{3,0}k_{4,1})^s$, it is necessary to eliminate $k_{1,1}^{r_1}$ from $SK_{0001,1}$, $k_{2,1}^{r_2}$ from $SK_{0001,2}$, $k_{3,1}^{r_3}$ from $SK_{0001,3}$, or $k_{4,0}^{r_4}$ from $SK_{0001,4}$, which is impossible. Hence, it is impossible to compute B from $SK_{0001,i}$.

Note that Encrypt allows multiple subsets. For each subset S, Encrypt is applied so that the number of Hdr's is the number of subsets. When a user receives a message, it chooses Hdr to which the user belongs and Decrypt is called for the Hdr.

5.2 General ID Scheme

Next, we generalize the proposed broadcast encryption to allow each ID to include $*$ as well as 0 and 1. This general ID scheme becomes a basic building block to build a CCA secure broadcast encryption scheme in section 6.

For labels x and y, we define $x \preceq y$ to indicate that x is covered by y, and $x \npreceq y$ to denote that x is not covered by y. For instance, $0*00 \preceq 0**0$ and $0*00 \npreceq 1**0$.

Definition 5. *x is covered by y, or $x \preceq y$ iff $\forall i$, $x_i = y_i$ or $y_i = *$ for $x = x_1 \dots x_l$ and $y = y_1 \dots y_l$.*

Definition 6. *x is not covered by y, or $x \npreceq y$ iff $\forall a \preceq x$, $\exists i$, $a_i \neq y_i$, $x_i \neq *$, and $y_i \neq *$.*

Setup(l,m): The setup is similar to the main scheme. The public key generation is equivalent and the private key generation is similar to equation 1, except that $h_{i,*} = 1$ and $h_{i,\bar{*}}^{r_j}$ populates two values of $h_{i,0}^{r_j}$ and $h_{i,1}^{r_j}$. Similarly, since interpretation of $k_{i,*}$ covers both $k_{i,0}$ and $k_{i,1}$, the secret key $SK_{ID,i}$ should be doubled into $SK_{ID,i,0}$ and $SK_{ID,i,1}$ if $b_i = *$. The resulting secret key for ID is $SK_{ID} = (SK_{ID,1}, \dots, SK_{ID,l})$ where if $b_i \neq *$ then i-th secret key is the same as the key generation in the main scheme:

$$SK_{ID,i} = (g^{r_i}, g_2^{\alpha}(h_{1,b_1} \cdots h_{l,b_l} k_{i,\bar{b}_i} g_3)^{r_i}, h_{1,\bar{b}_1}^{r_i}, \dots, h_{l,\bar{b}_l}^{r_i}, \\ k_{1,0}^{r_i}, k_{1,1}^{r_i}, \dots, k_{i-1,0}^{r_i}, k_{i-1,1}^{r_i}, k_{i,b_i}^{r_i}, k_{i+1,0}^{r_i}, k_{i+1,1}^{r_i}, \dots, k_{l,0}^{r_i}, k_{l,1}^{r_i}) \in \mathbb{G}^{3l+1}$$

else if $b_i = *$ then i-th secret key duplicated into two elements as follows: $SK_{ID,i} = (SK_{ID,i,0}, SK_{ID,i,1})$ with

$$\begin{aligned} SK_{ID,i,0} &= (g^{r_{i,0}}, g_2^{\alpha}(h_{1,b_1} \cdots h_{i-1,b_{i-1}} h_{i+1,b_{i+1}} \cdots h_{l,b_l} k_{i,0} g_3)^{r_{i,0}}, \\ &\quad h_{1,\bar{b}_1}^{r_{i,0}}, \dots, h_{i,0}^{r_{i,0}}, h_{i,1}^{r_{i,0}}, \dots, h_{l,\bar{b}_l}^{r_{i,0}}, \\ &\quad k_{1,0}^{r_{i,0}}, k_{1,1}^{r_{i,0}}, \dots, k_{i-1,0}^{r_{i,0}}, k_{i-1,1}^{r_{i,0}}, k_{i,1}^{r_{i,0}}, k_{i+1,0}^{r_{i,0}}, k_{i+1,1}^{r_{i,0}}, \dots, k_{l,0}^{r_{i,0}}, k_{l,1}^{r_{i,0}}) \\ SK_{ID,i,1} &= (g^{r_{i,1}}, g_2^{\alpha}(h_{1,b_1} \cdots h_{i-1,b_{i-1}} h_{i+1,b_{i+1}} \cdots h_{l,b_l} k_{i,1} g_3)^{r_{i,1}}, \\ &\quad h_{1,\bar{b}_1}^{r_{i,1}}, \dots, h_{i,0}^{r_{i,1}}, h_{i,1}^{r_{i,1}}, \dots, h_{l,\bar{b}_l}^{r_{i,1}}, \\ &\quad k_{1,1}^{r_{i,1}}, k_{1,1}^{r_{i,1}}, \dots, k_{i-1,0}^{r_{i,1}}, k_{i-1,1}^{r_{i,1}}, k_{i,0}^{r_{i,1}}, k_{i+1,0}^{r_{i,1}}, k_{i+1,1}^{r_{i,1}}, \dots, k_{l,0}^{r_{i,1}}, k_{l,1}^{r_{i,1}}) \end{aligned} \tag{5}$$

Encrypt(PK,S): Equivalent to equations 2 and 3.

Decrypt(S,ID,SK_{ID},Hdr): Consider an identity $ID = b_1 \dots b_l \in \{0,1,*\}^l$ and a subset $S = (c, d)$. If $ID \preceq c$ and $ID \npreceq d$ then ID can decrypt the message. Let j be an index such that $b_j = *$ and $d_j \neq *$, $b_j = 0$ and $d_j = 1$, or $b_j = 1$ and $d_j = 0$. If $b_j \neq *$ then the decryption is equivalent to equation 4. Assume that $b_j = *$. From $SK_{ID,j}$, we use SK_{ID,j,d_j} as private key. To regenerate decrypt key K using the given header $Hdr = (A_0, A_1)$ and the private key SK_{ID,j,d_j},

compute $B = a_1 \cdot \prod_{i=1,c_i=*,b_i\neq *}^{l} h_{i,\bar{b}_i}^{r_j} \prod_{i=1,c_i\neq *,b_i=*}^{l} h_{i,c_i}^{r_j}$
$\prod_{i=1,c_i=*,b_i=*}^{l} h_{i,0}^{r_j} h_{i,1}^{r_j} \cdot \prod_{i=1,i\neq j,d_i\neq *}^{l} k_{i,d_i}^{r_j}$ and output

$$\frac{e(A_0, B)}{e(a_0, A_1)} = K$$

5.3 Security Proof

We prove the security of our scheme under the decisional l-BDHE assumption without the random oracle model.

