sensors

Article

Combinatorial Subset Difference—IoT-Friendly Subset Representation and Broadcast Encryption

Jiwon Lee [1], Seunghwa Lee [2], Jihye Kim [3] and Hyunok Oh [1,*]

[1] Department of Information System, Hanyang University, Seoul 04763, Korea; jiwonlee@hanyang.ac.kr
[2] Department of Security Enhanced Smart Electric Vehicle, Kookmin University, Seoul 02707, Korea; ttyhgo@kookmin.ac.kr
[3] Department of Electrical Engineering, Kookmin University, Seoul 02707, Korea; jihyek@kookmin.ac.kr
* Correspondence: hoh@hanyang.ac.kr

Received: 28 April 2020; Accepted: 27 May 2020; Published: 2 June 2020

Abstract: In the Internet of Things (IoT) systems, it is often required to deliver a secure message to a group of devices. The public key broadcast encryption is an efficient primitive to handle IoT broadcasts, by allowing a user (or a device) to broadcast encrypted messages to a group of legitimate devices. This paper proposes an IoT-friendly subset representation called Combinatorial Subset Difference (CSD), which generalizes the existing subset difference (SD) method by allowing wildcards ($*$) in any position of the bitstring. Based on the CSD representation, we first propose an algorithm to construct the CSD subset, and a CSD-based public key broadcast encryption scheme. By providing the most general subset representation, the proposed CSD-based construction achieves a minimal header size among the existing broadcast encryption. The experimental result shows that our CSD saves the header size by 17% on average and more than 1000 times when assuming a specific IoT example of IP address with 20 wildcards and 2^{20} total users, compared to the SD-based broadcast encryption. We prove the semantic security of CSD-based broadcast encryption under the standard l-BDHE assumption, and extend the construction to a chosen-ciphertext-attack (CCA)-secure version.

Keywords: broadcast encryption; public key encryption; IP multicast; subset difference; wildcard

1. Introduction

In the recent applications, the Internet of Things (IoT) systems are more likely to involve multicast of privacy-sensitive information; for example, an IoT sensor network in the smart city involves personal whereabouts transmitted to multiple devices, and an IoT medical system requires sensitive health information to be delivered to the authorized devices. In these IoT secure communications, cryptographic primitives can provide useful functionalities and efficiencies. Especially, advanced encryption protocols like attribute-based encryption or broadcast encryption can handle simultaneous multicast efficiently; attribute-based encryption is often applied to the IoT devices [1,2], and broadcast encryption, which is a specific and efficient version of the attribute-based encryption, is also considered [3] for secure firmware updates in IoT.

Until now, existing cryptographic applications in IoT systems such as those in References [1–4] assumed device ID as an intrinsic identity string. However, in a network environment in which the IoT device is identified by an IP address, it is useful to use IP address as an ID. Moreover, an arbitrary network group can be effectively represented by IP address including wildcard (*), and if a part of IP is connected to attributes, a device group specified by a set of attributes can also be effectively expressed. This flexible representation denoted by IP characteristics or attributes covers any specific group even without knowing every individual ID predefined. This paper focuses on the efficient and scalable public key broadcast encryption scheme that transmits a message securely to any group of legitimate receiver represented as an flexible IP address in the IoT environment.

Broadcast Encryption. The public key broadcast encryption is an effective cryptosystem for a secure group communication which allows a user (or a device) to broadcast secure messages so that only privileged users (or devices) can decrypt them. To minimize the broadcasting payload, the design of broadcast encryption adopts the hybrid encryption; an original message is often encrypted with a simple symmetric key encryption (e.g., advanced encryption standard (AES)), and the applied symmetric key becomes an official message for public key broadcast encryption. The encrypted key is called a header, which is often regarded as an official ciphertext in the broadcast encryption literature. For the transmission, the privileged users are organized within multiple subsets according to the subset representation. Then the broadcast encryption algorithm is repeatedly applied to each subset; the headers are collected into a vector so that the receiver can find and decrypt the header for his own subset. The symmetric key encryption of the original message is broadcast to the entire users, but only privileged users can decrypt the symmetric key from the header and obtain the message.

Subset Representation. In the broadcast encryption, a subset representation is an important factor which strictly determines the number of subsets. The number of subsets decides the total header size since each subset requires an individual header. Therefore, it is recommended to maintain the subset representation as general as possible so that the representation can cover as many as privileged users, which can decrease the number of subsets. In the broadcast encryption, the subset representation is considered separately: the construction assumes there already exists pre-defined subsets covering the given privileged users.

Existing Methods. The well-known subset representations are complete subtree (CS), subset difference (SD), and interval. They are all based on the binary tree structure, where the users are denoted as leaf nodes of a binary tree. Figure 1 visualizes each representation with an example where green nodes are privileged and red nodes are revoked.

The CS representation covers the privileged users by denoting the parent node; each parent node itself is a subset which includes the descendant leaf nodes. For the example in Figure 1, node 9 covers users 17 and 18, and node 19 covers only user 19. The CS representation requires 8 subsets to cover the privileged users in Figure 1.

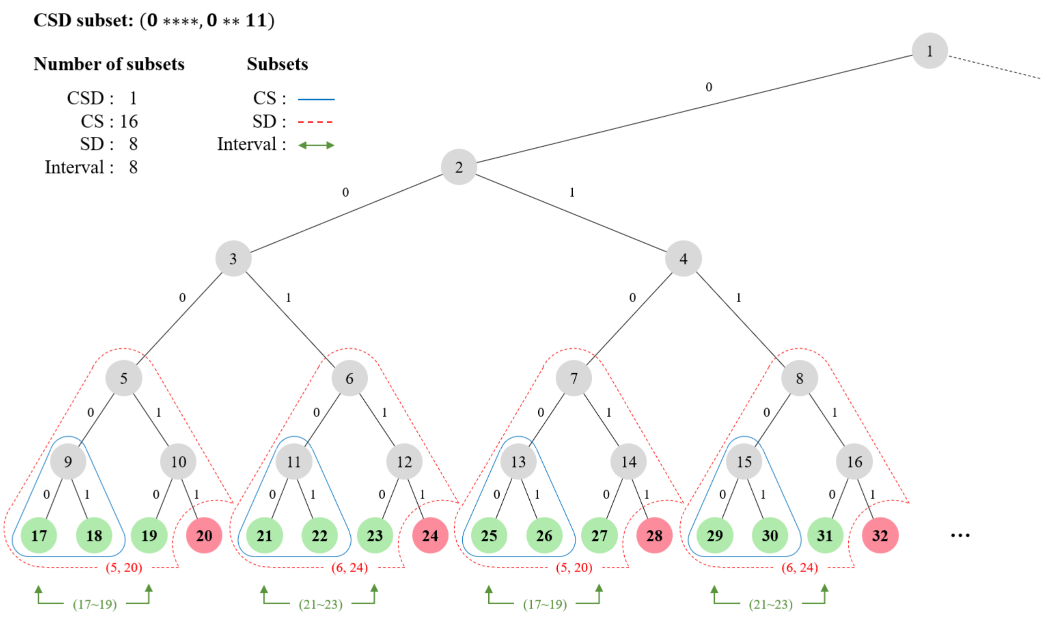


Figure 1. Example: subset representations in existing methods and Combinatorial Subset Difference (CSD).

The SD representation extends the CS by enhancing expressiveness; its covers users by denoting the difference of two subtrees as (c, d) where c is a covered set and d is a revoked set. For the example in Figure 1, a subset $(5, 20)$ which subtracts subtree 20 from subtree 5 covers users 17, 18, and 19. Since it generalizes the CS representation by adding a revoked set expression, the SD representation requires 4 subsets to cover the privileged users which is less than the CS representation.

The interval representation covers users within an interval, by denoting the left and right edge of the interval. For the example in Figure 1, a subset (17∼19) includes users 17, 18, and 19. The interval representation requires 4 subsets to cover the privileged users, which is comparable to the SD representation.

Limitations. The main restriction of the existing methods is that they are all limited to the *hierarchical* tree structure. When expressing the node with a binary label instead of a number, tree-based representations cannot effectively handle non-hierarchical combinations required to represent flexible IP addresses we consider. That is, as in Figure 1, each node can be expressed with a bitstring label which consists of $\{0, 1, *\}$, by considering the left branch as 0, right branch as 1, and both as wildcard $(*)$ starting from the root. For instance, node 8 can be expressed as a label 011, and a label 01$*$ can include nodes 7 (010) and 8 (011). In this case, the limitation is that each bit is determined from the root to the leaf node following the tree hierarchy; the wildcard cannot exist before 0 or 1 since the parent should be determined before the child nodes. For example, a label $*$00 is not expressible since the wildcard stands before 0, which makes it as a non-existent node in a binary tree.

A main concern is that this limitation prevents the broadcast encryption from being efficiently applied to the IoT systems. Consider the secure group communication of IoT systems where a device is identified by its IP address combined with attributes which is not restricted to the hierarchy. In other words, the wildcard occurs in any position to express a specific group representation (e.g., $0 * * * . * * 1. * * * .001$). However, since existing representation methods limit the position of wildcards as we described above, they cannot cover the general and flexible IP addresses within a single subset, which leads to a large number of subsets.

Combinatorial Subset Difference. To overcome the hierarchical limitations, we propose a new IoT-friendly subset representation called *combinatorial subset difference (CSD)*, which extends the SD representation by allowing any combinatorial labels, that is, allowing wildcards in any position. Specifically, the CSD expresses a subset similar to SD as (c, d), but where c is a covered label and d is a revoked label that can allow wildcards in any bits. Our CSD is the most general representation among the existing methods, indicating that any subset expressed in the existing representation can be transformed to the CSD subset. Therefore, the CSD can achieve minimal number of subsets in any cases. To cover the privileged users in the example of Figure 1 (by expressing each user as a label), the CSD requires only a single subset $(0 * * * *, 0 * * 11)$ while SD and interval representations require 4 different subsets.

