

# BESTIE: Broadcast Encryption Scheme for Tiny IoT Equipments

Jiwon Lee<sup>1</sup>, Jihye Kim<sup>2</sup>, and Hyunok Oh<sup>1</sup>

<sup>1</sup> Hanyang University, Seoul, Korea,  
{jiwonlee,hoh}@hanyang.ac.kr

<sup>2</sup> Kookmin University, Seoul, Korea,  
jihyek@kookmin.ac.kr

**Abstract.** In public key broadcast encryption, anyone can securely transmit a message to a group of receivers such that privileged users can decrypt it. The three important parameters of the broadcast encryption scheme are the length of the ciphertext, the size of private/public key, and the performance of encryption/decryption. It is suggested to decrease them as much as possible, however, it turns out that decreasing one increases the other in most schemes.

This paper proposes a new broadcast encryption scheme for tiny IoT equipments (BESTIE), minimizing the private key size in each user. In the proposed scheme, the private key size is  $O(\log n)$ , the public key size is  $O(\log n)$ , the encryption time per subset is  $O(\log n)$ , the decryption time is  $O(\log n)$ , and the ciphertext text size is  $O(r)$ , where  $n$  denotes the maximum number of users and  $r$  indicates the number of revoked users. The proposed scheme is the first subset difference based broadcast encryption scheme to reduce the private size  $O(\log n)$  without sacrificing the other parameters. We prove that our proposed scheme is secure under  $q$ -Simplified Multi-Exponent Bilinear Diffie-Hellman ( $q$ -SMEBDH) in the standard model.

**Keywords:** Broadcast encryption, public key encryption, subset difference, short key

## 1 Introduction

In a modern IoT infrastructure, the number of total devices tend to increase on a large scale, while the size of individual equipments become smaller. When dealing with secure communications for a massive number of resource-constrained devices, it is important not only to support flexible access control, but also to minimize transmission costs and device computation/storage overhead. Broadcast encryption is the fundamental cryptographic primitive to uphold secure communication to any group of privileged devices.

**Broadcast Encryption.** In the Broadcast encryption (BE) scheme, anyone can securely transmit a message to a group of receivers such that privileged users can

decrypt it. In BE protocol, the transmission consists of  $(S, Hdr, C_K)$  where  $S$  is a group (subset) of users,  $Hdr$  is a header which contains essential information for the session key  $K$ , and  $C_K$  is a symmetric key encryption (e.g. AES) of the message with using  $K$ . When receiving the following transmission, a user first extracts (or decrypts) the session key  $K$  from  $Hdr$ , then he/she uses the following symmetric key  $K$  for the decryption of  $C_K$ . If the user is not covered in  $S$ , this indicates that the user is revoked and should not be able to extract the key from  $Hdr$ . Moreover, the system should guarantee that even if all the revoked users collude, it should be impossible to learn any information about the  $K$  in the  $Hdr$ . The header is considered a real ciphertext in a BE field of research, since it holds the security of transmissions.

In BE systems, the main competitive issue was reducing the number of subsets to cut down the ciphertext header size. The subset group  $S$  works as an encryption unit in most BE schemes and privileged users are determined by multiple subsets. In this case, the header should include all of the corresponding encryptions of  $K$ . To be more concrete, suppose we have subsets of  $S_1, \dots, S_n$ , then the header  $\{Hdr\}$  is a vector that consists of  $Hdr_1, \dots, Hdr_n$ . Namely, the header size is strictly linear to the number of subsets which clarifies that the number of subsets needs to be minimized.

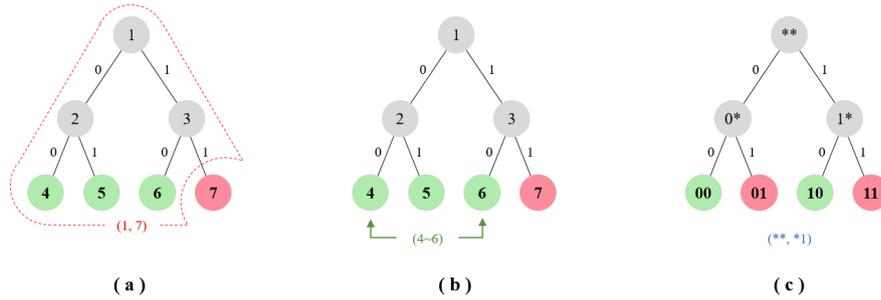


Fig. 1: Subset construction examples in (a) SD, (b) interval, and (c) CSD representations

Many schemes have been proposed [NP01, LCL<sup>+</sup>10, KLLO17, DF02, BGW05] in different representations with the purpose of reducing the number of subsets, i.e., the header size in BE. In particular, subset difference schemes [NP01, LKLP14] have been received a lot of attention and adopted practically from DVD and Blu-ray disc standards (AACS) to Pay-TV systems since SD schemes provide appropriate parameters of key size, execution time, and ciphertext size. Hence, this paper concentrates on the SD based approach.

By varying the SD framework, Lin's group proposed an interval coverage [LCL<sup>+</sup>10] which achieves a comparable header size to the SD approach. Moreover, Kim et

al. [KLLO17] devised a combinatorial subset difference (CSD), which extends the SD to be more general and expressive. Figure 1 shows an example to visualize each representation. The SD represents a subset with a subtraction (difference) of two subtrees, which is from a binary tree constructed with users mapped as leaf nodes. As an example in figure 1(a), the SD representation  $(1, 7)$  covers privileged users 4, 5, 6. The interval representation lets a subset denote a range of privileged users. In the example of figure 1(b),  $(4, 6)$  can cover the privileged users 4, 5, 6. The SD representation is likely to cover more users since subtrees provide more flexible depth compared to the fixed range of interval representation. However, SD has a limitation: it is bound to the tree hierarchy. When converting the binary tree to the bitwise representation  $(0, 1, *)$  by translating the left edge as 0 and the right edge as 1 (figure 1(c)), the wildcard  $(*)$  cannot be placed before the bit. The reason for this is that each bit is decided from top to bottom due to the hierarchy, thus an unfixed bit  $(*)$  can only exist when its parent is fixed. The SD and interval schemes, therefore, show analogous results in terms of header sizes; the SD scheme shows a header size of  $4r$  and the interval scheme shows a header size of  $3r$  in the worst case, where  $r$  is the number of revoked users. Note that it is hard to decide which scheme has a smaller header size, if not fixed in the worst cases.

Similar to SD, the CSD also represents a subset with a subtraction of two sets, but each subset is no longer a subtree; it is a label of binary bits which is a generalized expansion of the subset difference. In the example of figure 1(c), the CSD subset  $(**, *1)$  can cover the privileged users 00, 10. Note that the representation  $*1$  is impossible to visualize in the tree figure, since the tree is bound to the hierarchy. The CSD has removed the limitation of hierarchy that lies in the subset difference, and it can cover both the SD and the interval representation with a bit label. It is the most generalized form of subset construction that can cover all existing representations. The CSD cuts the header size down to  $2r$  even in the worst case, and shows that it can always cover users with less (or at least the same) subsets than SD representations.