Theorem 3. *Let $\mathbb{G}$ be a bilinear group of prime order p. Suppose the (decision) $(t, \epsilon, 4l)$-BDHE assumption holds in $\mathbb{G}$. Then our l-level CSD public key broadcast encryption system is (t', ϵ, l, m) semantically secure for arbitrary l, and $t' < t - O(\boldsymbol{e}l^2 2^l)$, where $\boldsymbol{e}$ is the maximum time for an exponentiation in $\mathbb{G}$.*

Proof. Suppose $\mathcal{A}$ has advantage ϵ in attacking the l-level CSD public key broadcast encryption system . Using $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that solves the (decision) $4l$-BDHE problem in $\mathbb{G}$.

For generators $h, g \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p^*$, let $y_i = g^{\alpha^i} \in \mathbb{G}$. Algorithm $\mathcal{B}$ is given as input a random tuple $(h, g, y_1, \ldots, y_{4l}, y_{4l+2}, \ldots, y_{8l}, T)$ that is either sampled from P_{BDHE} (where $T = e(h, g)^{(\alpha^{4l+1})}$) or from R_{BDHE} (where T is uniform and independent in $\mathbb{G}_1$). Algorithm $\mathcal{B}$'s goal is to output 1 when the input tuple is sampled from P_{BDHE} and 0 otherwise. Algorithm $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a selective subset game as follows:

Init: The game begins with $\mathcal{A}$ first outputting a subset $S^* = (c^*, d^*)$ that it intends to attack where c^*, $d^* \in \{0, 1, *\}^l$.

Setup: To generate the public key, algorithm $\mathcal{B}$ picks a random γ in $\mathbb{Z}_p$ and sets $g_1 = y_1 = g'^{\alpha}$ and $g_2 = y_{4l} g^{\gamma} = g^{\gamma + (\alpha^{4l})}$. Next, $\mathcal{B}$ picks random $\gamma_{1,0}, \gamma_{1,1}, \cdots, \gamma_{l,0}, \gamma_{l,1}$ and $\psi_{1,0}, \psi_{1,1}, \cdots, \psi_{l,0}, \psi_{l,1}$ in $\mathbb{Z}_p$, and sets $h_{i,0} = g^{\gamma_{i,0}} / y_{l-i+1}, h_{i,1} = g^{\gamma_{i,1}} / y_{2l-i+1}$, $k_{i,0} = g^{\psi_{i,0}} / y_{3l-i+1}, k_{i,1} = g^{\psi_{i,1}} / y_{4l-i+1}$ for $i = 1, \ldots, l$. Algorithm $\mathcal{B}$ also picks a random δ in $\mathbb{Z}_p$ and sets $g_3 = g^{\delta} \prod_{i=1}^{l} y_{l-i+1}^{c^*_{i,0}} y_{2l-i+1}^{c^*_{i,1}} y_{3l-i+1}^{d^*_{i,0}} y_{4l-i+1}^{d^*_{i,1}}$ such that $c^*_{i,0} = 0$ and $c^*_{i,1} = 1$ if $c^*_i = 1$, $c^*_{i,0} = 1$ and $c^*_{i,1} = 0$ if $c^*_i = 0$, and $c^*_{i,0} = 1$ and $c^*_{i,1} = 1$ if $c^*_i = *$. $d^*_{i,0}$ and $d^*_{i,1}$ are similarly defined by d^*_i except that $d^*_{i,0} = 0$ and $d^*_{i,1} = 0$ if $d^*_i = *$.

Consider a query for the private key corresponding to $ID = b_1 \ldots b_l \in \{0, 1\}^l$. We now show that we can simulate revoked users with following two cases. Case 1 shows process of simulating users that are not included in an inclusion label c^* in subset (c^*, d^*). In case 2, we show that we can also simulate users that are included in an exclusion label d^*.

Case 1: If $ID \not\preceq c^*$ there exists $k \in \{1, \ldots, l\}$ such that $b_k \neq c^*_k$, $b_k \neq *$, and $c^*_k \neq *$. We set k such that it is the smallest such index. To generate the private key $SK_{ID,j}$, B first picks random $\tilde{r}_1, \ldots, \tilde{r}_l$ in $\mathbb{Z}_p$. We pose $r_j = \alpha^{(3-b_k)l+k} + \tilde{r}_j$

for $j = 1, \ldots, l$. Next, $\mathcal{B}$ generates the private key $SK_{ID} = (SK_{ID,1}, \ldots, SK_{ID,l})$ where

$$SK_{ID,j} = (g^{r_j}, g_2^{\alpha}(h_{1,b_1} \cdots h_{l,b_l} k_{j,\bar{b}_j} g_3)^{r_j}, h_{1,\bar{b}_1}^{r_j}, \ldots, h_{l,\bar{b}_l}^{r_j}, \\ k_{1,0}^{r_j}, k_{1,1}^{r_j}, \ldots, k_{j-1,0}^{r_j}, k_{j-1,1}^{r_j}, k_{j,b_j}^{r_j}, k_{j+1,0}^{r_j}, k_{j+1,1}^{r_j}, \ldots, k_{l,0}^{r_j}, k_{l,1}^{r_j})$$

which is a properly distributed private key for the identity $ID = b_1 \ldots b_l$. We show that $\mathcal{B}$ can compute all elements of this private key given the values at its disposal. We use the fact that $y_i^{\alpha^j} = y_{i+j}$ for any i, j. Note that $b_{i,0} = \bar{b}_i$ and $b_{i,1} = b_i$ if $b_i \neq *$; otherwise, $b_{i,0} = b_{i,1} = 1$. If $b_j = *$ and an equation includes $\bar{b}_j$ then two equations are created by replacing $\bar{b}_j$ by 0 and 1. To generate the second component of the private key, first observe that

$$(h_{1,b_1} \cdots h_{l,b_l} k_{j,\bar{b}_j} g_3)^{r_j} = (\prod_{i=1}^{l} h_{i,0}^{b_{i,0}} h_{i,1}^{b_{i,1}} \cdot k_{j,\bar{b}_j} \cdot g_3)^{r_j}$$

$$= (\prod_{i=1}^{l} (\frac{g^{\gamma_{i,0}}}{y_{l-i+1}})^{b_{i,0}} (\frac{g^{\gamma_{i,1}}}{y_{2l-i+1}})^{b_{i,1}} \cdot \frac{g^{\psi_{j,\bar{b}_j}}}{y_{(3+\bar{b}_j)l-j+1}} \cdot g^{\delta} \prod_{i=1}^{l} y_{l-i+1}^{c^*_{i,0}} y_{2l-i+1}^{c^*_{i,1}} y_{3l-i+1}^{d^*_{i,0}} y_{4l-i+1}^{d^*_{i,1}})^{r_j}$$

$$= (g^{\delta+\psi_{j,\bar{b}_j}+\sum_{i=1,i\neq k}^{l}(b_{i,0}\gamma_{i,0}+b_{i,1}\gamma_{i,1})} \prod_{i=1,i\neq k}^{l} (y_{l-i+1}^{c^*_{i,0}-b_{i,0}} y_{2l-i+1}^{c^*_{i,1}-b_{i,1}}) \cdot$$