To adopt the CSD representation in practical broadcast encryption, it is required to construct an algorithm that can output appropriate CSD subsets from the given input of privileged users. Thus, we propose a heuristic algorithm which generates CSD subsets from the list of users. We first construct an SD algorithm that can output SD subsets from the input of privileged users, then extend the algorithm to cover the general CSD cases. The proposed CSD algorithm generates r subsets for the worst case while the SD generates $2r - 1$ for the worst case, where r is the number of revoked users. For any cases, the algorithm guarantees that the number of CSD subsets are always no more than the SD subsets.

CSD-based broadcast encryption. Designing the broadcast encryption is independent of the subset representation; since we proposed a new subset representation, it is also required to design a new broadcast encryption that can adopt the input of CSD subsets. In this paper, we propose a new CSD-based public key broadcast encryption that has a minimal total header size among the existing broadcast encryption due to the minimal number of subsets. The minimal header size indicates minimal transmission cost, which can be suitable for the IoT system applications. Moreover, our CSD-based broadcast encryption improves the encryption time and decryption time while maintaining the order of key sizes.

Table 1 shows the comparison between the CSD and the existing public key broadcast encryption schemes. The public key SD is from the specifically measured by applying the public key lifting transformation [5], which combines the hierarchical identity-based encryption (HIBE) [6] to the symmetric key SD-based broadcast encryption [7,8] from the advanced access content system (AACS) DVD standard [9]. The interval refers to the Lin's construction [10], which is based on a new interval representation from binary trees. The revocable SD refers to Lee and Park's construction [11], which is based on the same SD representation, but improves the order of key sizes and enhances identity revocation algorithm. The elements in revocable SD is within a composite order group, where the element size is 8 times larger than the normal prime order group for the security; the size of composite order group elements are denoted as $\times 8$. The header size refers to the total header size in the worst case, which is the product of the number of group elements per subset and the number of subsets. Among the state-of-the-art public key broadcast encryption, our CSD achieves a minimal header size of $2r$, due to the minimal number of subsets.

Table 1. Size and performance comparison among public key broadcast encryptions.

	Public Key SD ([6,7])	**Interval [10]**	**Revocable SD [11]**	**CSD**
PK (size)	$\log n$	$2 \log n$	7×8	$4 \log n$
SK (size)	$\log^3 n$	$2 \log^2 n$	$4 \log^2 n \times 8$	$3 \log^2 n$
Hdr (size)	$4r$	$3r$	$(3 \times 8 + 1) \cdot 2r$	$2r$
# subsets	$2r$	r	$2r$	r
# group elements	2	3	$3 \times 8 + 1$	2
Enc (computation)	$2r\mathbf{e} \log n$	$4r\mathbf{e}$	$12r\mathbf{e}$	$3r\mathbf{e}$
Dec (computation)	$\mathbf{e} \log n + 2p$	$2\mathbf{e} \log n + 4\mathbf{p}$	$\mathbf{e} + 4\mathbf{p}$	$2\mathbf{e} + 2\mathbf{p}$
Assumption	$l - BDHI$	$l - BDHE$	$DBDH$	$l - BDHE$

e = point multiplications, p = pairings, n = total users, r = revoked users.

The CSD also improves the encryption time and decryption time, compared to the other constructions. Nevertheless, the CSD still maintains the public key size within $O(\log n)$ and the secret key size within $O(\log^2 n)$, which is comparable to the existing public key broadcast encryption. To verify the theoretical improvements, we implemented the CSD, SD, and interval broadcast encryption on the Intel Edison embedded systems: the experimental results show that our CSD reduces the header size by 1,000 times on average when applied to the IP address examples, with also improving both encryption and decryption time, compared to the existing methods.

The security of our CSD-based public key broadcast encryption is proven under the standard bilinear diffie-hellman exponent ($l - BDHE$) assumption; we first prove the semantic security (CPA-secure) of our original construction, and then extend the construction to a CCA-secure version.

Contributions. We now summarize the contribution of our paper, in the sense of practicality and theoretical advances.

- **CSD subset representation:** we propose a new IoT-friendly subset representation called combinatorial subset difference (CSD), the most general representation that achieves minimal number of subsets among the existing methods.
- **CSD covering algorithm:** we construct a subset generation algorithm for the proposed CSD representation, which can output CSD subsets from the given input of privileged users.
- **CSD-based broadcast encryption:** we propose a new public key broadcast encryption based on the CSD representation, which achieves a minimal header size due to the minimal number of

subsets, suitable for the secure group communication in IoT systems. It also improves encryption and decryption time compared to the existing public key broadcast encryption.
- **Security proof:** we provide a formal security proof for the semantic security of our construction under the $l - BDHE$ assumption, in the standard model.
- **CCA-secure extension:** we also propose an extended CCA-secure version of our CSD-based public key broadcast encryption, along with a formal proof for the CCA security.
- **Implementation:** we implemented our construction on the actual IoT device to verify the performance, and provide experimental results showing comparison between the CSD and the existing broadcast encryption schemes.

This paper is organized as follows. Section 2 states related works. Section 3 describes the subset construction algorithm. Section 4 organizes preliminaries for broadcast encryption systems. We present our broadcast encryption scheme in Section 5, and prove the security for security analysis in Section 6. Then we extend it to a CCA-secure broadcast encryption scheme in Section 7, and prove the security in Section 8. Section 9 shows experimental results. Section 10 makes conclusion.

2. Related Work

2.1. Secure Multicast

Secure multicast is a traditional encryption model for multi-user infrastructure, which considers an environment that encrypts messages to multiple users simultaneously. Most of the secure multicast protocols work with a central manager responsible for the broadcast, and focuses on improving the efficiency of key distribution and transmission.

Among the secure multicast research, Chu et al. [12] proposed a notable end-to-end protocol which can secure both message and the copyright of the content. Their work support the dynamic group, which can let users join/leave or expel users during the system running. However, the security of their protocol relies on informal analysis against few attacks, and the fundamental technique remains on multiple encryption which let the encryption required for the number of total users in the group.

For the efficiency, Kumar et al. [13] proposed a protocol based on the centralized key distribution. By providing more efficient key distribution from the hierarchical tree structure, their protocol could achieve more flexible group control and efficient computation. Still, the total communication relies on the individual transmissions, which leads to a considerably large transmission load.

Recently, Wang et al. [14] implemented the concept of secure multicast on the IoT environment, by proposing an end-to-end key agreement protocol for the IoT devices. It focuses on the authentication of devices when interacting with the user's server, where the user can open a secure channel if the key is once agreed. For the security, their protocol deploy fuzzy extractor, which can use user's biometric information such as fingerprints as a secret key.

2.2. Broadcast Encryption

The broadcast encryption is a traditional cryptosystem where one can encrypt a message to the group of multiple users. In the broadcast encryption, the encryption can be handled for each group rather than individuals; it is an efficient primitive to be deployed on the secure multicast.

There are many constructions for the broadcast encryption, with different functionalities and improvements [5,6,8,15–29]. In the cryptographic literature, the broadcast encryption is categorized by two different criteria: symmetric/public key, and stateful/stateless.

Symmetric vs. Public. Similar to the categorization of standard encryption systems, the broadcast encryption can be either be based on the symmetric key settings or the public key (or asymmetric) settings. If the broadcast encryption is based on the symmetric key [25,29], only a user with the key can encrypt the message, that is, only the central authority can broadcast messages to other users. On the other hand, if the broadcast encryption is based on the public key [8,10,23], anyone can broadcast messages to other users.

Stateful vs. Stateless. The stateful broadcast encryption and stateless broadcast encryption is determined by the key update. In the stateful broadcast encryption [28], the secret key of users can be updated time to time, in order to help the revocation (proper broadcast). However, in the stateless broadcast encryption [8,25,26], the system allows the key exchange for only once in the initial setup. Stateful broadcast encryption can be easier to design with the help of key updates, but it also involves a price of simultaneous key managements. On the other hand, in the stateless broadcast encryption, the system can change revocation without any key management after the initialization.

In many interactive IoT systems, it is often required to allow each device to broadcast messages rather than restricting the broadcast to the central manager, which requires public key broadcast encryption. Also, stateless broadcast encryption is more appropriate than the stateful settings for the IoT systems, since small and numerous IoT devices are not suitable for frequent updates and managements. In these sense, the proposed broadcast encryption for combinatorial subset difference is a stateless public key broadcast encryption, which satisfies the IoT requirements.

3. Proposed Subset Construction Algorithm

Despite the efficiency of the combinatorial subset difference (CSD) representation, it is a remaining work to construct a concrete algorithm which returns CSD subsets from a list of privileged/revoked users. In this section, we propose a heuristic algorithm for the CSD subset construction. The proposed algorithm does not assure the optimal minimization of the subsets. However, it guarantees the number of subsets in CSD is no more than the existing subset difference (SD) algorithm [8]. For the worst case, it generates no more than r subsets while the existing SD algorithm generates $2r - 1$ subsets.

Since CSD is the generalization of SD, we first explain the how the SD algorithm minimizes the number of subsets. Then we show how to extend it to the CSD algorithm, by focusing on the difference between the SD and the CSD.

In the SD (and CSD) representation, a subset is represented as $S = (c, d)$, where c is a covered set (nodes to include), and d is a revoked set (nodes to exclude). For instance, in a tree structure, a subset S includes all descendants of the node c, except all descendants of the node d.

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

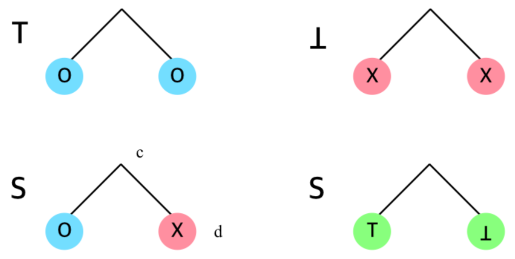


Figure 2. The definition of node types. Node O is a legitimate node and node X is a revoked node.

- Type $\top$: There is no revoked descendant of c.
- Type $\perp$: Node c is excluded by its parent, because either it is already covered by another subset or its all descendants are revoked.
- Type S : Node c becomes a subset (c, d), where all descendants of d are either revoked or already covered by other subsets.

The node type is determined by the types of its children. Therefore, the internal node type is determined from bottom to top, recursively. As in Figure 3, combining two internal nodes identifies the type (Figure 2) of their parent.