The public key broadcast encryption scheme for the CSD representation [KLLO17] has shown that BE can be applied efficiently to the secure multicast in IoT systems. Since the CSD can represent the generalized binary bits, it can cover the bit string of IP addresses for devices in an IoT system network. The experiments in [KLLO17] show that the CSD scheme is practical and appropriate for IoT multicasts within a large scale of devices.

**BE for Tiny Equipments.** While the number of IoT equipments increases, the size of the equipment itself decreases. In current IoT infrastructures, devices are likely to have no more than a few kilobytes of secure on-chip storage. Note that the key should not be stored in the off-chip flash storage (which could be larger), since they are exposed to the public and commonly extractable [FGL<sup>+</sup>17]. In this setting, the keys of BE should be short enough to be stored in the small-sized memory of tiny IoT equipments. To justify the usage of BE in various IoT systems with tiny devices, we list some specific application examples:

- **Secure multicast:** The research of [KLLO17] already justified the usage of BE for secure multicast. To be more specific, an IoT system manager may want to broadcast and distribute secure messages to the devices by using the subset difference of IPv6 address bit string. Current IoT equipments commonly utilize chips that have 4KB to 128KB of non-volatile memory (EEPROM or on-chip flash). Some devices tend to use trusted platform module (TPM) chips that can store and manage keys securely, and the TPM key storage also has a size of no more than 16KB<sup>3</sup>.
- **ECU firmware management:** The engine control unit (ECU) of a vehicle is known to have a key storage for its code and data encryption. In time-to-time firmware updates, the system needs to set privileged devices either to guarantee customized firmwares for different vehicles or to revoke the disclosed keys that are often used by other vendors. BE can provide an appropriate environment for the large scale of ECU firmware encryption management. The non-volatile on-chip memory of the ECU usually has a size of no more than 12KB.

Unfortunately none of the existing BE schemes are capable of satisfying the requirement of small sized keys in a setting with a massive number of devices. There were some noticeable works that focus on the key size of BE, like [LKLP14]. In [LKLP14], the authors proposed a scheme that reduces the private key (SK) size from  $O(\log^3 n)$  to  $O(\log^2 n)$ , compared to the original *SD* based schemes [DF02]. Interval scheme [LCL<sup>+</sup>10] also shows a same order of  $O(\log^2 n)$  for the SK size, while maintaining the same transmission complexity as [DF02, LKLP14]. The size complexity  $O(\log^2 n)$ , however, is not small enough to be practically applied for tiny IoT devices. In the secure multicast example, the current IPv6 standard considers  $2^{128}$  users. Therefore in the secure multicast for random devices, the system should provide a full spectrum of representations for the 128-bit address combinations. The ECU firmware case is also similar; the vehicle ECU has its own ID which usually consists of distinct 32 to 128 bits [SHO08]. This leads the private key to grow larger than 40KB for 32 bits or 640KB for 128 bits, which cannot be stored in the small on-chip storage of 12KB in ECU. In fact, it is an open problem to reduce the private key size to  $O(\log n)$  in the *SD* based BE approach.<sup>4</sup>

**BESTIE with Short-Key.** In this paper, we propose BESTIE a new broadcast encryption scheme which has a short key size for tiny IoT equipments. The proposed BESTIE has a key size of  $O(\log n)$ , which exceeds the current boundary of the key size  $O(\log^2 n)$  among the existing subset difference based broadcast encryptions. By applying BESTIE in the 128-bit ID systems, we can obtain a 7KB private key (SK) which can be easily stored in the secure on-chip memory of

<sup>3</sup> ATmega 128 microprocessor has 128KB flash and 4KB EEPROM, and Atmel TPM series provide 16KB of non-volatile key storage [ZQY08].

<sup>4</sup> This should be achieved with a reasonable cost; Goodrich [GST04] proposed a symmetric BE with the SK size of  $O(\log n)$ , but the computation cost is  $O(n)$  which is beyond practical (see table 1 for details).

tiny IoT equipments. Moreover, the BESTIE *does not sacrifice any other factor* such as execution (encryption/decryption) times or header sizes.

The main idea to reduce the key size in the proposed scheme is to share the same random value for every key, while different random values are applied for each key in the existing subset difference based approaches. In the original CSD scheme [KLLO17] as well as most subset difference based schemes,  $O(l)$  size key is required for each bit in the ID. I.e., given a private key  $SK = (SK_{ID_1}, \dots, SK_{ID_l})$ , each element  $SK_{ID_i}$  contains a primary key, and  $O(l)$  size auxiliary key for the other bit positions to build a decryption key, where  $l$  denotes the bit-length or  $\log n$ . In the existing schemes, each auxiliary key should contain an independent random value; otherwise combination of keys may generate an unauthorized decryption key. The proposed BESTIE devises a novel and secure way to reuse the  $O(l)$  auxiliary key for all primary keys. As a result, the BESTIE requires  $O(\log n)$  size key. The detailed construction is available in section 4.

Another interesting feature is that, unlike most existing schemes, BESTIE does not demand a public key (PK) for the decryption. Other schemes such as SD [LKLP14] or interval schemes [LCL<sup>+</sup>10] reconstruct the corresponding decryption key from the PK as well as the SK in the decryption phase, and thus need to maintain the PK in the device storage or receive the PK from the communication. On the other hand, since the decryption in BESTIE relies on the computation with the secret key SK only, there is no need to store the PK in the device. This indicates that BESTIE has an advantage in the PK storage and/or PK transmission overhead.

**Contributions.** We formally summarize the contributions of our BESTIE as follows:

1. **Theoretical advance:** The proposed BESTIE resolves a challenging problem to reduce the private key size to  $O(\log n)$  in the SD based BE approach, without sacrificing any other efficiency. Moreover, BESTIE is compatible with even CSD which is more expressive, and thus, more compact than SD.
2. **Practicality:** The BESTIE is applicable to large scale IoT systems ( $2^{128}$  devices) with a reasonable performance; it requires only 7KB private key size while the private key size is more than 600KB in the other existing SD based approaches.
3. **Security:** We prove that the BESTIE is collusion resistant and IND-sID-CPA-secure under the  $l$ -Simplified Multi-Exponent Bilinear Diffie-Hellman ( $l$ -SMEBDH) assumption (without the random oracle model). We also provide an IND-sID-CCA-secure version of the scheme.

Section 2 organizes related works. Section 3 describes a required background and definitions. We present the construction and the security proof of our proposed BESTIE in section 4, and extend it to the CCA-secure scheme in section 5. Section 6 analyzes experimental results. In section 7, we draw a conclusion.

## 2 Related Work

The broadcast encryption (BE) is a traditional cryptographic method, and there have been a variety of researches with different features [NP01, NP10, BW06, Del07, BH08, BSW06, GW09, YFDL04, CHK04, BBG05, BGW05, FN93, GST04, HS02, NNL01, DF02, WHA99, CGI<sup>+</sup>99b, CJKY08]. Also known as a revocation schemes, BE can provide efficient revocation of individual users, while letting the privileged users remain available to decrypt the broadcasted transmission. The listed categories below are the various viewpoints in BE, and every existing BE scheme has its own feature due to the different purposes.