$$\prod_{i=1}^{l} (y_{3l-i+1}^{d^*_{i,0}} y_{4l-i+1}^{d^*_{i,1}}) y_{3l-j+1}^{-b_{j,0}} y_{4l-j+1}^{-b_{j,1}} y_{l-k+1}^{c^*_{k,0}-b_{k,0}} y_{2l-k+1}^{c^*_{k,1}-b_{k,1}})^{r_j}$$

$$= (g^{\delta+\psi_{j,\bar{b}_j}+\sum_{i=1,i\neq k}^{l}(b_{i,0}\gamma_{i,0}+b_{i,1}\gamma_{i,1})} \prod_{i=1,i\neq k}^{l} (y_{l-i+1}^{c^*_{i,0}-b_{i,0}} y_{2l-i+1}^{c^*_{i,1}-b_{i,1}}) \cdot$$

$$\prod_{i=1}^{l} (y_{3l-i+1}^{d^*_{i,0}} y_{4l-i+1}^{d^*_{i,1}}) y_{3l-j+1}^{-b_{j,0}} y_{4l-j+1}^{-b_{j,1}} y_{(1+\bar{b}_k)l-k+1}^{c^*_{k,\bar{b}_k}} y_{(1+b_k)l-k+1}^{-1})^{r_j}$$

Let Z^{r_j} denote the product of y except the last one. That is

$$Z^{r_j} = (g^{\delta+\psi_{j,\bar{b}_j}+\sum_{i=1,i\neq k}^{l}(b_{i,0}\gamma_{i,0}+b_{i,1}\gamma_{i,1})} \prod_{i=1,i\neq k}^{l} (y_{l-i+1}^{c^*_{i,0}-b_{i,0}} y_{2l-i+1}^{c^*_{i,1}-b_{i,1}}) \cdot$$

$$\prod_{i=1}^{l} (y_{3l-i+1}^{d^*_{i,0}} y_{4l-i+1}^{d^*_{i,1}}) y_{3l-j+1}^{-b_{j,0}} y_{4l-j+1}^{-b_{j,1}} y_{(1+\bar{b}_k)l-k+1}^{c^*_{k,\bar{b}_k}})^{r_j}$$

$$= y_{(3-b_k)l+k}^{\delta+\psi_{j,\bar{b}_j}+\sum_{i=1,i\neq k}^{l}(b_{i,0}\gamma_{i,0}+b_{i,1}\gamma_{i,1})} \cdot \prod_{i=1,i\neq k}^{l} (y_{(4-b_k)l-i+k+1}^{c^*_{i,0}-b_{i,0}} y_{(5-b_k)l-i+k+1}^{c^*_{i,1}-b_{i,1}}) \cdot$$

$$\prod_{i=1}^{l} (y_{(6-b_k)l-i+k+1}^{d^*_{i,0}} y_{(7-b_k)l-i+k+1}^{d^*_{i,1}}) \cdot y_{(6-b_k)l-j+1}^{-b_{j,0}} y_{(7-b_k)l-j+1}^{-b_{j,1}} y_{(4-b_k+\bar{b}_k)l+1}^{c^*_{k,\bar{b}_k}} Z^{\bar{r}_j}$$

Since $(3-b_k)l+k < 4l$, $(4-b_k)l-i+k+1 = (4-b_k)l+1+(k-i) \neq 4l+1$ and $(5-b_k)l-i+k+1 \neq 4l+1$ due to $k \neq i$, $(7-b_k)l-i+k+1 > (6-b_k)l-i+k+1 \geq$

$4l+k+1 > 4l+1$, $(7-b_k)l-j+k+1 > (6-b_k)l-j+k+1 \geq 5l-l+k+1 > 4l+1$, $(4-b_k+\bar{b}_k)l+1 = 3l+1$ or $5l+1$, and Z is computable, $\mathcal{B}$ can compute all the terms in Z^{r_j} given the values at its disposal. Next observe that the last term, namely $y_{(1+b_k)l-k+1}^{-r_j}$, is:

$$y_{(1+b_k)l-k+1}^{-r_j} = y_{(1+b_k)l-k+1}^{-\tilde{r}_j} y_{(1+b_k)l-k+1}^{-\alpha^{(3-b_k)l+k}} = y_{(1+b_k)l-k+1}^{-\tilde{r}_j} y_{4l+1}^{-1}$$

Hence, the first component in the private key is equal to:

$$g_2^{\alpha}(h_{1,b_1}\cdots h_{l,b_l}k_{j,\bar{b}_j}g_3)^{r_j} = (y_{4l+1}y_1^{\gamma})Z^{r_j}(y_{(1+b_k)l-k+1}^{-\tilde{r}_j}y_{4l+1}^{-1}) = y_1^{\gamma}y_{(1+b_k)l-k+1}^{-\tilde{r}_j}Z^{r_j}$$

Since y_{4l+1} cancels out, all the terms in this expression are known to $\mathcal{B}$. Thus, $\mathcal{B}$ can compute the first private key component. The first component, g^{r_j}, is $y_{(3-b_k)l+k}g^{\tilde{r}_j}$ which $\mathcal{B}$ can compute. Similarly, the remaining elements $h_{1,\bar{b}_1}^{r_j},\cdots,h_{l,\bar{b}_l}^{r_j}, k_{1,0}^{r_j}, k_{1,1}^{r_j},\cdots,k_{j-1,0}^{r_j}, k_{j-1,1}^{r_j}k_{j,b_j}^{r_j}, k_{j+1,0}^{r_j}$, $k_{j+1,1}^{r_j},\cdots,k_{l,0}^{r_j}, k_{l,1}^{r_j}$ can be computed by $\mathcal{B}$ since they do not involve a y_{4l+1} term such that $h_{i,\bar{b}_i}^{r_j} = (\frac{g^{\gamma_{i,\bar{b}_i}}}{y_{(1+\bar{b}_i)l-i+1}})^{r_j} = (y_{(3-b_k)l+k}\cdot g^{\tilde{r}_j})^{\gamma_{i,\bar{b}_i}}\cdot$ $(y_{(4-b_k+\bar{b}_i)l-i+k+1}y_{(1+\bar{b}_i)l-i+1}^{\tilde{r}_j})^{-1}$ and $k_{i,t}^{r_j} = (\frac{g^{\psi_{i,t}}}{y_{(3+t)l-i+1}})^{r_j} = (y_{(3-b_k)l+k}\cdot g^{\tilde{r}_j})^{\psi_{i,t}}\cdot$ $(y_{(6-b_k+t)l-i+k+1}\cdot y_{(3+t)l-i+1}^{\tilde{r}_j})^{-1}$. Thus, $\mathcal{B}$ can derive $h_{i,\bar{b}_i}^{r_j}$ since $(4-b_k+\bar{b}_i)l-i+k+1 \neq (4-b_k+\bar{b}_i)l+1$ when $i \neq k$ and $(4-b_k+\bar{b}_i)l-i+k+1 = 3l+1$ or $5l+1$ when $i=k$. Moreover, $\mathcal{B}$ can derive $k_{i,t}^{r_j}$ since $(3-b_k)l+k \leq 4l$, $(6-b_k+t)l-i+k+1 \geq 5l-i+k+1 > 4l+1$, and $(3+t)l-i+1 \leq 4l$.