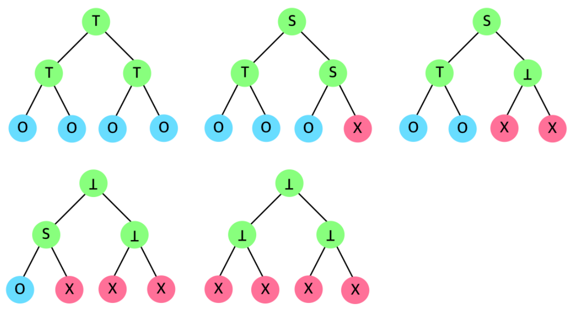


Figure 3. Node type examples in Subset Difference (SD) (and CSD).

For both SD and CSD, the examples in Figure 3 are valid. However, as in Figure 4, the CSD is different from SD in case when two S nodes are combined; the SD let the parent of two S be a type $\perp$, while CSD let it be another S. In the SD, a set can be fixed only if its descendants are all $\top$ or $\perp$. Therefore, two S nodes cannot be merged into a single set; the parent node becomes $\perp$ since two S type nodes are determined as individual subsets in SD. However, in CSD, the parent node becomes another S by merging two S nodes, without generating two subsets.

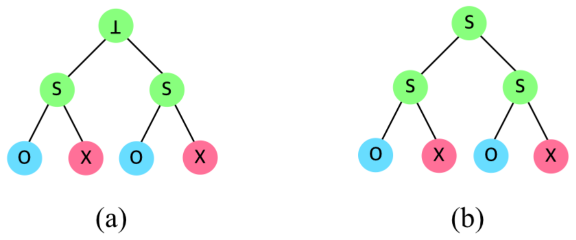


Figure 4. Category difference between (**a**) SD and (**b**) CSD.

Figure 5 shows an example where SD and CSD outputs different results, where users (nodes) 1 and 3 are revoked among 8 total users. When representing the revoked nodes in a binary format, 001 and 011 are revoked. The parent nodes are denoted as $*$. For instance, 00$*$ is a parent node of 000 and 001, and 01$*$ is a parent node of 010 and 011. The type of node 000 is $\top$ since nothing is revoked, and the type of node 001 is $\perp$ since it is revoked. The type of node 00$*$ is S, since its children 000 is included while its other children 001 is excluded, and it requires a subset $(00*, 001)$ to be covered. Similarly, the type of node 01$*$ is S. In case of the SD algorithm, the node $0**$ with type S children cannot be an inclusion label since both of its children are labeled as two different subsets $((00*, 001)$ and $(01*, 011))$ which cannot be merged. Thus, $0**$ should be excluded which determines its type as $\perp$. Then the type of the root note is S since its left child is $\perp$ and its right child is $\top$. As a result, three subsets are generated as $(***, 0**)$, $(00*, 001)$, and $(01*, 011)$.

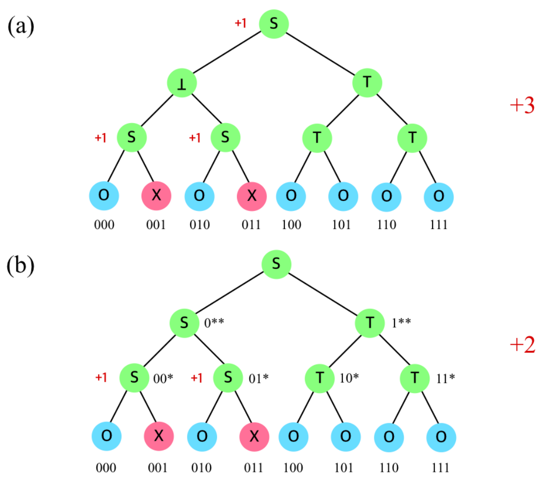


Figure 5. Subset construction in (**a**) SD, and (**b**) CSD.

On the other hand, in the CSD algorithm, node $0 * *$ is treated specially, different from the SD. Although the type of its children are both S, it is not necessary to exclude the node; if the node can be merged with a sibling node of type $\top$, it is possible to represent the node without generating a new subset. In Figure 5, the node $0 * *$ has two subsets $(00*, 000)$ and $(01*, 010)$, and it has a sibling node $1 * *$ of type $\top$. When merging subsets $(00*, 000)$ and $(01*, 010)$ with $(1 * *, \perp)$, it generates $(*0*, 000)$ and $(*1*, 010)$ where $*0*$ and $*1*$ can include all eight users while 000 and 010 can exclude 000 and 010 since the CSD allows $*$ in the middle of the representation regardless of the tree hierarchy. As a result, the CSD generates two subsets as $(*0*, 001)$ and $(*1*, 011)$, which does not generates subsets more than the number of revoked users.

Table 2 describes node type resolution in SD and CSD, where T_0 is the type of left child and T_1 is the type of right child. The subset generation indicates whether a new subset generation is required for the resolved node type; a new subset should be generated when one child is included while the other child is excluded. The only difference between SD and CSD occurs when the type of both children are S: the node is merged into type $\perp$ in SD while it is merged into another S type 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 shows how to build subsets in both SD and CSD methods. The only difference appears at line 27: the CSD proceeds an additional line 27 while the SD does not. Consequently, as shown in Table 2, line 29 ($resolveType(T_0, T_1)$) returns different values.

Algorithm 1 Subset construction for SD and CSD.

```
1: function $sd(Set, Unresolved, i, prefix, revSet)$
2:     if $i = depth$ then
3:         if $|revSet| > 0$ then
4:             return $\perp$
5:         else
6:             return $\top$
7:         end if
8:     else if $|revSet| = 0$ then
9:         return $\top$
10:    end if
11:    $S_0 = \{r | r \in revSet, bit(r, i+1) = 0\}$
12:    $S_1 = \{r | r \in revSet, bit(r, i+1) = 1\}$
13:    $T_0 \leftarrow sd(Set, U_0, i+1, prefix||0, S_0)$
14:    $T_1 \leftarrow sd(Set, U_1, i+1, prefix||1, S_1)$
15:    if $(T_0 = \perp) \wedge (T_1 = \top)$ then
16:        $s \leftarrow createSubset(prefix, i, 0)$
17:        $Unresolved \leftarrow s; Set \leftarrow Set \cup s$
18:    else if $(T_0 = \top) \wedge (T_1 = \perp)$ then
19:        $s \leftarrow createSubset(prefix, i, 1)$
20:        $Unresolved \leftarrow s; Set \leftarrow Set \cup s$
21:    else if $(T_0 = S) \wedge (T_1 = \top)$ then
22:        $\forall (c, d) \in U_0, c_i \leftarrow *; Unresolved \leftarrow U_0$
23:    else if $(T_0 = \top) \wedge (T_1 = S)$ then
24:        $\forall (c, d) \in U_1, c_i \leftarrow *; Unresolved \leftarrow U_1$
25:    else if $Combinatorial \wedge (T_0 = S) \wedge (T_1 = S)$ then
26:        // Only used for CSD
27:        $Unresolved \leftarrow U_0 \cup U_1$
28:    end if
29:    return $resolveType(T_0, T_1)$
30: end function
31: procedure $main$
32:    $revSet \leftarrow \{revoked\ user\ IDs'\}$
33:    $header \leftarrow sd(\perp, \perp, 0, \perp, revSet)$
34: end procedure
```

In the algorithm, the subset construction starts with an ID set of revoked users as an input. Function *sd* reads each ID of revoked users from the most significant bit (MSB) to the least significant bit (LSB), where *i* denotes the current index. When *i* reaches the LSB, the user is determined as either revoked or legitimate. Therefore, it returns $\perp$ or $\top$ as in lines from 2 to 7. Even if *i* does not reach LSB, it returns $\top$ if there is no revoked user. In line 11 through 12, the revoked users are divided into two sets, depending on the next bit of user's ID, and the function *sd* is called recursively for each set denoting tree traverse in lines 13 through 14. If the types of each children is $\perp$ and $\top$, a new subset should be created as described in lines 16 and 19. Function $createSubset(prefix, i, b)$ is for creating a subset (c, d) where c denotes an inclusion label and d denotes an exclusion label. Label c covers all users with $prefix_1, \ldots, prefix_i, *, \ldots,$ and label d covers all users with $prefix_1, \ldots, prefix_i, b, *, \ldots,$

where $prefix_k$ is a k-th bit of $prefix$. When the type of a child is S and the other one is $\top$, each inclusion label in the unresolved subset is updated to include the type $\top$ child, as in lines 22 and 24. In the CSD, the unresolved subsets for type S children are preserved as unresolved, since inclusion labels in unresolved subsets will be updated in case they include a type $\top$ node as in line 27.

Theorem 1. *For the combinatorial subset difference (CSD) representation, Algorithm 1 generates at most r subsets for r revoked users.*

Proof. As illustrated in Table 2, a subset is generated when a child node has a $\perp$ type. The resolved type becomes $\perp$, at least only if a child node is $\perp$. A leaf node (user) has type $\perp$ only if it is revoked. If there are r revoked nodes, there are at most r leaf nodes with type $\perp$. Hence, the number of generated subsets is no more than r. $\square$

Unlike CSD, the number of subsets in the SD can exceed r, since a node can still be type $\perp$ even if both children have type S.

For the worst case comparison, it is straightforward to deduce that the number of subsets in CSD does not exceed the number of subsets in SD. For the subset generation, a subset is created when a child is $\perp$ and the other child is $\top$, both in SD and CSD algorithm. Also in both algorithms, a node can be type $\top$ when its children are both $\top$, while it should be $\perp$ when one child is $\perp$ and the other child is $\perp$ or S. However, in SD, a node becomes $\perp$ when its children are both S, which may require an additional subset. Therefore, the number of subsets in CSD algorithm is no more than that of SD algorithm.