- **Stateful vs. stateless:** There are two types of BE schemes, which are stateful schemes [CMN99, CGI<sup>+</sup>99a, SM03] and stateless schemes [NP01, HS02, GST04]. In the stateful BE scheme, the key exchange occurs more than once. On the other hand, the stateless BE scheme allows the key exchange only once in the initial setup. Stateful schemes can be useful in a setting that can allow users to interact after the initial setup. However, in real practice such as Pay-TV systems or IoT networks, once the devices are deployed it becomes a big burden to update all keys synchronously.
- **Symmetric vs. asymmetric:** The BE can be also categorized as a symmetric BE [GST04, CJKY08] and asymmetric BE. In the symmetric BE, only a trusted user who has a symmetric key can encrypt and broadcast the message to the receivers. An asymmetric BE, known as a public key broadcast encryption (PKBE), enables any user to broadcast the encrypted information.

It is clear that a stateful BE scheme and a symmetric BE scheme have more limitations in terms of its usage; this paper proposes a stateless public key (asymmetric) BE scheme. For a more practical use, most optimizations of BE schemes are focused on reducing header or key sizes.

- **Header size:** The main objective of the BE research was to reduce the header size, which decides the transmission overhead. Since the header size relies on the number of subsets, there were many works that proposed subset construction/representation methods [NNL01, GST04, HS02, LCL<sup>+</sup>10]. The most common representations were the complete subtree (CS) [NNL01], the subset difference (SD) [NNL01], and the interval encryption [LCL<sup>+</sup>10]. The CS method covers users with root nodes of subtrees. The SD method covers users with a subtraction of two subtrees. The interval encryption covers users with ranges of privileged users. Recently, the work of [KLLO17] proposed a combinatorial subset difference (CSD), which covers users with a subtraction of two non-hierarchical bit-labels. The SD scheme has a header size of  $4r$ , the interval scheme has a header size of  $3r$ , and the CSD scheme has a header size of  $2r$ , each for the worst cases when  $r$  is the number of revoked users.
- **Key size:** Some works have focused on reducing the PK/SK size of BE, although there usually is a trade-off between the size and the encryption/decryption

time. The work of [LKL14] has succeeded on reducing the SK size to the order of  $O(\log^2 n)$  in the subset difference. The interval scheme also obtained an order of  $O(\log^2 n)$  for the SK size. Until now, even a symmetric key BE (which is limited, but more generally efficient) has a boundary of  $O(\log^2 n)$  for the size of SK.

**SD-based BE.** Among the existing BE schemes, our main focus is on the SD-based methods (e.g. SD, interval, CSD), which achieves the header size of  $O(r)$ . In methods that do not use SD, the header size is impractical since it depends on the number of total users  $n$  instead of the number of revocation  $r$ . For instance, Boneh et al. [BGW05] proposed a notable scheme which covers the users as groups of indices; their general construction gains the header size of  $O(\sqrt{n})$  (PK size:  $O(\sqrt{n})$ , SK size:  $O(1)$ , encryption time:  $O(n)$ , decryption time:  $O(\sqrt{n})$ ). However, in general practice, the revocation tends to remain small while the total user grows large in various applications: the number of revocation  $r$  is much smaller than the  $\sqrt{n}$ .

Table 1: Comparison of costs between SD-based public-key BE. *ref.  $n =$  the number of total users, and  $r =$  the number of revoked users*

	<b>BESTIE (ours)</b>	<b>LKLP'14 [LKLP14]</b>	<b>Lin'10 [LCL<sup>+</sup>10]</b>	<b>DF'02 [DF02]</b>	<b>NNL'01 [NNL01]</b>	<b>GST'04 [GST04]</b>
<b>PK Size</b>	$O(\log n)$	$O(1)$	$O(\log n)$	$O(\log n)$	N/A	N/A
<b>SK Size</b>	$O(\log n)$	$O(\log^2 n)$	$O(\log^2 n)$	$O(\log^3 n)$	$O(\log^2 n)$	$O(\log n)$
<b>CT Size</b>	$O(r)$	$O(r)$	$O(r)$	$O(r)$	$O(r)$	$O(kr)$
<b>Enc Time</b>	$O(r)$	$O(r)$	$O(r)$	$O(r \log n)$	$O(r \log n)$	$O(kr \log(n/k))$
<b>Dec Time</b>	$O(\log n)$	$O(1)$	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(n^{1/k})$
<b>Enc type</b>	Asymmetric	Asymmetric	Asymmetric	Asymmetric	Symmetric	Symmetric
<b>Assumption</b>	$q - SMEBDH$	$q - SMEBDH$	$q - BDHE$	$q - BDHI$	One-way func.	One-way func.
<b>ROM</b>	No	Yes	No	No	No	No

Table 1 shows the order of costs in SD-based BE schemes that achieve the header (CT) size of  $O(r)$ , where  $n$  is the number of total users and  $r$  is the number of revoked users. NNL'01 [NNL01] is the original SD scheme which is a symmetric key BE. DF'02 [DF02] proposed a transformation technique that converts a symmetric key BE to public key BE by utilizing hierarchical identity-based encryption (HIBE); the shown results are obtained by applying the BBG-HIBE [BBG05] scheme to the NNL'01<sup>5</sup>. Lin'10 [LCL<sup>+</sup>10] refers to the interval encryption, which is similar to the SD in a way that the users are represented in a binary tree; the secret key size is  $O(\log^2 n)$ . LKLP'14 [LKLP14] proposes a

<sup>5</sup> DF'02 [DF02] states  $O(1)$  PK size and  $O(\log^2 n)$  SK size, but it refers to the HIBE keys; remind that the BBG-HIBE key has  $O(\log n)$  elements.

more efficient SD scheme with utilizing the random oracle, which also achieves the secret key size of  $O(\log^2 n)$ . GST'04 [GST04] is a symmetric key BE which focuses on the  $O(\log n)$  SK size. However, it sacrifices the decryption time to  $O(n)$ , or increase CT size and encryption time by a given constant  $k$  to mitigate the decryption time. Compared with all existing SD-based BE schemes, BESTIE is the first approach to obtain a  $O(\log n)$  SK size while providing overall decent performance. Moreover, since BESTIE does not sacrifice any other factors, it retains a small CT size, a small PK size, and fast encryption/decryption performance.

**Attribute Based Encryption.** From a high-level perspective, BE can be considered as a special case of attribute based encryption (ABE) [GPSW06, EMN<sup>+</sup>09, ZH10, AHL<sup>+</sup>12]; if we let each bitwise ID be an attribute and define subset inclusion as an access policy of ciphertext-policy ABE (CP-ABE), it can provide the same functionality of BE. However, as most general cases are not as efficient as special cases, ABE cannot achieve time and size costs comparable to BE. For instance, in the CP-ABE with constant-size ciphertext [AHL<sup>+</sup>12], the key size grows linear to the number of attributes. Since the access policy requires a bitwise representation of the ID and subsets, the key size roughly grows to  $O(2^n)$  which is beyond practical.