Case 2: If $ID \preceq d^*$, to generate the private key $SK_{ID,j}$ for identity $ID = b_1\ldots b_l$, $\mathcal{B}$ first picks a random $\tilde{r}_1,\cdots,\tilde{r}_l$ in $\mathbb{Z}_p$. We pose $r_j = \alpha^{(1-\bar{b}_j)l+j}+\tilde{r}_j$ for $j=1,\cdots,l$. Next, $\mathcal{B}$ generates the private key $SK_{ID} = (SK_{ID,1},\cdots,SK_{ID,l})$ The private key parameters are computed similarly. To generate the second component of the private key, first observe that

$$(h_{1,b_1}\cdots h_{l,b_l}k_{j,\bar{b}_j}g_3)^{r_j} =$$
$$(g^{\delta+\psi_{j,\bar{b}_j}+\sum_{i=1}^{l}(b_{i,0}\gamma_{i,0}+b_{i,1}\gamma_{i,1})}\prod_{i=1}^{l}(y_{l-i+1}^{c_{i,0}^*-b_{i,0}}y_{2l-i+1}^{c_{i,1}^*-b_{i,1}})$$
$$\prod_{i=1,i\neq j}^{l}(y_{3l-i+1}^{d_{i,0}^*}y_{4l-i+1}^{d_{i,1}^*})\cdot y_{(3+b_j)l-j+1}^{d_{j,b_j}^*}\cdot y_{(3+\bar{b}_j)l-j+1}^{-1})^{r_j}$$

Let Z^{r_j} denote the product of y except the last one. That is

$$Z^{r_j} = (g^{\delta+\psi_{j,\bar{b}_j}+\sum_{i=1}^{l}(b_{i,0}\gamma_{i,0}+b_{i,1}\gamma_{i,1})}\prod_{i=1}^{l}(y_{l-i+1}^{c_{i,0}^*-b_{i,0}}y_{2l-i+1}^{c_{i,1}^*-b_{i,1}})$$
$$\prod_{i=1,i\neq j}^{l}(y_{3l-i+1}^{d_{i,0}^*}y_{4l-i+1}^{d_{i,1}^*})\cdot y_{(3+b_j)l-j+1}^{d_{j,b_j}^*})^{r_j}$$

$\mathcal{B}$ can compute all the terms in Z^{r_j} given the values at its disposal since $l-i+1+(1-\bar{b}_j)l+j \leq (2-\bar{b}_j)l-i+j+1 \leq 3l$, $2l-i+1+(1-\bar{b}_j)l+j \leq 4l$, $3l-i+1+(1-\bar{b}_j)l+j = (4-\bar{b}_j)l-i+j+1 \neq 4l+1$, $4l-i+1+(1-\bar{b}_j)l+j = (5-\bar{b}_j)l-i+j+1 \neq 4l+1$ due to $i \neq j$, and $(3+b_j)l-j+1+(1-\bar{b}_j)l+j = (4+b_j-\bar{b}_j)l+1 = 3l+1$ or $5l+1$. Note that if $b_j = *$ then $d^*_{j,0} = d^*_{j,1} = 0$, and each case that $b_j = 0$ or $b_j = 1$ is considered. Next observe that the last term, namely $y^{-r_j}_{(3+\bar{b}_j)l-j+1}$, is:

$$y^{-r_j}_{(3+\bar{b}_j)l-j+1} = y^{-\tilde{r}_j}_{(3+\bar{b}_j)l-j+1} y^{-\alpha^{(1-\bar{b}_j)l+j}}_{(3+\bar{b}_j)l-j+1} = y^{-\tilde{r}_j}_{(3+\bar{b}_j)l-j+1} \cdot y^{-1}_{4l+1}$$

Hence, the first component in the private key is equal to:

$$g_2^{\alpha}(h_{1,b_1}\cdots h_{l,b_l}k_{j,\bar{b}_j}g_3)^{r_j} =$$
$$(y_{4l+1}y_1^{\gamma})Z^{r_j}(y^{-\tilde{r}_j}_{(3+\bar{b}_j)l-j+1}/y_{4l+1}) = y_1^{\gamma}y^{-\tilde{r}_j}_{(3+\bar{b}_j)l-j+1}Z^{r_j}$$

Since y_{4l+1} cancels out, all the terms in this expression are known to $\mathcal{B}$. Thus, $\mathcal{B}$ can compute the first private key component. The first component, g^{r_j}, is $y_{(1-\bar{b}_j)l+j}g^{\tilde{r}_j}$ which $\mathcal{B}$ can compute. Similarly, the remaining elements $h^{r_j}_{1,\bar{b}_1}, \cdots, h^{r_j}_{l,\bar{b}_l}, k^{r_j}_{1,0}, k^{r_j}_{1,1}, \ldots, k^{r_j}_{j-1,0}, k^{r_j}_{j-1,1}, k^{r_j}_{j,b_j}, k^{r_j}_{j+1,0}, k^{r_j}_{j+1,1}, \ldots, k^{r_j}_{l,0}, k^{r_j}_{l,1}$ can be computed by $\mathcal{B}$ since they do not involve a y_{4l+1} term since $h^{r_j}_{i,\bar{b}_i} = (\frac{g^{\gamma_{i,\bar{b}_i}}}{y_{(1+\bar{b}_i)l-i+1}})^{r_j} = (y_{(1-\bar{b}_j)l+j}\cdot g^{\tilde{r}_j})^{\gamma_{i,\bar{b}_i}} \cdot (y_{(2+\bar{b}_i-\bar{b}_j)l-i+j+1}y^{\tilde{r}_j}_{(1+\bar{b}_i)l-i+1})^{-1}$ and $k^{r_j}_{i,t} = (\frac{g^{\psi_{i,t}}}{y_{(3+t)l-i+1}})^{r_j} = (y_{(1-\bar{b}_j)l+j}\cdot g^{\tilde{r}_j})^{\psi_{i,t}} \cdot (y_{(4+t-\bar{b}_j)l-i+j+1}\cdot y^{\tilde{r}_j}_{(3+t)l-i+1})^{-1}$. Thus, $\mathcal{B}$ can derive a valid private key for ID since $(4+t-\bar{b}_j)l-i+j+1 \neq (4+t-\bar{b}_j)l+1$ when $i \neq j$ and $(4-t+\bar{b}_j)l-i+j+1 = 3l+1$ or $5l+1$ due to $t = b_j$ when $i = j$.