4. Preliminaries

4.1. Public Key Broadcast Encryption

We follow the standard definition of public key broadcast encryption from Reference [23]. In the broadcast encryption, it is often combined with the symmetric key encryption, where the encryption encapsulates the session key instead of arbitrary messages, and the encryption is performed for each subset S. Formally, a public key broadcast encryption Π contains three functions as below:

- $(PK, \{SK_{ID}\}_{ID\in\{0,1\}^l}) \leftarrow \mathsf{Setup}(l, m)$: the setup algorithm initializes the system by setting up public parameters, and generating private keys for stateless devices. From user ID length l and session key length m, the setup returns a public key PK and 2^l secret keys $\{SK_{ID}\}_{ID\in\{0,1\}^l}$.
- $(S, Hdr, C_M) = (Hdr, K) \leftarrow \mathsf{Encrypt}(PK, S)$: the encrypt algorithm is performed for each subset S. It first outputs a header Hdr and a message encryption key $K \in \mathcal{K}$ from the input of subset $S \subseteq \{0,1\}^l$ and a public key PK. Then, for the message M to be broadcast to a set S, let C_M be an encryption of M under the session key K. The transmission for the users in S is selected as (S, Hdr, C_M), where Hdr is called broadcast ciphertext and C_M is called broadcast body.
- $K \in \mathcal{K} \leftarrow \mathsf{Decrypt}(S, ID, SK_{ID}, Hdr)$: the decrypt algorithm is performed by the individual privileged user. From the input of subset $S \subseteq \{0,1\}^l$, user ID $ID \in \{0,1\}^l$, secret key SK_{ID}, and a header Hdr, if $ID \in S$ (that is privileged) then it produces the message encryption key $K \in \mathcal{K}$. Then the key K is utilized to decrypt the C_M to retrieve plaintext M.

The encryption system must let every user in S obtain message M correctly. In other words, for all $S \subseteq \{0,1\}^l$ and all $ID \in S$, if $(PK, (SK_{ID_{0^l}}, \ldots, SK_{ID_{1^l}})) \overset{\$}{\leftarrow} \mathsf{Setup(l, m)}$, $(Hdr, K) \overset{\$}{\leftarrow} \mathsf{Encrypt(PK, S)}$ then it should satisfy $\mathsf{Decrypt(S, ID, SK_{ID}, Hdr) = K}$.

For the security definition of the broadcast encryption, we describe the selective security for chosen plaintext attacks (IND-sID-CPA-security) and chosen ciphertext attacks (IND-sID-CCA-security) as in Reference [23]. Then we separate the security notions for single-set security and multi-set security, depending on the number of representations of the challenged sets. Finally we show that the single-set security implies the multi-set security.

4.2. Security Definition

We define the single-set security by the selective game between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$. Both $\mathcal{C}$ and $\mathcal{A}$ knows the ID length l and the session key length m as a default inputs. The security game proceed as follows:

Init: $\mathcal{A}$ begins by selecting a set S^* to claim the users which it intends to attack.
Setup: $\mathcal{C}$ runs $\mathsf{Setup(l,m)}$ to get a public key PK and secret keys $SK_{0^l}, \cdots, SK_{1^l}$, and gives $\mathcal{A}$ the PK and all secret keys SK_{ID} for $ID \notin S^*$.
Query phase1: *(optional for CCA)* $\mathcal{A}$ can adaptively issue decryption queries $q_1, \ldots, q_m$; each query consists of (ID, S, Hdr) for $S \subseteq S^*$ and $ID \in S$. $\mathcal{C}$ replies the query by $\mathsf{Decrypt(S, ID, SK_{ID}, Hdr)}$.
Challenge: $\mathcal{C}$ runs $\mathsf{Encrypt(S^*, PK)}$ to get (Hdr^*, K) for $K \in \mathcal{K}$. Then, $\mathcal{C}$ flips a coin $b \in \{0,1\}$. If $b = 1$, $\mathcal{C}$ sets $K^* = K$, otherwise $\mathcal{C}$ chooses a random $R \in \mathcal{K}$ and sets $K^* = R$ to respond (Hdr^*, K^*) back to $\mathcal{A}$.
Query phase2: *(optional for CCA)* $\mathcal{A}$ can continue asking decryption queries $q_{m+1}, \ldots, q_{q_D}$; each query consists of (ID, S, Hdr) for $S \subseteq S^*$ and $ID \in S$, but the only constraint is $Hdr \neq Hdr^*$. $\mathcal{C}$ replies the query same as in phase 1.
Guess: $\mathcal{A}$ guesses $b' \in \{0,1\}$ for b. If $b = b'$, $\mathcal{A}$ wins the game.

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

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

A multi-set security is also defined by the selective game between the Ch and $\mathcal{A}$, similar to the single-set security game above, but the only difference is that the challenged set is represented by multiple subsets.

Init: $\mathcal{A}'$ begins by selecting a set $S^* = (S_1^*, \cdots, S_w^*)$ to claim users which it intends to attack.
Setup: $\mathcal{C}$ runs $\mathsf{Setup(l,m)}$ to get a public key PK and secret keys $SK_{0^l}, \cdots, SK_{1^l}$, and gives $\mathcal{A}'$ the PK and all secret keys SK_{ID} for $ID \notin S^*$.
Query phase1: *(optional for CCA)* $\mathcal{A}'$ can adaptively issue decryption queries $q_1, \ldots, q_d$; each query consists of (ID, S, Hdr) where $S \subseteq S_i^*$ for all i and $ID \in S$. $\mathcal{C}$ replies the query by $\mathsf{Decrypt(S, ID, SK_{ID}, Hdr)}$.
Challenge: $\mathcal{C}$ runs $\mathsf{Encrypt(S_i^*, PK)}$ for $i = 1, \cdots, w$ to get (Hdr_i^*, K_i) for $K \in \mathcal{K}$. Then, $\mathcal{C}$ flips a coin $b \in \{0,1\}$. If $b = 1$, $\mathcal{C}$ sets $K^* = (K_1, \cdots, K_w)$, otherwise $\mathcal{C}$ chooses randoms $R_i \in \mathcal{K}$ for $i = 1, \cdots, w$ and sets $K^* = (R_1, \cdots, R_w)$ to respond (Hdr^*, K^*) back to $\mathcal{A}'$.
Query phase2: *(optional for CCA)* $\mathcal{A}'$ can continues asking decryption queries $q_{q_d+1}, \ldots, q_{q_D}$; each query consists of (ID, S, Hdr) with $S \subseteq S_i^*$ for all i and $ID \in S$, but the only constraint is $Hdr \neq Hdr^*$. $\mathcal{C}$ replies the query same as in phase 1.
Guess: $\mathcal{A}$ guesses $b' \in \{0,1\}$ for b. If $b = b'$, $\mathcal{A}$ wins the game.

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

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

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

Theorem 2. *Suppose a single-set public key broadcast encryption* Π *is* (t, ϵ, l, m, q_D)*-CCA secure. Then a multi-set public key broadcast encryption* Π' *is* $(t, \epsilon', l, m, q_D)$*-CCA secure for arbitrary* $\epsilon' < \epsilon * w$*, where* w *is the number of subsets [30].*

Proof. For the proof sketch, the basic idea is to convert the challenge key in the multi-set scheme from a real key set K^* to a random key set $\overline{K}^*$ by using hybrid games which change each key in the single-set scheme from a real key to a random key. If the adversary cannot distinguish the changes of each key in the single-set scheme, then it also cannot distinguish the changes of challenge key in the multi-set scheme since the number of hybrid games is within 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

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

Finally, we obtain the inequality relation as follows:

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

□

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.*

For the scheme construction, we first provide a semantically-secure scheme, and then extend it to gain CCA security.

4.3. Bilinear Groups and Pairings

In References [31,32], the bilinear maps and bilinear map groups are defined as follows:

- There exists two multiplicative groups $\mathbb{G}$ and $\mathbb{G}_1$, which are cyclic groups of prime order p.
- Let g be a generator of the group $\mathbb{G}$.

For two groups $\mathbb{G}$ and $\mathbb{G}_1$ as above, the bilinear map is defined as a map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$, which satisfies the following properties:

- The map is bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$
- The map is non-degenerate: $e(g, g) \neq 1$.

$\mathbb{G}$ is a bilinear group if the operation in $\mathbb{G}$ is efficient and there is a group $\mathbb{G}_1$ with an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$.

4.4. Computational Complexity Assumptions

The security of our scheme is based on the bilinear Diffie-Hellman Exponent (BDHE) assumption which is commonly used as in References [6,23]. It is a natural extension of bilinear Diffie-Hellman Inversion (BDHI) assumption.

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 $e(h, g)^{\alpha^{l+1}} \in \mathbb{G}_1$. For simplicity, after g and α are determined, 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 choices $g, h \in \mathbb{G}$, α in $\mathbb{Z}_p$, and $\mathcal{A}$'s random bits. The decisional version of the l-BDHE problem is defined in a similar manner; let $\vec{y}_{g,\alpha,l} = (y_1, \ldots, y_l, y_{l+2}, \ldots, y_{2l})$. An algorithm $\mathcal{B}$ which outputs $b \in \{0, 1\}$ has advantage ϵ in solving decisional l-BDHE if

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

where the probability is over the random choices of $g, h \in \mathbb{G}$, α in $\mathbb{Z}_p$, $T \in \mathbb{G}_1$, and $\mathcal{B}$'s random bits.

Definition 4. *The (decisional) (t, ϵ, l)-BDHE assumption holds in $\mathbb{G}$, if for any t-time adversary the advantage is less than ϵ on solving the (decisional) l-BDHE problem in $\mathbb{G}$.*

The t and ϵ are often omitted, and referred as (decisional) l-BDHE in $\mathbb{G}$.

5. CSD-based Broadcast Encryption

In this section, we present our public key broadcast encryption based on the combinatorial subset difference (CSD) representation. The CSD-based broadcast encryption accepts an input of CSD subset S; as standard broadcast encryption, it runs the broadcast encryption algorithm for each subset S. We first show the formal construction in Section 5.1, along with appropriate examples to aid the comprehension. Then we provide the formal proof for the collusion-resistant semantic security in Section 6.