### 3 Preliminaries

In this section, we provide backgrounds and preliminary definitions. Subsection 3.1 describes the basic definition of public key broadcast encryption (PKBE). Subsection 3.2 defines the formal security model for our proposed system. Subsection 3.3 gives a remark for the mathematical background about bilinear maps and pairings in elliptic curve groups. In subsection 3.4, we describe the cryptographic assumption which our system is based on. Subsection 3.5 gives a summary for the combinatorial subset difference, which is a subset cover method our system adopts.

#### 3.1 Public Key Broadcast Encryption

In a public key broadcast encryption (PKBE) system, the original message  $m$  is commonly encrypted to  $C_K$ , which is often called the broadcast body, with a simple symmetric key algorithm (e.g. AES block cipher). Then the symmetric key  $M$  is encrypted with the PKBE encryption, so that the legitimate receivers can obtain the symmetric key  $M$  and use it for the symmetric decryption of  $C_K$  to obtain  $m$ . In the following decryption of BE systems, the symmetric key  $M$  is considered as a message; the symmetric key encryption/decryption process (i.e.  $m, C_K$ ) is common and often omitted in BE schemes.

The PKBE encryption is required for each subset, and the header (or the broadcast ciphertext)  $Hdr$  for each subset is collected into a vector  $\{Hdr\} = \{(S_i, Hdr_{S_i})\}_{i=1}^w$  where  $w$  is the number of total subsets. A legitimate user decrypts the message by looking for the  $Hdr_{S_i}$  corresponding to the subset  $S_i$

where it belongs to, obtaining  $M$  from  $Hdr_{S_i}$  with the PKBE decryption, and finally decrypting the message  $m$  from the broadcast body  $C_K$ . Formally, a PKBE system  $\Pi$  consists of four algorithms:

- Setup**( $l, \lambda$ ) takes user's ID bit-length  $l$  and session key length  $\lambda$  as inputs. It outputs public parameters  $PK$  and a master key  $MK$ .
- KeyGen**( $ID, MK, PK$ ) takes user's  $l$ -bit  $ID$ , master key  $MK$ , and public key  $PK$  as inputs. It outputs a private key set  $SK_{ID}$ .
- Encrypt**( $S, PK, M$ ) takes a subset  $S$ , and a public key  $PK$  and a message  $M$  as inputs. It outputs a broadcast ciphertext  $Hdr_S$  for the subset  $S$ .
- Decrypt**( $S, ID, SK_{ID}, Hdr_S$ ) takes a subset  $S$ , a user id  $ID \in \{0, 1\}^l$ , private key  $SK_{ID}$  for user  $ID$ , and a header  $Hdr_S$  as inputs. If  $ID \in S$ , then it outputs message  $M$ .

The system is correct if every user in  $S$  can get the message  $M$ . Namely, for all  $S$  and all  $ID \in S$ , if  $(PK, MK) \leftarrow \text{Setup}(l, \lambda)$ ,  $SK_{ID} \leftarrow \text{KeyGen}(ID, MK, PK)$ , and  $Hdr_S \leftarrow \text{Encrypt}(S, PK, M)$  then  $\text{Decrypt}(S, ID, SK_{ID}, Hdr_S)$  extracts  $M$ .

### 3.2 Security Model

In this section, we describe a selective semantic security (IND-sID-CPA) and a selective CCA-security (IND-sID-CCA) for broadcast encryption as in [BGW05, KLLO17]. Depending on whether the number of challenged sets is represented as a single subset or as multiple subsets, we separate security notions as a single-set security and a multi-set security. Consequently, the single-set security implies a multi-set security as shown in [KLLO17].

The single-set security is defined as a following game between an adversary  $\mathcal{A}$  and a challenger  $\mathcal{C}$ . Both  $\mathcal{C}$  and  $\mathcal{A}$  are given  $l$  and  $\lambda$ , the user ID length and the key length respectively, as inputs. Note that the collusion resistance is straightforward, since the secret keys for all users (except the selected target) are distributed before the challenge.

- Init:** Algorithm  $\mathcal{A}$  outputs a set  $S^*$  of users to attack.
- Setup:** The challenger  $\mathcal{C}$  performs  $\text{Setup}(l, \lambda)$  to obtain a public key  $PK$  and a master key  $MK$ .
- KeyGen:** The challenger  $\mathcal{C}$  runs  $\text{KeyGen}(ID, MK, PK)$  to obtain private keys  $SK_{0^l}, \dots, SK_{1^l}$ .  $\mathcal{C}$  then provides  $\mathcal{A}$  with the public key  $PK$  and all private keys  $SK_{ID}$  for  $ID \notin S^*$ .
- Phase 1:** (*optional for CCA*) Attacker  $\mathcal{A}$  adaptively issues decryption queries  $q_1, \dots, q_d$  where a decryption query consists of the triple  $(S, ID, Hdr_S)$  with  $S \subseteq S^*$  and  $ID \in S$ .  $\mathcal{C}$  responds with  $\text{Decrypt}(S, ID, SK_{ID}, Hdr_S)$ .
- Challenge:** For the challenge, algorithm  $\mathcal{A}$  outputs two messages  $M_0$  and  $M_1$ .  $\mathcal{C}$  picks  $\xi \in \{0, 1\}$ , encrypts the message  $M_\xi$  by running  $\text{Encrypt}(S^*, PK, M_\xi)$  to obtain  $Hdr_{S^*}$ , and gives  $Hdr_{S^*}$  to the attacker  $\mathcal{A}$ .
- Phase 2:** (*optional for CCA*) Attacker  $\mathcal{A}$  continues to adaptively issue decryption queries  $q_{d+1}, \dots, q_D$  where a decryption query consists of  $(S, ID, Hdr_S)$  with  $S \subseteq S^*$  and  $ID \in S$ . The only constraint is that  $Hdr_S \neq Hdr_{S^*}$ .  $\mathcal{C}$  responds as in query phase 1.

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

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

**Definition 1.** A public key broadcast encryption  $\Pi$  is  $(t, \epsilon, l, \lambda)$ -single-set-CPA secure if for every  $t$ -time adversary  $\mathcal{A}$  we have that  $|\text{AdvSSBr}_{\mathcal{A}, \Pi}(l, \lambda) - 1/2| < \epsilon$ .

**Definition 2.** A public key broadcast encryption  $\Pi$  is  $(t, \epsilon, l, \lambda, d, D)$ -single-set-CCA secure if  $|\text{AdvSSBr}_{\mathcal{A}, \Pi}(l, \lambda) - 1/2| < \epsilon$  for every  $t$ -time adversary  $\mathcal{A}$  with at most  $D$  decryption queries.

The multi-set security game is defined similar to the single-set security game, except the challenged set is given as multiple subsets.

**Init:** Algorithm  $\mathcal{A}$  outputs a set  $S^* = \{S_1^*, \dots, S_w^*\}$  of users to attack.

**Setup:** The challenger  $\mathcal{C}$  executes  $\text{Setup}(l, \lambda)$  to obtain a public key  $PK$  and a master key  $MK$ .

**KeyGen:** The challenger  $\mathcal{C}$  runs  $\text{KeyGen}(ID, MK, PK)$  to obtain private keys  $SK_{0^i}, \dots, SK_{1^i}$ .  $\mathcal{C}$  gives  $\mathcal{A}$  all private keys  $SK_{ID}$  for  $ID \notin S_i^*$  where  $i = 1, \dots, w$ .