Thus, $\mathcal{B}$ can derive a valid private key for ID. Finally, $\mathcal{B}$ gives $\mathcal{A}$ public key $PK = (g, g_1, g_2, g_3, h_{1,0}, h_{1,1}, \cdots, h_{l,0}, h_{l,1}, k_{1,0}, k_{1,1}, \cdots, k_{l,0}, k_{l,1})$ and SK_{ID} such that $ID \not\preceq c^*$ or $ID \preceq d^*$. Observe that all these values are distributed uniformly and independently in $\mathbb{G}$ as required. The master key corresponding to these system parameters is $g_2^{\alpha} = g^{\alpha(\alpha^{4l}+\gamma)} = y_{4l+1}y_1^{\gamma}$, which is unknown to $\mathcal{B}$ since $\mathcal{B}$ does not have y_{4l+1}.

To generate the challenge, $\mathcal{B}$ computes Hdr^* as $(h,$ $h^{\delta+\sum_{i=1}^{l}(\gamma_{i,0}c^*_{i,0}+\gamma_{i,1}c^*_{i,1}+\psi_{i,0}d^*_{i,0}+\psi_{i,1}d^*_{i,1})})$. It sets $K^* = T \cdot e(y_1, h^{\gamma})$ and gives (Hdr^*, K^*) as the challenge to $\mathcal{A}$. We claim that when $T = e(g,h)^{4l+1}$(i.e. the input to $\mathcal{B}$ is a $4l$-BDHE tuple) then (Hdr^*, K^*) is a valid challenge to A as in real attack. To see this, write $h = g^c$ for some (unknown) $c \in \mathbb{Z}_p$. Then

$$
\begin{aligned}
&h^{\delta+\sum_{i=1}^{l}(\gamma_{i,0}c^*_{i,0}+\gamma_{i,1}c^*_{i,1}+\psi_{i,0}d^*_{i,0}+\psi_{i,1}d^*_{i,1})} \\
&= (\prod_{i=1}^{l}(\frac{g^{\gamma_{i,0}}}{y_{l-i+1}})^{c^*_{i,0}}(\frac{g^{\gamma_{i,1}}}{y_{2l-i+1}})^{c^*_{i,1}}(\frac{g^{\psi_{i,0}}}{y_{3l-i+1}})^{d^*_{i,0}}(\frac{g^{\psi_{i,1}}}{y_{4l-i+1}})^{d^*_{i,1}}\cdot \\
&(g^{\delta}\prod_{i=1}^{l} y_{l-i+1}^{c^*_{i,0}} y_{2l-i+1}^{c^*_{i,1}} y_{3l-i+1}^{d^*_{i,0}} y_{4l-i+1}^{d^*_{i,1}}))^{c} \\
&= (h_{1,c^*_1}\cdots h_{l,c^*_l} k_{1,d^*_1}\cdots k_{l,d^*_l} g_3)^{c}
\end{aligned}
$$

and

$$
e(g,h)^{(\alpha^{4l+1})}\cdot e(y_1,h^{\gamma}) = (e(y_1,y_{4l})\cdot e(y_1,g^{\gamma}))^{c} = e(y_1,y_{4l}g^{\gamma})^{c} = e(g_1,g_2)^{c}.
$$

Therefore, by definition, Hdr^* is a valid encryption of the key $e(y_{4l+1},g)^c$. Furthermore, $e(y_{4l+1},g)^c = e(g,h)^{4l+1} = T = K_b$ and hence (Hdr, K^*) is a valid challenge to $\mathcal{A}$. On the other hand, when T is random in $\mathbb{G}$(i.e. the input to $\mathcal{B}$ is a random tuple) then K^* are just random independent key of $\mathcal{K}$ in the $\mathcal{A}$'s view.

Guess:

Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. Algorithm $\mathcal{B}$ concludes its own game by outputting the b'. If $b' = 1$ then $\mathcal{B}$ outputs 1 meaning $T = e(g,h)^{(\alpha^{4l+1})}$. Otherwise, it outputs 0 meaning T is random in $\mathbb{G}_1$.

When the input tuple is sampled from P_{BDHE} (where $T = e(g,h)^{(\alpha^{4l+1})}$) then $\mathcal{A}$'s view is identical to its view in a real attack game and therefore $\mathcal{A}$ satisfies $|Pr[b'=1] - 1/2| \geq \epsilon$. When the input tuple is sampled from R_{BDHE} (where T is uniform in $\mathbb{G}_1$) then $Pr[b'=1] = 1/2$. Therefore, with g, h uniform in $\mathbb{G}$, α uniform in $\mathbb{Z}_p$, and T uniform in $\mathbb{G}_1$ we have that

$$
\begin{aligned}
&|Pr[B(g,h,\boldsymbol{y}_{g,\alpha,4l}, e(g,h)^{(\alpha^{4l+1})}) = 0] \\
&\quad - Pr[B(g,h,\boldsymbol{y}_{g,\alpha,4l}, T) = 0]| \geq |(1/2+\epsilon) - 1/2| = \epsilon
\end{aligned} \tag{6}
$$

as required, which completes the proof of the theorem.

6 Chosen Ciphertext Secure Broadcast Encryption

In the following description, a vector $V = (v_1, \cdots, v_n)$ is interchangeably represented as $v_1 \ldots v_n$. With two vectors $V = (v_1, \cdots, v_n)$ and $V' = (v'_1, \cdots, v'_m)$, we denote $V||V' = (v_1, \cdots, v_n, v'_1, \cdots, v'_m)$.

We extend our semantically secure broadcast encryption scheme using the similar technique in [4] to attain chosen ciphertext security. Given a strong one-time signature scheme $(SigKeygen, Sign, Verify)$ with verification keys which are mapped to $\{0,1\}^z$, we enable construction of an l-level public key broadcast encryption system $\Pi = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ secure against chosen-ciphertext attacks using the $(l+z)$-level $\Pi' = (\mathsf{Setup}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ semantically secure broadcast encryption scheme. The intuition is that $ID =$

$(b_1, \cdots, b_l) \in \{1, 0, *\}^l$ in Π is mapped to $ID' = ID||*^z = (b_1, \cdots, b_l, *, \cdots, *) \in \{1, 0, *\}^{l+z}$ in Π'. Thus, the secret key SK_{ID} for ID in Π is the secret key $SK_{ID'}$ in Π'. Recall that $SK'_{ID||*^z}$ can generate secret keys of all descendants of node $ID||*^z$, i.e., $SK'_{ID||0^z}, \cdots, SK'_{ID||1^z}$. When encrypting a key $K \in \mathcal{K}$ to ID in Π, the sender generates a z-bit verification key $V_{sig} = (e_1, \cdots, e_z) \in \{0,1\}^z$ and then encrypts K to the $ID' = ID||V_{sig}$ using Π'.