5.1. Main Scheme

The main formal construction of the CSD-based public key broadcast encryption for a CSD subset S is described as follows:

$\mathsf{Setup}(l,m)$: The maximum number of users are set as 2^l, which can be interpreted as l-level public key broadcast encryption, and the message space is set as $\{0, 1\}^m$. To initialize and generate the keys, the setup algorithm selects random integers $\alpha, \beta \in \mathbb{Z}_p$, and $O(l)$ random group elements $g \in \mathbb{G}$, $g_2, g_3, h_{1,0}, h_{1,1}, \ldots, h_{l,0}, h_{l,1}, k_{1,0}, k_{1,1}, \ldots, k_{l,0}, k_{l,1} \in \mathbb{G}$. Then it computes $g_1 = g^{\alpha} \in \mathbb{G}$. The public key is set as

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

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

For an identity $ID = b_1 \cdots b_l$, the setup algorithm generates a secret key SK_{ID} from the master secret key. It selects random $r_1, \ldots, r_l \in \mathbb{Z}_p$ and set $SK_{ID} = (SK_{ID,1}, \ldots, SK_{ID,l})$ as

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

where $\bar{b}_i$ represents a bit *NOT* of b_i, that is, $1 - b_i$.

Example 1. *For an identity $ID = 010$,*

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

Encrypt(PK, S): *The encrypt algorithm takes a subset corresponding to the combinatorial subset difference (CSD) representation as an input. The CSD subset is represented as $S = (c, d)$, for $c, d \in \{0, 1, *\}^l$. The label c determines the covered (included) nodes, and the label d determines the revoked (excluded) nodes. The labels can express wildcard ($*$), which can embrace multiple nodes. For example, a subset $(**0*, 0*01)$ indicates all users $**0*$ except users in $0*01$. $**0*$ includes $0000, 0001, 0100, 0101, 1000, 1001, 1100, 1101$ and $0*01$ includes $0001, 0101$, therefore, label $(**0*, 0*01)$ covers $0000, 0100, 1000, 1001, 1100, 1101$.*

For each subset $S = (c, d)$, let $c = c_1 \cdots c_l$ and $d_1 \cdots d_l$. To generate the header Hdr and encryption key K, the encrypt algorithm selects a random $s \in \mathbb{Z}_p$ and set Hdr and K as

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

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

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 2. *For subset $(**0*, 0*01)$,*

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

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

Decrypt(S, ID, SK_{ID}, Hdr): *Consider an identity $ID = b_1 \cdots b_l$ and a subset $S = (c, d)$. Let j be an index where the bit in b and d is different, that is, such that $b_j = 1$ and $d_j = 0$, or $b_j = 0$ and $d_j = 1$. For example, if $d = 0*01$ and $ID = 0000$, then $j = 4$ since $b_4 = 0$ and $d_4 = 1$.*

From $SK_{ID} = (SK_{ID,1}, \ldots, SK_{ID,l})$, the decrypt algorithm choose $SK_{ID,j}$ such that index j satisfies the condition stated above. To retrieve decrypt key K from the header $Hdr = (A_0, A_1)$ and the secret key $SK_{ID,j} = (a_0, a_1, h_{1,\bar{b}_1}^{r_j}, \ldots, h_{l,\bar{b}_l}^{r_j}, k_{1,0}^{r_j}, k_{1,1}^{r_j}, \ldots, k_{j-1,0}^{r_j}, k_{j-1,1}^{r_j}, k_{j,b_j}^{r_j}, k_{j+1,0}^{r_j}, k_{j+1,1}^{r_j}, \ldots, k_{l,0}^{r_j}, k_{l,1}^{r_j})$, let $B = a_1 \cdot \prod_{i=1,c_i=*}^{l} h_{i,\bar{b}_i}^{r_j} \cdot \prod_{i=1,i\neq j,d_i\neq *}^{l} k_{i,d_i}^{r_j}$ and output

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

For a valid ciphertext, it satisfies that

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

For the example above, 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 $SK_{0000,4} = (g^{r_4}, g_2^{\alpha}(h_{1,0}h_{2,0}h_{3,0}h_{4,0}k_{4,1}g_3)^{r_4}, \dots)$, we compute B as follows.

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

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

Now, let us analyze the opposite case of $ID = 0010$, which is not included by an inclusion label $**0*$ from the subset $(**0*, 0*01)$. Since $SK_{0010,i} = (g^{r_i}, g_2^{\alpha}(h_{1,0}h_{2,0}h_{3,1}h_{4,0}k_{i,\bar{b}_i})^{r_i}, \dots)$ and $A_1 = (h_{1,0}h_{1,1}h_{2,0}h_{2,1}h_{3,0}h_{4,0}h_{4,1}\cdots)^s$, it is necessary to eliminate $h_{3,1}^{r_i}$ from $SK_{0010,i}$, which is not available. Thus, it is impossible to calculate B using $SK_{0010,i}$.

Let us examine $ID = 0001$ for instance, which is included in the exclusion label $0*01$ of subset $(**0*, 0*01)$. Since $SK_{0001,1} = (g^{r_1}, g_2^{\alpha}(\cdots k_{1,1})^{r_1}, \dots)$, $SK_{0001,2} = (g^{r_2}, g_2^{\alpha}(\cdots k_{2,1})^{r_2}, \dots)$, $SK_{0001,3} = (g^{r_3}, g_2^{\alpha}(\cdots k_{3,1})^{r_3}, \dots)$, $SK_{0001,4} = (g^{r_4}, g_2^{\alpha}(\cdots k_{4,0})^{r_4}, \dots)$, 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 not possible. Thus, B cannot be computed from $SK_{0001,i}$.

Notice that the encrypt algorithm is applied for each subset S. In other words, the number of Hdr is equal to the number of subsets. When the user receives a message, he chooses Hdr for the set where he belongs and call Decrypt for that specific Hdr.

6. CPA-Security Analysis

In this section, we formally prove the semantic security (IND-CPA-security) of our CSD-based broadcast encryption in Section 5 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 $t' < t - O(\mathbf{e}l^2 2^l)$, where $\mathbf{e}$ is the maximum time for an exponentiation in $\mathbb{G}$.*

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

For generators $h, g \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p^*$, let $y_i = g^{\alpha^i} \in \mathbb{G}$. As a beginning of the game, $\mathcal{B}$ starts a game with the $4l$-BDHE problem, to get a random tuple $(h, g, y_1, \dots, y_{4l}, y_{4l+2}, \dots, y_{8l}, T)$ which 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 $\mathbb{G}_1$). The goal of $\mathcal{B}$'s is to output 1 if the tuple is sampled from P_{BDHE}, or output 0 otherwise. $\mathcal{B}$ proceeds by interacting with $\mathcal{A}$ in a following selective subset game:

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 $g_2 = y_{4l}g^{\gamma} = g^{\gamma+(\alpha^{4l})}$. $\mathcal{B}$ 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 $\mathbb{Z}_p$, to set $h_{i,0} = g^{\gamma_{i,0}}/y_{l-i+1}, h_{i,1} = g^{\gamma_{i,1}}/y_{2l-i+1}, k_{i,0} = g^{\psi_{i,0}}/y_{3l-i+1}, k_{i,1} = g^{\psi_{i,1}}/y_{4l-i+1}$ for $i = 1, \ldots, l$. $\mathcal{B}$ 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 secret key corresponding to $ID = b_1 \ldots b_l \in \{0,1\}^l$. We now show that the revoked users can be simulated with two cases as follows. Case 1 shows how to simulate users that are not in the inclusion label c^* in subset (c^*, d^*). Case 2 shows that we can also simulate users included in the 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 to be the smallest of such index. To generate the secret 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 secret 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 secret key for the identity $ID = b_1 \ldots b_l$. $\mathcal{B}$ can compute all elements for the secret key, given the values at its disposal, considering 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 secret key, first observe that

$$\begin{aligned}
(h_{1,b_1} \cdots h_{l,b_l} k_{j,\bar{b}_j} g_3)^{r_j} &= (\prod_{i=1}^{l} h_{i,0}^{b_{i,0}} h_{i,1}^{b_{i,1}} \cdot k_{j,\bar{b}_j} \cdot g_3)^{r_j} \\
&= (\prod_{i=1}^{l} (\frac{g^{\gamma_{i,0}}}{y_{l-i+1}})^{b_{i,0}} (\frac{g^{\gamma_{i,1}}}{y_{2l-i+1}})^{b_{i,1}} \cdot \frac{g^{\psi_{j,\bar{b}_j}}}{y_{(3+\bar{b}_j)l-j+1}} \cdot g^{\delta} \prod_{i=1}^{l} y_{l-i+1}^{c_{i,0}^*} y_{2l-i+1}^{c_{i,1}^*} y_{3l-i+1}^{d_{i,0}^*} y_{4l-i+1}^{d_{i,1}^*})^{r_j} \\
&= (g^{\delta+\psi_{j,\bar{b}_j}+\Sigma_{i=1,i\neq k}^{l}(b_{i,0}\gamma_{i,0}+b_{i,1}\gamma_{i,1})} \prod_{i=1,i\neq k}^{l} (y_{l-i+1}^{c_{i,0}^*-b_{i,0}} y_{2l-i+1}^{c_{i,1}^*-b_{i,1}}) \cdot \prod_{i=1}^{l} (y_{3l-i+1}^{d_{i,0}^*} y_{4l-i+1}^{d_{i,1}^*}) y_{3l-j+1}^{-b_{j,0}} y_{4l-j+1}^{-b_{j,1}} y_{l-k+1}^{c_{k,0}^*-b_{k,0}} y_{2l-k+1}^{c_{k,1}^*-b_{k,1}})^{r_j} \\
&= (g^{\delta+\psi_{j,\bar{b}_j}+\Sigma_{i=1,i\neq k}^{l}(b_{i,0}\gamma_{i,0}+b_{i,1}\gamma_{i,1})} \prod_{i=1,i\neq k}^{l} (y_{l-i+1}^{c_{i,0}^*-b_{i,0}} y_{2l-i+1}^{c_{i,1}^*-b_{i,1}}) \cdot \prod_{i=1}^{l} (y_{3l-i+1}^{d_{i,0}^*} y_{4l-i+1}^{d_{i,1}^*}) y_{3l-j+1}^{-b_{j,0}} y_{4l-j+1}^{-b_{j,1}} y_{(1+\bar{b}_k)l-k+1}^{c_{k,\bar{b}_k}^*} y_{(1+b_k)l-k+1}^{-1})^{r_j}.
\end{aligned}$$