**Phase 1:** (optional for CCA) Attacker  $\mathcal{A}$  adaptively issues decryption queries  $q_1, \dots, q_d$  where a decryption query consists of the triple  $(S, ID, Hdr_S)$  with  $S \subseteq S^*$  and  $ID \in S$ .  $\mathcal{C}$  responds with  $\text{Decrypt}(S, ID, SK_{ID}, Hdr_S)$ .

**Challenge:** For the challenge, algorithm  $\mathcal{A}$  outputs two messages  $M_0$  and  $M_1$ .  $\mathcal{C}$  picks  $\xi \in \{0, 1\}$ , encrypts the message  $M_\xi$  by running  $\text{Encrypt}(S_i^*, PK, M_\xi)$  to obtain  $Hdr_{S_i^*}$  for  $i = 1, \dots, w$ , and gives all  $Hdr_{S_i^*}$  to the attacker  $\mathcal{A}$ .

**Phase 2:** (optional for CCA) Attacker  $\mathcal{A}$  continues to adaptively issue decryption queries  $q_{d+1}, \dots, q_D$  where a decryption query consists of  $(S, ID, Hdr_S)$  with  $S \subseteq S^*$  and  $ID \in S$ . The only constraint is that  $Hdr_{S_i} \neq Hdr_{S_i^*}$ .  $\mathcal{C}$  responds as in query phase 1.

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

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

**Definition 3.** A public key broadcast encryption  $\Pi$  is  $(t, \epsilon, l, \lambda)$ -multi-set-CPA secure if  $|\text{AdvMSBr}_{\mathcal{A}, \Pi}(l, \lambda) - 1/2| < \epsilon$  for every  $t$ -time adversary  $\mathcal{A}$ .

**Definition 4.** A public key broadcast encryption  $\Pi$  is  $(t, \epsilon, l, \lambda, d, D)$ -multi-set-CCA secure if for every  $t$ -time adversary  $\mathcal{A}$  with at most  $D$  decryption queries we have that  $|\text{AdvMSBr}_{\mathcal{A}, \Pi}(l, \lambda) - 1/2| < \epsilon$ .

In [KLLO17], it is shown that the single-set security implies the multi-set security.

**Theorem 1.** [KLLO17] Suppose the public key broadcast encryption  $\Pi$  is  $(t, \epsilon, l, \lambda)$ -single-set-CPA secure ( $(t, \epsilon, l, \lambda, d, D)$ -single-set-CCA secure). Then public key broadcast encryption  $\Pi$  is  $(t, \epsilon', l, \lambda)$ -multi-set-CPA secure ( $(t, \epsilon', l, \lambda, d, D)$ -multi-set-CCA secure) for  $\epsilon' < \epsilon * w$ , where  $w$  is the number of subsets.

### 3.3 Bilinear Groups

We briefly examine bilinear maps and bilinear map groups. We adopt the following notation [Jou04, BF03, GPS08].

1.  $\mathbb{G}_1, \mathbb{G}_2$  and  $\mathbb{G}_T$  are (multiplicative) cyclic groups of prime order  $p$ .
2.  $g_1$  and  $g_2$  are generators of  $\mathbb{G}_1$  and  $\mathbb{G}_2$ , respectively.
3.  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  denotes a bilinear map.

Let  $\mathbb{G}_1, \mathbb{G}_2$ , and  $\mathbb{G}_T$  be groups as above. A bilinear map is a map  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  with satisfying the following properties:

1. Bilinear: for all  $u \in \mathbb{G}_1, v \in \mathbb{G}_2$  and  $a, b \in \mathbb{Z}_p$ , we have  $e(u^a, v^b) = e(u, v)^{ab}$
2. Non-degenerate:  $e(g_1, g_2) \neq 1$ .

We say that  $\mathbb{G}_1$ , and  $\mathbb{G}_2$  are bilinear groups if the group action in  $\mathbb{G}_1$  and  $\mathbb{G}_2$  can be computed efficiently and there exist a group  $\mathbb{G}_T$  and an efficiently computable bilinear map  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  as above.

### 3.4 Computational Complexity Assumptions

The security of our system is based on a complexity assumption called  $q$ -simplified multi exponent bilinear Diffie-Hellman ( $q$ -SMEBDH) assumption. The  $q$ -SMEBDH assumption was originally introduced in [LKL14], but without formal analysis on the hardness of the assumption. In this paper, we formally show that the  $q$ -SMEBDH is a weaker assumption than the  $q$ -bilinear Diffie-Hellman exponent known as  $q$ -BDHE, by reducing  $q$ -SMEBDH to the  $q$ -BDHE.

**Assumption 1** ( *$q$ -Simplified Multi-Exponent Bilinear Diffie-Hellman,  $q$ -SMEBDH*).

Let  $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$  describe the bilinear group of prime order  $p$  with the security parameter  $\lambda$ . Let  $g_1$  and  $g_2$  be generators of  $\mathbb{G}_1$  and  $\mathbb{G}_2$ , respectively. The  $q$ -SMEBDH assumption is that if the challenge tuples  $P = ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), g_1, g_2, g_1^c, g_2^c, \{g_1^{a_i}, g_2^{a_i}, g_1^{b/a_i}, g_2^{b/a_i}\}_{1 \leq i \leq q}, \{g_1^{ba_i/a_j}, g_2^{ba_i/a_j}\}_{1 \leq i, j, i \neq j, \leq q})$  and  $T$  are given, no PPT algorithm  $\mathcal{B}$  can distinguish  $T = T_0 = e(g_1, g_2)^{bc}$  from  $T = T_1 = e(g_1, g_2)^d$  with more than a negligible advantage. The advantage of  $\mathcal{B}$  is defined as  $Adv_{\mathcal{B}}^{q\text{-SMEBDH}}(\lambda) = Pr[\mathcal{B}(P, T_0) = 0] - Pr[\mathcal{B}(P, T_1) = 0]$  where the probability is taken over the random choice of  $a_1, \dots, a_i, b, c, d \in \mathbb{Z}_p$ .

We prove that the  $q$ -SMEBDH is weaker than the well-known  $q$  bilinear Diffie-Hellman exponent assumption ( $q$ -BDHE). The (decisional)  $q$ -BDHE problem is stated as follows[BGG05, BGW05, DGB13, CMM16]: given a vector of elements

$$(g_1, h_1, \{g_1^{\alpha^i}\}_{i \in [2q], i \neq q+1}, g_2, h_2, \{g_2^{\alpha^i}\}_{i \in [2q], i \neq q+1}) \in \mathbb{G}_1^{2q+1} \times \mathbb{G}_2^{2q+1}$$

as input, it should be hard to distinguish  $e(g_1, h_2)^{\alpha^{q+1}}$  ( $= e(h_1, g_2)^{\alpha^{q+1}}$ ) from random where  $\log_{g_1} h_1 = \log_{g_2} h_2$ .