In more details, l-level Π is constructed using $(l+z)$-level Π' and an one-time signature scheme as follows:

$\mathsf{Setup}(l, m)$: Let 2^l be the maximum number of users and $\{0,1\}^m$ be the message space. Assume that the signature verification key space is $\{0,1\}^z$. Run semantically secure broadcast encryption scheme Π' to obtain the public key and master secret key.

$$PK, master-key, \{SK'_{ID'}\}_{ID' \in \{0,1\}^{l+z}} \leftarrow \mathsf{Setup}'(l+z)$$

To generate private key SK_{ID} for an identity $ID = b_1 \dots b_l$ using the master secret, encode ID to $ID' = ID||\underbrace{* * \cdots *}_{z}$. The secret key $SK'_{ID'}$ is generated from the key generation algorithm in Setup' of Π'. Let $SK_{ID} = SK'_{ID'} = (SK'_{ID',1}, \dots, SK'_{ID',l})$. and output $PK, master-key, \{SK_{ID}\}_{ID \in \{0,1\}^l}$

$\mathsf{Encrypt}(PK, S)$: Run $SigKeyGen(1^z)$ algorithm to obtain a signature signing key K_{sig} and a verification key V_{sig}. Assume that $V_{sig} = e_1 \dots e_z$. For a given $S = (c, d)$, run $\mathsf{Encrypt}$' to obtain header Hdr and encryption key K

$$Hdr, K \leftarrow \mathsf{Encrypt}'(PK, S)$$

and output the pair (Hdr, K).

$\mathsf{Decrypt}(S, ID, SK_{ID}, Hdr)$: Let $Hdr = ((C_0, C_1), \sigma, V_{sig})$.

1. Verify that σ is the valid signature of (C_0, C_1) under the key V_{sig}. If invalid, output $\perp$.
2. Otherwise, encode ID to $ID' = ID||\underbrace{* * \cdots *}_{z}$, run $\mathsf{Decrypt}'(S, ID', SK_{ID}, Hdr)$ and output encryption key K.

Similarly, if ID contains $*$ then the extension proposed in section 5.2 is applied.

Correctness can be shown with a similar calculation to the one in section 5. Note that the user key size increases from $O(l^2)$ to $O((l+z)l)$ and the header size is enlarged by the size of a signature and a verification key.

Theorem 4. *Let $\mathbb{G}$ be a bilinear group of prime order p. For any positive integer l, the above public key broadcast encryption system Π is $(t, \epsilon_1 + \epsilon_2, l, m, q_D)$ CCA-secure assuming the public key broadcast encryption system Π' is $(t', \epsilon_1, l+z, m, 0)$ semantically secure in $\mathbb{G}$ and signature scheme is $(t'', \epsilon_2, z, 1)$ strongly existentially unforgeable. And $t < t' - (2(l+z)\boldsymbol{a} + 2\boldsymbol{p})q_D - t_s$, where $\boldsymbol{a}$ is point addition time, $\boldsymbol{p}$ is pairing time, and t_s is sum of $SigKeyGen$, $Sign$ and $Verify$ computation time.*

Proof. Suppose there exists a t-time adversary, $\mathcal{A}$, such that $|AdvBr_{\mathcal{A},\Pi} - 1/2| > \epsilon_1 + \epsilon_2$. We build an algorithm $\mathcal{B}$, that has advantage $|AdvBr_{B,\Pi'} - 1/2| > \epsilon_1$ in $\mathbb{G}$. Algorithm $\mathcal{B}$ proceeds as follows.

Init: Algorithm $\mathcal{B}$ runs $\mathcal{A}$ and receives set S^* in which users $\mathcal{A}$ wishes to be challenged on. And $\mathcal{B}$ runs the $SigKeyGen$ algorithm to obtain a signature signing key K^*_{sig} and a verification key $V^*_{sig} \in \{0,1\}^z$. Let $V^*_{sig} = e_1 \dots e_z$, then $\mathcal{B}$ makes $S^{**} = \{U||V^*_{sig} \mid U \in S^*\}$ and outputs it.

Setup:(l, m) $\mathcal{B}$ gets the public key PK of Π' and also gets secret keys $SK_{ID'}$ for revoked $ID' \notin S^{**}$ from challenger $\mathcal{C}$. Note that $ID' \notin S^{**}$ iff $\forall x$ such that $x \preceq ID'$, $x \notin S^{**}$. In addition, $ID \in S^*$ iff $\forall x$ such that $x \preceq ID$, $x \in S^*$.

Since Π' can generate secret keys in a compressed way using $*$, wlog, ID' can be categorized into the following two formats:

1. $ID' = ID||\underbrace{* * \cdots *}_{z}$ for $ID \notin S^*$
2. $ID' = ID||\underbrace{* \cdots *}_{k-1}\bar{e}_k\underbrace{* \cdots *}_{z-k}$ for $ID \in S^*$ and $k \in \{1, \dots, z\}$.

$\mathcal{B}$ responds with PK and secret keys $SK'_{ID'}$ of the first type of ID'. (Recall that the secret key $SK_{ID} = SK'_{ID'}$ where $ID' = ID||\underbrace{* * \cdots *}_{z}$.) The secret keys $SK'_{ID'}$ of the second type of ID' are used to respond to the decryption queries of $\mathcal{A}$ as described in the below.

Query phase1: Algorithm $\mathcal{A}$ issues decryption queries. Let (ID, S, Hdr) be a decryption query where $S \subseteq S^*$ and $ID \in S$. Let $Hdr = ((C_0, C_1), \sigma, V_{sig})$. Algorithm $\mathcal{B}$ responds as follows:

1. Run $Verify$ to check the signature σ on (C_0, C_1) using verification key V_{sig}. If the signature is invalid $\mathcal{B}$ responds with $\perp$.
2. If $V_{sig} = V^*_{sig}$, a *forge* event happens, algorithm $\mathcal{B}$ outputs a random bit $b \xleftarrow{\$} \{0,1\}$, and aborts the simulation.
3. Otherwise, $\mathcal{B}$ decrypts the header using the second type of secret keys. Let $V = \underbrace{* \cdots *}_{k-1}\bar{e}^*_k\underbrace{* \cdots *}_{z-k}$ where $k \in \{1, \dots, z\}$. Since $V_{sig} \neq V^*_{sig}$, $V_{sig} \preceq V$. Hence, $\mathcal{B}$ can compute $SK_{ID||V_{sig}}$ from $SK_{ID||V}$. Using $SK_{ID||V_{sig}}$, B can regenerate $K \leftarrow \mathsf{Decrypt}'(S, ID, SK_{ID||V_{sig}}, (C_0, C_1))$.

Challenge: $\mathcal{B}$ gets the challenge (Hdr, K^*) from $\mathcal{C}$. To generate challenge for $\mathcal{A}$, $\mathcal{B}$ computes Hdr^* as follows:

$$\sigma^* \leftarrow Sign(Hdr, K^*_{sig})$$
$$Hdr^* \leftarrow (Hdr, \sigma^*, V^*_{sig})$$

$\mathcal{B}$ replied with (Hdr^*, K^*) to $\mathcal{A}$.

Query phase2: Same as in query phase1.

Guess: The $\mathcal{A}$ outputs a guess $b \in \{0, 1\}$. $\mathcal{B}$ outputs b.