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

$$\begin{aligned}
Z^{r_j} =& (g^{\delta+\psi_{j,\bar{b}_j}+\Sigma_{i=1,i\neq k}^{l}(b_{i,0}\gamma_{i,0}+b_{i,1}\gamma_{i,1})} \prod_{i=1,i\neq k}^{l} (y_{l-i+1}^{c_{i,0}^*-b_{i,0}} y_{2l-i+1}^{c_{i,1}^*-b_{i,1}}) \cdot \prod_{i=1}^{l} (y_{3l-i+1}^{d_{i,0}^*} y_{4l-i+1}^{d_{i,1}^*}) y_{3l-j+1}^{-b_{j,0}} y_{4l-j+1}^{-b_{j,1}} y_{(1+\bar{b}_k)l-k+1}^{c_{k,\bar{b}_k}^*})^{r_j} \\
=& y_{(3-b_k)l+k}^{\delta+\psi_{j,\bar{b}_j}+\Sigma_{i=1,i\neq k}^{l}(b_{i,0}\gamma_{i,0}+b_{i,1}\gamma_{i,1})} \cdot \prod_{i=1,i\neq k}^{l} (y_{(4-b_k)l-i+k+1}^{c_{i,0}^*-b_{i,0}} y_{(5-b_k)l-i+k+1}^{c_{i,1}^*-b_{i,1}}) \cdot \prod_{i=1}^{l} (y_{(6-b_k)l-i+k+1}^{d_{i,0}^*} y_{(7-b_k)l-i+k+1}^{d_{i,1}^*}) \\
& \cdot y_{(6-b_k)l-j+1}^{-b_{j,0}} y_{(7-b_k)l-j+1}^{-b_{j,1}} y_{(4-b_k+\bar{b}_k)l+1}^{c_{k,\bar{b}_k}^*} Z^{\tilde{r}_j}.
\end{aligned}$$

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, $\mathcal{B}$ is able to compute all the terms in Z^{r_j}. Next, when observing the last term, $y_{(1+b_k)l-k+1}^{-r_j}$ 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 secret key is equal to:

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

Notice that the unknown term y_{4l+1} cancels out, which indicates that $\mathcal{B}$ can compute the first secret key component. The first component, g^{r_j}, is $y_{(3-b_k)l+k}g^{\tilde{r}_j}$ which can be computed by $\mathcal{B}$. In a similar manner, 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 secret key $SK_{ID,j}$ for identity $ID = b_1\ldots b_l$, $\mathcal{B}$ first selects 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 secret key $SK_{ID} = (SK_{ID,1},\cdots,SK_{ID,l})$ The secret key parameters are computed similarly. To generate the second component of the secret key, first observe that

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

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 $y_{(3+\bar{b}_j)l-j+1}^{-r_j}$, is:

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

Therefore, the first component in the secret key is equal to:

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

Notice that the unknown y_{4l+1} cancels out, which indicates that $\mathcal{B}$ can compute the first secret key component. The first component, g^{r_j}, is $y_{(1-\bar{b}_j)l+j}g^{\tilde{r}_j}$ which can be computed by $\mathcal{B}$. In a similar manner, $\mathcal{B}$ can compute the remaining elements $h_{1,\bar{b}_1}^{r_j},\cdots,$ $h_{l,\bar{b}_l}^{r_j},k_{1,0}^{r_j},k_{1,1}^{r_j},\ldots,k_{j-1,0}^{r_j},k_{j-1,1}^{r_j},k_{j,b_j}^{r_j},k_{j+1,0}^{r_j},k_{j+1,1}^{r_j},\ldots,k_{l,0}^{r_j},k_{l,1}^{r_j}$ since they do not involve y_{4l+1} as $h_{i,\bar{b}_i}^{r_j} = (\frac{g^{\gamma_{i,\bar{b}_i}}}{y_{(1+\bar{b}_i)l-i+1}})^{r_j} = (y_{(1-\bar{b}_j)l+j}\cdot g^{\tilde{r}_j})^{\gamma_{i,\bar{b}_i}}\cdot(y_{(2+\bar{b}_i-\bar{b}_j)l-i+j+1}y_{(1+\bar{b}_i)l-i+1}^{\tilde{r}_j})^{-1}$ and $k_{i,t}^{r_j} = (\frac{g^{\psi_{i,t}}}{y_{(3+t)l-i+1}})^{r_j} = (y_{(1-\bar{b}_j)l+j}\cdot g^{\tilde{r}_j})^{\psi_{i,t}}\cdot(y_{(4+t-\bar{b}_j)l-i+j+1}\cdot y_{(3+t)l-i+1}^{\tilde{r}_j})^{-1}$. Thus, $\mathcal{B}$ can derive a valid secret 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 $i = j$.

Therefore, $\mathcal{B}$ can derive a valid secret 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 $ID \npreceq c^*$ or $ID \preceq d^*$. All the values are within an independent and uniform distribution from $\mathbb{G}$, as required. The master key for the corresponding system is supposed to be $g_2^{\alpha} = g^{\alpha(\alpha^{4l}+\gamma)} = y_{4l+1}y_1^{\gamma}$, which is not known to $\mathcal{B}$ since $\mathcal{B}$ does not know y_{4l+1}.

For the challenge, $\mathcal{B}$ generates 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^*) to $\mathcal{A}$, as a challenge. If $T = e(g,h)^{4l+1}$ (i.e., the input to $\mathcal{B}$ is from the $4l$-BDHE) then the challenge (Hdr^*, K^*) is valid from A's view as in the real attacking game. This can be seen by writing $h = g^c$ for some (unknown) $c \in \mathbb{Z}_p$ as follows:

$$\begin{aligned}
&h^{\delta+\sum_{i=1}^{l}(\gamma_{i,0}c_{i,0}^*+\gamma_{i,1}c_{i,1}^*+\psi_{i,0}d_{i,0}^*+\psi_{i,1}d_{i,1}^*)} \\
&= (\prod_{i=1}^{l}(\frac{g^{\gamma_{i,0}}}{y_{l-i+1}})^{c_{i,0}^*}(\frac{g^{\gamma_{i,1}}}{y_{2l-i+1}})^{c_{i,1}^*}(\frac{g^{\psi_{i,0}}}{y_{3l-i+1}})^{d_{i,0}^*}(\frac{g^{\psi_{i,1}}}{y_{4l-i+1}})^{d_{i,1}^*} \cdot (g^{\delta}\prod_{i=1}^{l} y_{l-i+1}^{c_{i,0}^*}y_{2l-i+1}^{c_{i,1}^*}y_{3l-i+1}^{d_{i,0}^*}y_{4l-i+1}^{d_{i,1}^*}))^c \\
&= (h_{1,c_1^*}\cdots h_{l,c_l^*}k_{1,d_1^*}\cdots k_{l,d_l^*}g_3)^c
\end{aligned}$$

and

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

Thus, by definition, Hdr^* is a valid encryption of $e(y_{4l+1}, g)^c$. Also, observe that $e(y_{4l+1}, g)^c = e(g,h)^{4l+1} = T = K_b$, which confirms that (Hdr, K^*) is a valid challenge to the $\mathcal{A}$. On the other hand, if T is random in $\mathbb{G}$ (i.e., the input to $\mathcal{B}$ is a random tuple), then K^* is just a random independent key of $\mathcal{K}$ from $\mathcal{A}$'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 $\mathbb{G}_T$.

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

$$|Pr[B(g, h, \vec{y}_{g,\alpha,4l}, e(g,h)^{(\alpha^{4l+1})}) = 0] - Pr[B(g, h, \vec{y}_{g,\alpha,4l}, T) = 0]| \geq |(1/2 + \epsilon) - 1/2| = \epsilon. \quad (5)$$

as required, which completes the proof of the theorem. □

7. CCA-secure Broadcast Encryption

In this section, we extend our proposed CPA-secure combinatorial subset difference public key broadcast encryption scheme to obtain CCA-security by using similar technique from Reference [33] which transforms a CPA-secure identity-based encryption to the CCA-secure encryption. However, to apply the similar technique, it is required to allow wildcards ($*$) in the ID (as well as the subset) since the technique is based on the identity-based encryption. Therefore, we first construct a general ID construction, which is a generalized version of our CSD-based broadcast encryption which allows wildcards in the user ID in Section 7.1. Then, based on the general ID construction, we build a CCA-secure CSD-based public key broadcast encryption in Section 7.2. Finally, we provide a formal proof for the CCA-security in Section 8.

7.1. General ID Construction

In this section, we extend the proposed broadcast encryption scheme to a generalized ID version, which allows $*$ in each ID (as well as subsets). This general ID scheme is a basic building block for building a CCA-secure broadcast encryption in Section 7.

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 generation of public key is equivalent and the generation of secret key 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:

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

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

$$\begin{aligned} SK_{ID,i,0} =& (g^{r_{i,0}}, g_2^{\alpha}(h_{1,b_1} \cdots h_{i-1,b_{i-1}} h_{i+1,b_{i+1}} \cdots h_{l,b_l} k_{i,0} g_3)^{r_{i,0}}, \\ & h_{1,\bar{b}_1}^{r_{i,0}}, \dots, h_{i,0}^{r_{i,0}}, h_{i,1}^{r_{i,0}}, \dots, h_{l,\bar{b}_l}^{r_{i,0}}, k_{1,0}^{r_{i,0}}, k_{1,1}^{r_{i,0}}, \dots, k_{i-1,0}^{r_{i,0}}, k_{i-1,1}^{r_{i,0}}, k_{i,1}^{r_{i,0}}, k_{i+1,0}^{r_{i,0}}, k_{i+1,1}^{r_{i,0}}, \dots, k_{l,0}^{r_{i,0}}, k_{l,1}^{r_{i,0}}) \\ SK_{ID,i,1} =& (g^{r_{i,1}}, g_2^{\alpha}(h_{1,b_1} \cdots h_{i-1,b_{i-1}} h_{i+1,b_{i+1}} \cdots h_{l,b_l} k_{i,1} g_3)^{r_{i,1}}, h_{1,\bar{b}_1}^{r_{i,1}}, \dots, h_{i,0}^{r_{i,1}}, \\ & h_{i,1}^{r_{i,1}}, \dots, h_{l,\bar{b}_l}^{r_{i,1}}, k_{1,1}^{r_{i,1}}, k_{1,1}^{r_{i,1}}, \dots, k_{i-1,0}^{r_{i,1}}, k_{i-1,1}^{r_{i,1}}, k_{i,0}^{r_{i,1}}, k_{i+1,0}^{r_{i,1}}, k_{i+1,1}^{r_{i,1}}, \dots, k_{l,0}^{r_{i,1}}, k_{l,1}^{r_{i,1}}). \end{aligned} \tag{6}$$

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 secret key. To regenerate decrypt key K using the given header $Hdr = (A_0, A_1)$ and the secret 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

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

7.2. CCA-Secure Construction

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

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 $\Pi = (\text{Setup}, \text{Encrypt}, \text{Decrypt})$ secure against chosen-ciphertext attacks using the $(l + z)$-level $\Pi' = (\text{Setup}', \text{Encrypt}', \text{Decrypt}')$ 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 $ID||*^z$, that is, $SK'_{ID||0^z}, \cdots, SK'_{ID||1^z}$. 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 the signature verification key space as $\{0,1\}^z$. To obtain the public key and master secret key, run semantically secure broadcast encryption Π'.