**Lemma 1.** *If there is an adversary  $\mathcal{A}$  which solves a  $q$ -SMEBDH problem with  $\epsilon$  advantage in time  $\tau$  then there is an adversary which solves a  $q$ -BDHE problem with  $\epsilon$  advantage in time  $\tau + q^2$ .*

*Proof.* We will reduce a  $q$ -BDHE problem to a  $q$ -SMEBDH problem. Assume that  $(g'_1, \{g'_1 \alpha^i\}_{i \in [2q], i \neq q+1}, g'_2, h, \{g'_2 \alpha^i\}_{i \in [2q], i \neq q+1})$  is given. To reduce it to  $q$ -SMEBDH, choose random exponents  $v_1, \dots, v_q \in \mathbb{Z}_p$ . Let  $a_i = \alpha^i \cdot v_i$ . Let  $b = \alpha^{q+1}$ .

$$\begin{aligned} g_1^{a_i} &= g'_1 \alpha^i v_i & g_2^{a_i} &= g'_2 \alpha^i v_i \\ g_1^{b/a_i} &= g'_1 \alpha^{q+1-i} v_i^{-1} & g_2^{b/a_i} &= g'_2 \alpha^{q+1-i} v_i^{-1} \\ g_1^{ba_j/a_i} &= g'_1 \alpha^{q+1+j-i} v_j v_i^{-1} & g_2^{ba_j/a_i} &= g'_2 \alpha^{q+1+j-i} v_j v_i^{-1} \end{aligned}$$

Note that since  $i \neq j$ ,  $q+1+j-i \neq q+1$  and  $2 \leq q+1+j-i \leq 2q$ . Let  $g_1^c = h_1$  and  $g_2^c = h_2$ .

If for a given  $q$ -SMEBDH there is an adversary  $\mathcal{A}$  which decides whether  $T = e(g_1^b, g_2^c)$  with  $\epsilon$  advantage then using  $\mathcal{A}$ , we can decide whether  $T = e(g_1, h_2)^{\alpha^{q+1}}$  with  $\epsilon$  advantage since  $e(g_1, h_2)^{\alpha^{q+1}} = e(g_1, g_2^c)^b$ .

### 3.5 Combinatorial Subset Difference

The subset cover representation method of our system is based on the combinatorial subset difference (CSD) proposed in [KLLO17]. The CSD uses a more general, thus, more compact representation method which is extended from the subset difference (SD). The subset difference is the most common representation method in the broadcast encryption (BE) in literature, which denotes a subset with a difference of two subtrees. To be more specific, the SD method constructs a binary tree by mapping the users to the leaf nodes, and represents the subset of privileged users by subtracting the two complete subtrees denoted as the root node of each subtree (i.e.  $(CL, RL)$  where  $CL$  is a covered set and  $RL$  is a revoked set).

CSD [KLLO17] is a more universal type of representation method that consists of a subtraction of two *non-hierarchical* labels. It is similar to the SD method, but  $CL$  and  $RL$  are no longer subtrees; labels are bit-strings which consist of  $\{0, 1, *\}$  where a wildcard  $*$  includes both 0 and 1. CSD is a more generalized expression compared to the SD and includes all possible SD combinations. The number of subsets in CSD is always smaller than that of SD, or at least the same. The header sizes in CSD are  $2r$  in the worst case, while they are  $4r$  in the SD in the worst case ( $r =$  the number of revoked users).

A secure and efficient BE construction compatible with CSD is more challenging than a construction based on SD. Since the key structure is not bound to the tree structure anymore, there are more representation cases that a privileged user has to decrypt using its key. Thus, the BE scheme with CSD may cause key size growth to cover additional cases and is even harder to reduce the key size. In this paper, we propose the first BE scheme which minimizes the key size to be logarithmic and is compatible with even CSD as well as SD.

## 4 Proposed Broadcast Encryption Scheme

In this section, we propose BESTIE, a broadcast encryption scheme applicable to tiny IoT equipments and prove its security. In subsection 4.1, we describe our intuitions of how to compress the key size. We construct our proposed BE scheme in subsection 4.2 and prove its security in subsection 4.3.

### 4.1 Main Idea

Before the formal description, we informally elaborate a sketch of the idea that lies behind the BESTIE. The main contribution of the BESTIE is to compress the private key size from  $O(\log^2 n)$  to  $O(\log n)$ .

As mentioned in section 1, most subset difference based schemes require  $O(l)$  size key for each bit in the ID. I.e., given a private key  $SK = (SK_{ID_1}, \dots, SK_{ID_l})$ , each element  $SK_{ID_i}$  contains a primary key, and  $O(l)$  size auxiliary key. In the existing schemes, each auxiliary key should contain an independent random value; otherwise combination of keys may generate an unauthorized decryption key.

In [KLLO17], the CSD subset is represented as  $(CL, RL)$  where  $CL$  is a covered set and  $RL$  is a revoked set and a user with an ID should be able to construct its decryption key only if it belongs to the covered set but NOT to the revoked set, i.e.,  $ID \in CL$  and  $ID \in \neg RL$ . The combination of key elements derives a decryption key. However, the combination should be performed in a restricted way to prevent from generating any unauthorized decryption key. To ensure that the combination generates only legitimate keys, the scheme has the auxiliary keys with different random exponents  $r_i$  for each primary key. The resulting  $SK$  for the user with  $ID = ID_1 \dots ID_l$  in the CSD is summarized as follows.

$$SK_{ID_i} = \{g^{\alpha(\dots k_{i,\overline{ID}_i})^{r_i}}\} \cup \{k_{j,0}^{r_i}, k_{j,1}^{r_i} \mid j \neq i, j \in [1, l]\} \quad (1)$$

where sets  $\{g^{\alpha(\dots k_{i,\overline{ID}_i})^{r_i}}\}$  and  $\{k_{j,0}^{r_i}, k_{j,1}^{r_i} \mid j \neq i, j \in [1, l]\}$  include a primary key and auxiliary keys, respectively.

Thus the key size becomes  $O(\log^2 n)$ , since  $i \in [1, l]$ . (i.e.,  $O(\log n)$  per combination  $\times \log n$  combinations). Existing subset difference based BE schemes [LCL<sup>+</sup>10, LKLP14] have the similar approach. Hence the known lower key size bound has been  $O(\log^2 n)$ .

In our approach, we detach  $k_{i,\overline{ID}_i}$  from the primary key  $g^{\alpha(\dots k_{i,\overline{ID}_i})^{r_i}}$  of  $SK_{ID_i}$  in equation 1 by splitting the master key  $\alpha$  into a pair  $(\alpha - \alpha_w, \alpha_w)$ , and apply the same random  $r$  to the auxiliary keys as follows:

$$SK_{ID} = \{g^{\alpha - \alpha_w(\dots)^r}\} \cup \{g^{\alpha_w} k_{j,\overline{ID}_j}^r, k_{j, ID_j}^r \mid j \in [1, l]\} \quad (2)$$

Now, the key is divided into two parts such that a decryption key can be constructed only if both conditions  $ID \in CL$  and  $ID \in \neg RL$  are satisfied.

Note that if at least a single bit in  $ID$  is different from  $RL$  (i.e.,  $ID \notin RL$ ), then  $g^{\alpha w}$  and  $g^{\alpha - \alpha w}$  can be combined, outputting the decryption key  $g^\alpha$ .