We see that algorithm $\mathcal{B}$ can simulate all queries to run $\mathcal{A}$. $\mathcal{B}$'s success probability as follows:

$$\begin{aligned} |AdvBr_{B,\Pi'} - \frac{1}{2}| &\geq |AdvBr_{\mathcal{A},\Pi} - \frac{1}{2}| - Pr[forge] \\ &> (\epsilon_1 + \epsilon_2) - Pr[forge] \end{aligned}$$

To conclude the proof of Theorem 4 it remains to bound the probability that $\mathcal{B}$ aborts the simulation as a result of *forge*. We claim that $Pr[forge] < \epsilon_2$. Otherwise one can use $\mathcal{A}$ to forge signatures with probability at least ϵ_2. Briefly, we can construct another simulator that knows the private key, but receives K^*_{sig} as a challenge in an existential forgery game. In the above experiment, $\mathcal{A}$ causes an abort by submitting a query that includes an existential forgery under K^*_{sig} on some ciphertexts. Our simulator is able to use this forgery to win the existential forgery game. Note that during the game the adversary makes only one chosen message query to generate the signature needed for the challenge ciphertext. Thus, $Pr[forge] < \epsilon_2$.

It now follows that $\mathcal{B}$'s advantage is at least ϵ_1 as required.

7 Experiment

In this section, we present the implementation of our combinatorial subset difference (CSD) algorithm which provides organized subsets that can exclude revoked users with proposed *wildcard* (*) notation.

We obtain the execution time, power consumption of encryption and decryption, and the ciphertext header size in the proposed CSD, the existing subset difference (SD) [14, 3], and the interval algorithm [23].

To measure the execution time and the energy consumption, we have implemented the algorithms based on the PBC (pairing based cryptography) library with *a_param* and executed them on Intel Edison with a 32-bit Intel Atom processor 500 MHz and ublinux 3.10.17.

Table 3 represents the encryption/decryption time in seconds (per subset) in proposed CSD, SD, and interval schemes. While encryption/decryption of the CSD scheme and encryption of interval scheme perform point additions only, encryption/decryption of SD scheme and decryption of interval scheme compute point multiplications which are much slower than point additions. Therefore, as the depth increases, encryption/decryption of SD scheme and decryption of interval scheme become slower. The CSD scheme enhances encryption performance per subset by 6 and 1.3 times and decryption performance by 10 and 19 times compared with SD and interval schemes, respectively. While decryption is performed for a single subset in every scheme, encryption should be applied for all subsets. Hence the total encryption time is computed as the multiplication of the number of subsets and the execution time per subset. Since the average

power consumption to execute every scheme remains as 0.67 W, the total energy consumption is just proportional to the total execution time.

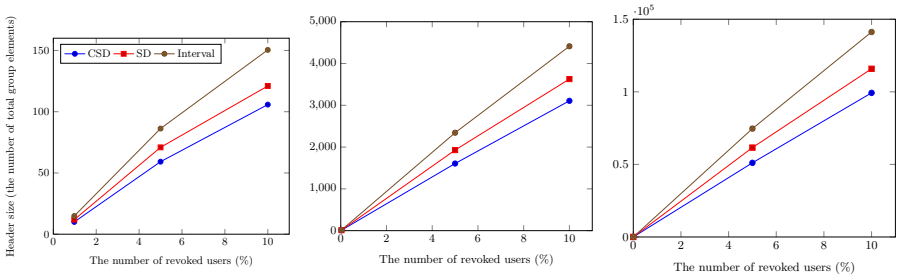
Table 3: Encryption and decryption time per subset in CSD, SD, and interval schemes on Atom processor. Average power consumption is 0.67 W.

	Depth	CSD	SD [14, 3]	Interval [23]
Enc (s)	10bit	0.20	0.70	0.24
	15bit	0.20	0.94	0.25
	20bit	0.20	1.18	0.25
Dec (s)	10bit	0.17	1.02	1.67
	15bit	0.17	1.37	2.41
	20bit	0.17	1.75	3.17

Figure 6 illustrates the number of subsets in CSD, SD, and interval algorithms by randomly generating $1\% \sim 10\%$ revoked users among 2^{10}, 2^{15}, and 2^{20} users. In the graphs, x-axis represents the ratio of revoked users and y-axis denotes the header size. The header size is proportional to the number of subsets and the number of group elements per subset. While the number of group elements per subset is 2 in SD and CSD, it is 3 in the interval algorithm. Therefore, the interval scheme requires a larger header than SD and CSD even if it generates fewer subsets than SD and CSD. In the result, the number of subsets and the total number of group elements are $0.96r$ and $1.92r$ in CSD, $1.15r$ and $2.3r$ in SD, and $0.93r$ and $2.8r$ in interval scheme on average for r revoked users. It indicates that the proposed CSD reduces the header size by 16.6% and 31.4% than SD and interval schemes, respectively.

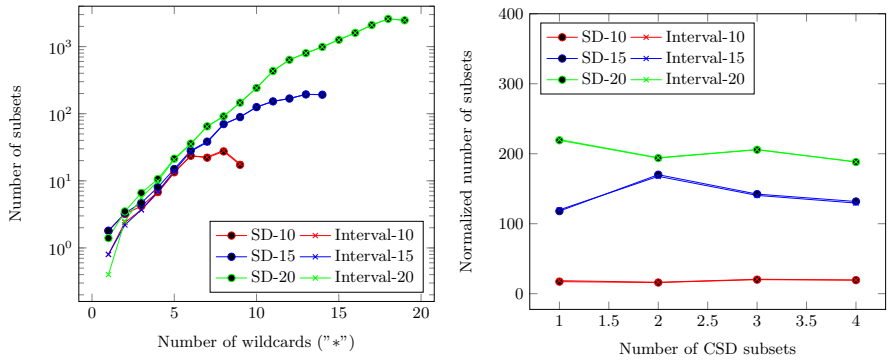
Figure 7 (a) represents how many subsets are required in SD and interval scheme formats for a subset represented in CSD format. In the graph, y-axis represents the number of subsets represented in SD and interval scheme formats, and x-axis presents the number of wildcards in a single CSD subset, where SD-l and Interval-l indicate that the number of users is 2^l. Generally, more subsets are required in SD and interval scheme formats as more wildcards are included in a CSD subset format. For instance, $15 \sim 20$ SD/interval subsets are required for a single CSD subset including 5 wildcards. Likely, 250 SD/interval subsets are required for a single CSD subset including 10 wildcards; 1250 SD/interval subsets are required for a single CSD subset including 15 wildcards.

Figure 7 (b) shows the number of subsets in SD and interval scheme representations normalized by the number of CSD subsets where the total numbers of users are $2^{10}, 2^{15}$, and 2^{20}. In the experiment, wildcards are randomly generated in CSD subsets. Regardless of the number of CSD subsets, SD and interval schemes generate 18, 120, and 220 times more subsets than CSD scheme on average for 2^{10}, 2^{15}, and 2^{20} users, respectively.