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

To generate secret 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 receive 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).

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

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

In a similar way, if ID contains $*$ the extension in Section 7.1 is applied.

The correctness is straightforward from a similar calculation as 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.

8. CCA-Security Analysis

In this section, we formally prove the security against the chosen-ciphertext attack (CCA-security) of the CCA-secure CSD-based broadcst encryption in Section 7 assuming the presence of a one-time signature scheme.

Theorem 4. *Let $\mathbb{G}$ be a bilinear group of prime order p. For all positive integer l, the above public key broadcast encryption Π is $(t, \epsilon_1 + \epsilon_2, l, m, q_D)$ CCA-secure when assuming the public key broadcast encryption Π' is $(t', \epsilon_1, l+z, m, 0)$ semantically secure in $\mathbb{G}$ and the signature scheme is $(t'', \epsilon_2, z, 1)$ strongly and 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 $\mathcal{B}$, which has an advantage $|AdvBr_{\mathcal{B},\Pi'} - 1/2| > \epsilon_1$ 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 $SigKeyGen$ to get 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 manner 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:

$\mathcal{A}$ can adaptively issue decryption queries. The decryption query consists of (ID, S, Hdr), where $S \subseteq S^*$, $ID \in S$, and $Hdr = ((C_0, C_1), \sigma, V_{sig})$. For the query, $\mathcal{B}$ replies as follows:

1. Run *Verify* to check the validity of signature σ on (C_0, C_1) by using the verification key V_{sig}. If the signature is not valid, $\mathcal{B}$ replies with $\perp$.
2. If $V_{sig} = V^*_{sig}$, an event *forge* happens, $\mathcal{B}$ 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 $SK_{ID||V_{sig}}$, $\mathcal{B}$ 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:

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

$\mathcal{B}$ replies 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:

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

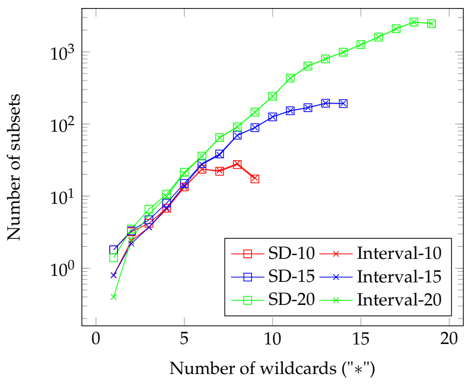
To conclude the proof, it is required to bound the probability of $\mathcal{B}$ aborting the simulation from *forge*. It can be seen that $Pr[forge] < \epsilon_2$; otherwise, the adversary $\mathcal{A}$ can be utilized to forge signatures with a probability of at least ϵ_2. In brief, we can construct another simulator which knows the secret key, but receives K^*_{sig} for the challenge in a forgery game. In the experiment above, $\mathcal{A}$ causes an abort by submitting a query which includes an existential forgery under K^*_{sig} on some ciphertexts. Our simulator can use $\mathcal{A}$ to win the existential forgery game. During the game, the adversary makes only a single chosen message query to generate the signature required for the challenge ciphertext. Thus, $Pr[forge] < \epsilon_2$, which concludes that $\mathcal{B}$'s advantage is at least ϵ_1 as required. □

9. Experiments

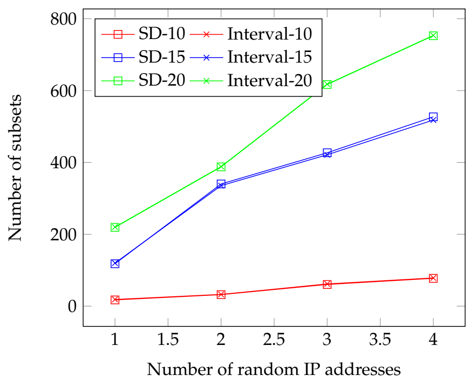
In this section, we show the experimental results on the proposed combinatorial subset difference (CSD), and compare the result to the other public key broadcast encryption. We implemented our CSD subset representation and CSD-based broadcast encryption, and also implemented the subset difference (SD) [7,8] and interval [10] representation along with its broadcast encryption on the Intel Edison IoT device with a 500 Mhz 32 bit Atom processor. Specifically, the SD-based broadcast encryption is implemented as a public key broadcast encryption by applying the public key lifting transformation [5], which combines the hierarchical identity-based encryption (HIBE) [6] to the original symmetric Naor SD construction [7,8] from the advanced access content system (AACS) DVD standard [9]. For the comparison, we first show the number of subsets generated from each algorithm to confirm the efficiency of CSD subset representation. We also verify that the CSD-based broadcast encryption achieves the minimal header size, due to the least number of subsets. Then for the broadcast encryption, we observe the practicality of our CSD-based broadcast encryption by comparing the key size and the performance (i.e., encryption time and decryption time) to the other constructions.

Subset Representation. The proposed CSD subset representation is the most general representation among the existing subset representations. Therefore, it can cover more users within a subset, which decreases the number of total subsets. When considering the IoT application, the improvement is more emphasized; the non-hierarchical IP address generates numerous subsets in the existing hierarchy-based representations (e.g., SD or interval), while the CSD can cover the non-hierarchical IP address in a single representation. For example, for a binary IP address $0 * *. * *1. * * * .001$, existing representation methods end up generating many subsets, while the CSD can efficiently cover it with a single subset.

Figure 6 shows the subset generations in the IP address application: Figure 6a presents the number of subsets when randomly increasing the number of wildcards in a single binary IP address, and Figure 6b presents the number of subsets when increasing the random IP addresses. The suffix of representations denotes a total bit of a user: for example, $SD - 10$ indicates an SD representation for 10-bit users, or 2^{10} total users. In both (a) and (b), the CSD representation can cover an IP address with a single subset regardless of the total users. When observing (a), the number of subsets in SD and interval representation when the IP address includes more wildcards or the total users increase, while the CSD can cover the address within a single subset in any case. When assuming a binary IP address with 20 wildcards in 2^{20} total users, the CSD can remain a single subset while the SD requires more than 1000 subsets; the CSD can save the header size 1000 times compared to the current standard SD representation. In (b), the SD and interval representation generates more subsets for more IP addresses or more total users, while the number of CSD subsets are same as the number of IP addresses.



(a)

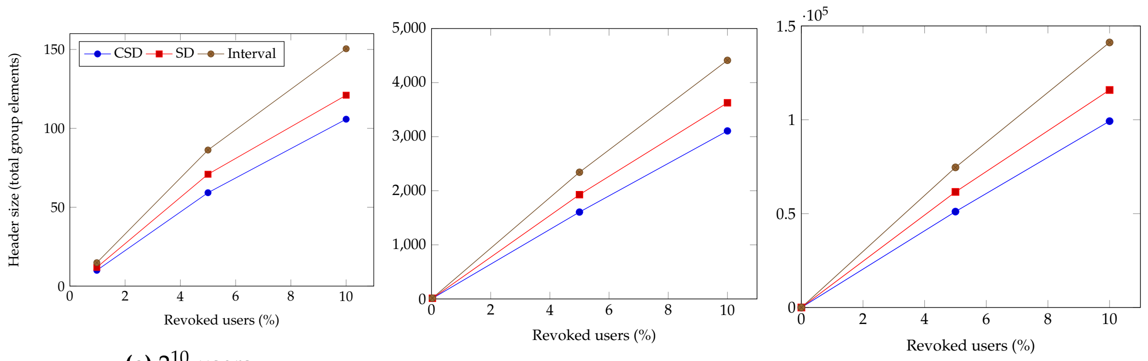


(b)

Figure 6. The number of subsets in IP address multicast: (**a**) for a single binary IP address (**b**) for multiple addresses.

Header Size. Since the CSD representation generates minimal subsets, the CSD-based broadcast encryption achieves minimal (total) header sizes when applied to the broadcast encryption. We verify the result by implementing and running the broadcast encryption constructions on the IoT machine, and measuring the size of the total header output size.

Figure 7 compares the header size of each broadcast encryption, by randomly revoking the users: by varying the total users to (a) 2^{10}, (b) 2^{15}, and (c) 2^{20}. The header size is measured by the number of group elements, where a single group element is a 20-bytes object from the pairing-based cryptography (PBC) library. In all cases, the CSD-based broadcast encryption outputs the smallest headers, compared to the SD-based and interval-based broadcast encryption.



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

Figure 7. Comparison of header sizes in broadcast encryption.

Key Size. The CSD-based broadcast encryption is also efficient in terms of key sizes; it maintains a public key and secret key size comparable to the standard broadcast encryption such as SD-based and interval-based broadcast encryption.

Table 3 shows the public key (PK) size and secret key (SK) size of the CSD-based, SD-based, and interval-based broadcast encryption. The public key size in CSD-based broadcast encryption is proportional to the user depth n, within the same order of $O(\log n)$ as the SD-based and interval-based construction. The secret key size is similar to the interval-based construction, within the same order of $O(\log^2 n)$ for user depth n.

Table 3. Comparison of key sizes in broadcast encryption.