The full construction is more complex and we describe it in the next section.

## 4.2 Construction

In the following construction, we denote  $ID_i$ ,  $CL_i$ , and  $RL_i$  the  $i$ th bit of a bit-string  $ID$ ,  $CL$ , and  $RL$ , respectively. In addition, we denote  $H(ID) = h_0 \prod_{i=1}^l h_{i, ID_i}$ ,  $K(ID) = k_0 \prod_{i=1}^l k_{i, ID_i}$ ,  $h_{i,*} = h_{i,0} \cdot h_{i,1}$  and  $k_{i,*} = 1$ .

**Setup**( $l, \lambda$ ): This algorithm first generates the bilinear groups  $\mathbb{G}_1, \mathbb{G}_2$  of prime order  $p$  of bit size  $\theta(\lambda)$ . It selects random elements  $g_1 \in \mathbb{G}_1$  and  $g_2 \in \mathbb{G}_2$ . It selects a random exponent  $\alpha \in \mathbb{Z}_p$ . It chooses  $O(l)$  random group elements  $h_0, h_{1,0}, h_{1,1}, \dots, h_{l,0}, h_{l,1}, k_0, k_{1,0}, k_{1,1}, \dots, k_{l,0}, k_{l,1} \in \mathbb{G}_1$ . It outputs a master key  $MK = g_1^\alpha$  and a public key as

$$PK = ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), g, h_0, h_{1,0}, h_{1,1}, \dots, h_{l,0}, h_{l,1}, \\ k_0, k_{1,0}, k_{1,1}, \dots, k_{l,0}, k_{l,1}, \Omega = e(g_1, g_2)^\alpha).$$

**KeyGen**( $ID, MK, PK$ ): This algorithm takes as input  $ID = ID_1 \dots ID_l$ , the master key  $MK$ , and the public key  $PK$ . It chooses random exponents  $\alpha_w$  and  $r \in \mathbb{Z}_p$ , and outputs a private key  $SK_{ID}$  as

$$SK_{ID} = (x_0, x_1, \dots, x_l, y_0, y_1 \dots, y_{2l}, z) \\ = (g_1^{\alpha - \alpha w} H(ID)^r, h_{1, \overline{ID_1}}^r, \dots, h_{l, \overline{ID_l}}^r, \\ k_0^r, g_1^{\alpha w} k_{1, \overline{ID_1}}^r, k_{1, ID_1}^r, \dots, g_1^{\alpha w} k_{l, \overline{ID_l}}^r, k_{l, ID_l}^r, g_2^r).$$

**Encrypt**( $S, PK, M$ ): This algorithm takes  $S = (CL, RL) = (CL_1 \dots CL_l, RL_1 \dots RL_l)$  as input labels, the public key  $PK$ , and a message  $M \in \mathbb{G}_T$  as inputs. It selects a random exponent  $t \in \mathbb{Z}_p$  and outputs a ciphertext by implicitly including  $S = (CL, RL)$  as

$$Hdr_S = \{C_0 = \Omega^t \cdot M, C_1 = g_2^t, C_2 = H(CL)^t, C_3 = K(RL)^t\}.$$

**Decrypt**( $S, ID, SK_{ID}, Hdr_S$ ): This algorithm takes a subset  $S = (CL, RL)$ , a user's  $ID$ , a private key  $SK_{ID}$ , and a ciphertext  $Hdr_S$  for  $S$  as inputs. Let  $P = \{i | ID_i \neq RL_i \wedge RL_i \neq *\}$  and  $Q = \{i | ID_i = RL_i \wedge RL_i \neq *\}$ . Let  $d$  denote the number of bits which in  $ID$  are different from  $RL$  or  $d = |P|$ .

If  $d > 0$ , it parses  $SK_{ID} = (x_0, x_1, \dots, x_l, y_0, y_1 \dots, y_{2l}, z)$ .

Then it computes

$$x' = x_0 \prod_{CL_i = *} x_i \\ y' = (y_0 \prod_{i \in P} y_{2i-1} \prod_{i \in Q} y_{2i})^{d^{-1}}$$

and outputs a message as

$$M = C_0 \cdot e(x' \cdot y', C_1)^{-1} \cdot e(C_2 \cdot C_3^{d-1}, z).$$

Otherwise, it outputs  $\perp$ .

The correctness is verified by the following equation.

$$\begin{aligned} x' &= x_0 \prod_{CL_i=*} x_i = g_1^{\alpha-\alpha w} (h_0 \prod_{i=1}^l h_{i, ID_i})^r \cdot \prod_{CL_i=*} h_{i, ID_i}^r \\ &= g_1^{\alpha-\alpha w} (h_0 \prod_{CL_i \neq *} h_{i, ID_i} \prod_{CL_i=*} h_{i,*})^r = g_1^{\alpha-\alpha w} H(CL)^r \\ y' &= (y_0 \prod_{i \in P} y_{2i-1} \prod_{i \in Q} y_{2i})^{d-1} \\ &= (k_0^r g_1^{\alpha w d} \prod_{RL_i \neq *} k_{i, RL_i}^r)^{d-1} \\ &= g_1^{\alpha w} (k_0 \prod_{RL_i \neq *} k_{i, RL_i})^{rd-1} = g_1^{\alpha w} K(RL)^{rd-1} \end{aligned}$$

Since  $x' = g_1^{\alpha-\alpha w} H(CL)^r$ ,  $y' = g_1^{\alpha w} K(RL)^{rd-1}$ , and  $x' \cdot y' = g_1^{\alpha} H(CL)^r K(RL)^{rd-1}$ ,

$$\begin{aligned} \frac{e(x' \cdot y', C_1)}{e(C_2 \cdot C_3^{d-1}, z)} &= \frac{e(g_1^{\alpha} H(CL)^r K(RL)^{rd-1}, g_2^t)}{e(H(CL)^t K(RL)^{td-1}, g_2^r)} \\ &= e(g_1, g_2)^{\alpha t} = \Omega^t. \end{aligned}$$

### 4.3 Security Proof

**Theorem 2.** *Let  $\mathbb{G}_1$  and  $\mathbb{G}_2$  be bilinear groups of prime order  $p$ . Suppose the (decision)  $(t, \epsilon, 4q)$ -SMEBDH assumption holds in  $\mathbb{G}_1 \times \mathbb{G}_2$ . Then the proposed public key broadcast encryption system is  $(t', \epsilon, q, \lambda)$  semantically secure for arbitrary  $q$ , and  $t' < t + O(\epsilon q^2)$ , where  $\epsilon$  is the maximum time for an exponentiation in  $\mathbb{G}_1$  and  $\mathbb{G}_2$ .*

*Proof.* Suppose  $\mathcal{A}$  has advantage  $\epsilon$  in attacking the proposed public key broadcast encryption system. Using  $\mathcal{A}$ , we construct an algorithm  $\mathcal{B}$  that solves the (decision)  $4q$ -SMEBDH problem.