(a) 2^{10} users (b) 2^{15} users (c) 2^{20} users

Fig. 6: Ciphertext header size in CSD, SD and interval schemes



(a) (b)

Fig. 7: (a) The number of subsets by varying the number of wildcards in a CSD subset, and (b) the number of subsets normalized by the number of CSD subsets in SD and interval schemes

8 Discussion

We discuss about construction of ID based broadcast encryption (IDBE). Assuming ID is restricted in $\{0,1\}^l$, the main CSD scheme becomes IDBE if we set a public key of CSD as a public parameter of IDBE and generate secret keys using a master key g_2^{α}. To handle arbitrary identity such as email ID $\in \{0,1\}^*$, we extend the CSD scheme by hashing ID with a collision resistant hash function $H : \{0,1\}^* \rightarrow \{0,1\}^l$ during key generation and encryption. The security of the extended scheme using H is implied by the security of the original CSD scheme by a standard argument.

We also examine extension to hierarchical identity based BE (HIDBE). If we consider single-bit ID $\in \{0,1,*\}$ then the general CSD scheme itself becomes HIDBE. For example, a higher level ID vector $(0,*,*)$ can generate secret keys for its lower ID vectors $(0,0,*)$ and $(0,1,*)$. Although this HIDBE version is very limited, it gives us intuition for an expandability aspect.

Finally, HIDBE with general IDs can be constructed by amplifying IDBE. For k-level l-bit IDs, kl-level IDBE is accounted. (Again, we assume a collision resistant hash function $H : \{0,1\}^* \rightarrow \{0,1\}^l$ during key generation and encryption.) For irregular identities $ID = (I_1, \cdots, I_k)$ with $I_i \in \{0,1\}^*$ for $i = 1, \cdots, k$, we hash each I_i with H to convert identities in the format: $(H(I_1) = \{0,1\}^l, \cdots, H(I_k) = \{0,1\}^l)$. The kl-level IDBE scheme can handle this format of identities and by allowing $*$ notations we can construct HIDBE with general IDs.

9 Conclusion

We propose a combinatorial subset difference public key broadcast encryption (CSD) scheme which provides a generalized subset different representation allowing wildcards in any bit position. The proposed CSD is applicable to a secure multicast as well as minimizes the header size compared with existing BE schemes. We prove semantic security of our proposed CSD scheme under l-BDHE assumption without the random oracle model.

Experimental results show that the proposed CSD scheme not only reduces the ciphertext header size by 17% and 31%, but also improves encryption performance (per subset) by 6 and 1.3 times and decryption performance by 10 and 19 times compared with SD and interval schemes, respectively. In addition, for subsets represented in a non-hierarchical manner, the proposed CSD reduces the number of subsets by a factor of 1000 times compared with SD and interval schemes.

References

1. AACS - Advanced Access Content System. In *http://www.aacsla.com*, 2006.
2. D. Boneh and X. Boyen. Efficient Selective Identity-Based Encryption Without Random Oracles. *J. Cryptology*, 24(4):659–693, 2011.
3. D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.
4. D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-Ciphertext security from Identity-Based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.
5. D. Boneh and M. K. Franklin. Identity-Based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.
6. D. Boneh, C. Gentry, and B. Waters. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In V. Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005.
7. D. Boneh and M. Hamburg. Generalized identity based and broadcast encryption schemes. In *Advances in Cryptology-ASIACRYPT 2008*, pages 455–470. Springer, 2008.

8. D. Boneh, A. Sahai, and B. Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *Advances in Cryptology-EUROCRYPT 2006*, pages 573–592. Springer, 2006.
9. D. Boneh and B. Waters. A fully collusion resistant broadcast, trace, and revoke system. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 211–220. ACM, 2006.
10. R. Canetti, J. A. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *Proceedings IEEE INFOCOM '99*, pages 708–716, 1999.
11. R. Canetti, S. Halevi, and J. Katz. Chosen-Ciphertext security from Identity-Based encryption. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 207–222, 2004.
12. J. H. Cheon, N. Jho, M. Kim, and E. S. Yoo. Skipping, cascade, and combined chain schemes for broadcast encryption. *IEEE Trans. Information Theory*, 54(11):5155–5171, 2008.
13. C. Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In *Advances in Cryptology–ASIACRYPT 2007*, pages 200–215. Springer, 2007.
14. Y. Dodis and N. Fazio. Public Key Broadcast Encryption for Stateless Receivers. In J. Feigenbaum, editor, *Digital Rights Management Workshop*, volume 2696 of *Lecture Notes in Computer Science*, pages 61–80. Springer, 2002.
15. K. Emura, A. Miyaji, A. Nomura, K. Omote, and M. Soshi. A Ciphertext-Policy Attribute-Based encryption scheme with constant ciphertext length. In *Information Security Practice and Experience, 5th International Conference, ISPEC*, pages 13–23, 2009.
16. A. Fiat and M. Naor. Broadcast Encryption. In D. R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, 1993.
17. C. Gentry and B. Waters. *Advances in Cryptology - EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, chapter Adaptive Security in Broadcast Encryption Systems (with Short Ciphertexts), pages 171–188. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
18. M. T. Goodrich, J. Z. Sun, and R. Tamassia. Efficient Tree-Based Revocation in Groups of Low-State Devices. In M. K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 511–527. Springer, 2004.
19. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS*, pages 89–98, 2006.
20. D. Halevy and A. Shamir. The LSD Broadcast Encryption Scheme. In M. Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 47–60. Springer, 2002.
21. A. Joux. A one round protocol for tripartite Diffie-Hellman. *J. Cryptology*, 17(4):263–276, 2004.
22. K. Lee, W. K. Koo, D. H. Lee, and J. H. Park. Public-key revocation and tracing schemes with subset difference methods revisited. In *European Symposium on Research in Computer Security*, pages 1–18. Springer, 2014.
23. H. Lin, Z. Cao, X. Liang, M. Zhou, H. Zhu, and D. Xing. How to construct interval encryption from binary tree encryption. In *International Conference on Applied Cryptography and Network Security*, pages 19–34. Springer, 2010.

24. D. Naor, M. Naor, and J. Lotspiech. Revocation and Tracing Schemes for Stateless Receivers. In J. Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, 2001.
25. M. Naor and B. Pinkas. Efficient trace and revoke schemes. In *Financial cryptography*, pages 1–20. Springer, 2001.
26. M. Naor and B. Pinkas. Efficient trace and revoke schemes. *Int. J. Inf. Sec.*, 9(6):411–424, 2010.
27. D. M. Wallner, E. J. Harder, and R. C. Agee. Key management for multicast: Issues and architectures. In *Internet draft*, 1999.
28. D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *Proceedings of the ACM Conference on Computer and Communications Security*, CCS '04, pages 354–363, New York, NY, USA, 2004. ACM.
29. Z. Zhou and D. Huang. On efficient ciphertext-policy attribute based encryption and broadcast encryption: extended abstract. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS*, pages 753–755, 2010.