	Depth	Public Key SD ([6,7])	Interval [10]	CSD (Ours)
PK (KB)	10-bit	0.25	0.49	0.83
	15-bit	0.37	0.73	1.24
	20-bit	0.50	0.98	1.65
SK (KB)	10-bit	19.61	4.03	5.92
	15-bit	66.18	9.07	13.32
	20-bit	156.88	16.13	23.69

Performance. For the performance of encryption and decryption, the CSD-based broadcast encryption improves the encryption and decryption time compared to the SD-based and interval-based broadcast encryption. The main factor is that our CSD-based construction does not involve any public parameter related computation, while the other constructions require additional delegation from the public parameters which is proportional to the user depth.

Table 4 shows the encryption time and decryption time of the CSD-based, SD-based, and interval-based broadcast encryption, by varying the user depth from 10-bits to 20-bits (total users of $2^{10}, 2^{15}, 2^{20}$). Our CSD-based broadcast encryption maintains almost the same performance for all cases, while the other constructions suffer from the increase of encryption and decryption time proportional to the user depth.

Table 4. Comparison of encryption time and decryption time (per subset) in broadcast encryption.

	Depth	Public Key SD ([6,7])	Interval [10]	CSD (Ours)
Enc (s)	10-bit	0.70	0.24	0.20
	15-bit	0.94	0.25	0.20
	20-bit	1.18	0.25	0.20
Dec (s)	10-bit	1.02	1.67	0.17
	15-bit	1.37	2.41	0.17
	20-bit	1.75	3.17	0.17

Open Source. The experiment resources are publicly available at GitHub as an open source (https://github.com/snp-lab/CSD), for the open science and reproducible research. The resources consist of *subset construction* algorithm for each combinatorial subset difference (CSD), subset difference (SD), and interval, and *broadcast encryption* implementation for CSD-based, SD-based, and interval-based scheme.

10. Conclusions

We propose the combinatorial subset difference (CSD), which is a new subset representation for the broadcast encryption. The CSD representation is the most general representation compared to state-of-the-arts such as subset difference (SD) or interval representations. Due to the general coverage, the CSD representation generates minimal subsets which lead to the minimal header size when applied to the broadcast encryption. The CSD representation is the only representation to cover the non-hierarchical IP addresses, making it as a suitable choice for IoT network multicast. We design and implement the algorithm for CSD subset generation, and verify the practicality of our CSD in the IoT systems.

We also propose the CSD-based public key broadcast encryption, which can be efficiently applied to the IP multicast in IoT systems. The CSD-based broadcast encryption achieves minimal (total) header size due to the minimal subsets of the CSD representation, and it also improves the encryption time and decryption time compared to the existing broadcast encryption. Our experimental results verify that the CSD-based broadcast encryption reduces the header size 31% for the worst cases, improves encryption time by 6 times, and improves decryption time by 10 times compared to the SD-based broadcast encryption. For the security, we prove the collusion-resistance and semantic security of the CSD-based construction under the standard l-BDHE assumption, and extend the construction to the CCA-secure broadcast encryption.

Author Contributions: Conceptualization, J.K. and H.O.; methodology, J.K. and H.O.; software, J.L. and S.L.; validation, J.L. and S.L.; writing–original draft preparation, J.L. and S.L.; writing–review and editing, J.K., J.L., S.L., and H.O. All authors have read and agree to the published version of the manuscript.

Funding: This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Ministry of Science and ICT Korea (2017-0-00661, 2016-6-00599), and a Semiconductor Industry Collaborative Project between Hanyang University and Samsung Electronics Co. Ltd.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yao, X.; Chen, Z.; Tian, Y. A lightweight attribute-based encryption scheme for the Internet of Things. *Future Gener. Comput. Syst.* **2015**, *49*, 104–112. [CrossRef]
2. Belguith, S.; Kaaniche, N.; Laurent, M.; Jemai, A.; Attia, R. Phoabe: Securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted iot. *Comput. Netw.* **2018**, *133*, 141–156. [CrossRef]
3. Kim, J.Y.; Hu, W.; Sarkar, D.; Jha, S. ESIoT: Enabling secure management of the internet of things. In Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, Boston, MA, USA, 18–20 July 2017; pp. 219–229.

4. Mao, Y.; Li, J.; Chen, M.R.; Liu, J.; Xie, C.; Zhan, Y. Fully secure fuzzy identity-based encryption for secure IoT communications. *Comput. Stand. Interfaces* **2016**, *44*, 117–121. [CrossRef]
5. Dodis, Y.; Fazio, N. Public Key Broadcast Encryption for Stateless Receivers. *Lect. Notes Comput. Sci.* **2002**, *2696*, 61–80.
6. Boneh, D.; Boyen, X.; Goh, E. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In Proceedings of the Advances in Cryptology—EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, 22–26 May 2005; pp. 440–456.
7. Bhattacherjee, S.; Sarkar, P. Complete tree subset difference broadcast encryption scheme and its analysis. *Des. Codes Cryptogr.* **2013**, *66*, 335–362. [CrossRef]
8. Naor, D.; Naor, M.; Lotspiech, J. Revocation and Tracing Schemes for Stateless Receivers. *Lect. Notes Comput. Sci.* **2001**, *2139* 41–62.
9. Henry, K.; Sui, J.; Zhong, G. An Overview of the Advanced Access Content System (AACS). *Cent. Appl. Cryptogr. Res. (Cacr)* **2007**, *25*, 1–24.
10. Lin, H.; Cao, Z.; Liang, X.; Zhou, M.; Zhu, H.; Xing, D. How to construct interval encryption from binary tree encryption. In Proceedings of the International Conference on Applied Cryptography and Network Security, Bogota, Colombia, 5–7 June 2019; pp. 19–34.
11. Lee, K.; Park, J.H. Identity-Based Revocation from Subset Difference Methods under Simple Assumptions. *IEEE Access* **2019**, *7*, 60333–60347. [CrossRef]
12. Chu, H.; Qiao, L.; Nahrstedt, K.; Wang, H.; Jain, R. A secure multicast protocol with copyright protection. *Acm Sigcomm Comput. Commun. Rev.* **2002**, *32*, 42–60. [CrossRef]
13. Kumar, V.; Kumar, R.; Pandey, S. A computationally efficient centralized group key distribution protocol for secure multicast communications based upon RSA public key cryptosystem. *J. King Saud-Univ.-Comput. Inf. Sci.* **2018**, in press. [CrossRef]
14. Wang, X.; Zhu, H. A Novel Two-party Key Agreement Protocol with the Environment of Wearable device using Chaotic maps. *DSPR* **2019**, *3* 12–23.
15. Naor, M.; Pinkas, B. Efficient trace and revoke schemes. *Int. J. Inf. Sec.* **2010**, *9*, 411–424. doi:10.1007/s10207-010-0121-2. [CrossRef]
16. Boneh, D.; Waters, B. A fully collusion resistant broadcast, trace, and revoke system. In Proceedings of the 13th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 30 October–3 November 2006; pp. 211–220.
17. Delerablée, C. Identity-based broadcast encryption with constant size ciphertexts and private keys. In Proceedings of the ASIACRYPT: International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, 2–6 December 2007; pp. 200–215.
18. Boneh, D.; Hamburg, M. Generalized identity based and broadcast encryption schemes. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security. Melbourne, VIC, Australia, 7–11 December 2008; pp. 455–470.
19. Boneh, D.; Sahai, A.; Waters, B. Fully collusion resistant traitor tracing with short ciphertexts and private keys. *Adv. Cryptol. Eurocrypt* **2006**, *2006*, 573–592.
20. Gentry, C.; Waters, B. *Advances in Cryptology—EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, 26–30 April 2009*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 171–188.
21. Yao, D.; Fazio, N.; Dodis, Y.; Lysyanskaya, A. 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, New York, NY, USA, 25–29 October 2004; pp. 354–363.
22. Canetti, R.; Halevi, S.; Katz, J. Chosen-Ciphertext Security from Identity-Based Encryption. In Proceedings of the Advances in Cryptology—EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, 2–6 May 2004; pp. 207–222.
23. Boneh, D.; Gentry, C.; Waters, B. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. *Lect. Notes Comput. Sci.* **2005**, *3621*, 258–275.
24. Fiat, A.; Naor, M. Broadcast Encryption. *Lect. Notes Comput. Sci.* **1993**, *773*, 480–491.
25. Goodrich, M.T.; Sun, J.Z.; Tamassia, R. Efficient Tree-Based Revocation in Groups of Low-State Devices. *Lect. Notes Comput. Sci.* **2004**, *3152*, 511–527.

26. Halevy, D.; Shamir, A. The LSD Broadcast Encryption Scheme. *Lect. Notes Comput. Sci.* **2002**, *2442*, 47–60.
27. Wallner, D.M.; Harder, E.J.; Agee, R.C. Key Management for Multicast: Issues and Architectures. Available online: https://www.hjp.at/(de,st_b)/doc/rfc/rfc2627.html (accessed on 2 June 2020).
28. Canetti, R.; Garay, J.A.; Itkis, G.; Micciancio, D.; Naor, M.; Pinkas, B. Multicast Security: A Taxonomy and Some Efficient Constructions. *Proc. IEEE* **1999**, *2*, 708–716.
29. Cheon, J.H.; Jho, N.; Kim, M.; Yoo, E.S. Skipping, Cascade, and Combined Chain Schemes for Broadcast Encryption. *IEEE Trans. Inf. Theory* **2008**, *54*, 5155–5171. doi:10.1109/TIT.2008.928959. [CrossRef]
30. Lee, K.; Koo, W.K.; Lee, D.H.; Park, J.H. Public-key revocation and tracing schemes with subset difference methods revisited. *Eur. Symp. Res. Comput. Secur.* **2014**, *8713*, 1–18.
31. Joux, A. Multicollisions in iterated hash functions. Application to cascaded constructions. *Annu. Int. Cryptol. Conf.* **2004**, *3152*, 306–316.
32. Boneh, D.; Franklin, M. Identity-based encryption from the Weil pairing. *Siam J. Comput.* **2003**, *32*, 586–615. [CrossRef]
33. Boneh, D.; Canetti, R.; Halevi, S.; Katz, J. Chosen-Ciphertext Security from Identity-Based Encryption. *SIAM J. Comput.* **2007**, *36*, 1301–1328. doi:10.1137/S009753970544713X. [CrossRef]