For generators  $g_1 \in \mathbb{G}_1$  and  $g_2 \in \mathbb{G}_2$ , and  $b \in \mathbb{Z}_p$ , algorithm  $\mathcal{B}$  is given as input random tuples  $P = ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), g_1, g_2, \{g_1^{a_i}, g_2^{a_i}, g_1^{b/a_i}, g_2^{b/a_i}\}_{1 \leq i \leq 4q}, \{g_1^{ba_i/a_j}, g_2^{ba_i/a_j}\}_{1 \leq i, j, i \neq j, \leq 4q}, g_1^c, g_2^c)$  and  $T$  that is either sampled from  $P_{SMEBDH}$  (where  $T = e(g_1, g_2)^{bc}$ ) or from  $R_{SMEBDH}$  (where  $T$  is uniform and independent in  $\mathbb{G}_T$ ). Algorithm  $\mathcal{B}$ 's goal is to output 1 when the input tuple  $T$  is sampled from  $P_{SMEBDH}$  and 0 otherwise. Note that we let  $l = q$  in this proof. Algorithm  $\mathcal{B}$  interacts with  $\mathcal{A}$  in a selective subset game as follows:

**Init:** The game begins with  $\mathcal{A}$  outputting a subset  $S^* = (CL^*, RL^*)$  to attack where  $CL^*, RL^* \in \{0, 1, *\}^l$ .

**Setup:** To generate the public key, algorithm  $\mathcal{B}$  chooses random exponents  $\gamma_1, \gamma_2, v_1, \dots, v_{4l} \in \mathbb{Z}_p$ , and sets  $h_{i,j} = g_1^{a_{2i-1+j}} \cdot g_1^{v_{2i-1+j}}$ ,  $k_{i,j} = g_1^{a_{2l+2i-1+j}} \cdot g_1^{v_{2l+2i-1+j}}$  for  $i \in \{1, \dots, l\}$  and  $j \in \{0, 1\}$ ,  $h_0 = (\prod_{i=1}^l h_{i, CL_i^*})^{-1} \cdot g_1^{\gamma_1}$  and  $k_0 = (\prod_{i=1}^l k_{i, RL_i^*})^{-1} \cdot g_1^{\gamma_2}$ . Let  $\alpha = b$ .

**KeyGen:** To generate a private key  $SK_{ID}$  for user  $ID \in \{0, 1\}^l$ , algorithm  $\mathcal{B}$  considers the following three cases.

(i)  $ID \notin CL^*$  :

Algorithm  $\mathcal{B}$  chooses random exponents  $r'$  and  $\alpha_w \in \mathbb{Z}_p$  and sets  $r = \frac{-b}{a_{2j-1+ID_j}} + r'$  where  $ID_j \neq CL_j^*$ .

Algorithm  $\mathcal{B}$  can easily compute  $g_1^{v_i r}$ ,

$$\text{since } g_1^{v_i r} = g_1^{\frac{-b}{a_{2j-1+ID_j}} \cdot v_i} \cdot g_1^{v_i r'}.$$

Algorithm  $\mathcal{B}$  computes  $x_0$  as follows:

$$\begin{aligned} \prod_{i=1}^l h_{i, ID_i}^r &= \prod_{i=1}^l (g_1^{a_{2i-1+ID_i}})^{\frac{-b}{a_{2j-1+ID_j}} + r'} \cdot g_1^{v_{2i-1+ID_i} r} \\ &= \prod_{i=1, i \neq j}^l g_1^{-b \cdot \frac{a_{2i-1+ID_i}}{a_{2j-1+ID_j}}} \cdot g_1^{-b} \cdot \prod_{i=1}^l (g_1^{a_{2i-1+ID_i}})^{r'} \cdot g_1^{v_{2i-1+ID_i} r} \\ h_0^r &= ((\prod_{i=1}^l h_{i, CL_i^*})^{-1} \cdot g_1^{\gamma_1})^r \\ &= (\prod_{i=1}^l g_1^{-a_{2i-1+CL_i^*}} \cdot g_1^{\gamma_1})^{\frac{-b}{a_{2j-1+ID_j}} + r'} \cdot g_1^{-v_{2i-1+CL_i^*} r} \\ &= \prod_{i=1}^l g_1^{b \cdot \frac{a_{2i-1+CL_i^*}}{a_{2j-1+ID_j}}} \cdot g_1^{\frac{-b}{a_{2j-1+ID_j}} \gamma_1} \cdot g_1^{-v_{2i-1+CL_i^*} r} \cdot h_0^{r'} \\ x_0 &= g_1^{\alpha - \alpha_w} H(ID)^r = g_1^{b - \alpha_w} \cdot h_0^r \cdot \prod_{i=1}^l h_{i, ID_i}^r \\ &= g_1^{b - \alpha_w} \prod_{i=1}^l g_1^{b \cdot \frac{a_{2i-1+CL_i^*}}{a_{2j-1+ID_j}}} \cdot g_1^{\frac{-b}{a_{2j-1+ID_j}} \gamma_1} \cdot g_1^{-v_{2i-1+CL_i^*} r} \cdot h_0^{r'} \\ &\quad \cdot \prod_{i=1, i \neq j}^l g_1^{-b \cdot \frac{a_{2i-1+ID_i}}{a_{2j-1+ID_j}}} \cdot g_1^{-b} \cdot \prod_{i=1}^l (g_1^{a_{2i-1+ID_i}})^{r'} \cdot g_1^{v_{2i-1+ID_i} r} \\ &= g_1^{-\alpha_w} \prod_{i=1}^l g_1^{b \cdot \frac{a_{2i-1+CL_i^*}}{a_{2j-1+ID_j}}} \cdot g_1^{\frac{-b}{a_{2j-1+ID_j}} \gamma_1} \cdot h_0^{r'} \cdot \prod_{i=1, i \neq j}^l g_1^{-b \cdot \frac{a_{2i-1+ID_i}}{a_{2j-1+ID_j}}} \\ &\quad \cdot \prod_{i=1}^l (g_1^{a_{2i-1+ID_i}})^{r'} \cdot g_1^{-v_{2i-1+CL_i^*} r} \cdot g_1^{v_{2i-1+ID_i} r}. \end{aligned}$$

Algorithm  $\mathcal{B}$  computes  $x_i$ ,  $y_0$ ,  $y_i$ , and  $z$  as follows:

$$\begin{aligned}
 x_i &= h_{i, \overline{ID}_i}^r = g_1^{\frac{a_{2i-1+\overline{ID}_i} \cdot (\frac{-b}{a_{2j-1+\overline{ID}_j}} + r')}{}} \cdot g_1^{v_{2i-1+\overline{ID}_i} r} \\
 &= g_1^{-b \cdot \frac{a_{2i-1+\overline{ID}_i}}{a_{2j-1+\overline{ID}_j}}} \cdot (g_1^{a_{2i-1+\overline{ID}_i}})^{r'} \cdot g_1^{v_{2i-1+\overline{ID}_i} r}. \\
 y_0 &= k_0^r = \left( \left( \prod_{i=1}^l k_{i, RL_i^*} \right)^{-1} \cdot g_1^{\gamma_2} \right)^{\frac{-b}{a_{2j-1+\overline{ID}_j}} + r'} \\
 &= \prod_{i=1}^l g_1^{b \cdot \frac{a_{2l+2i-1+RL_i^*}}{a_{2j-1+\overline{ID}_j}}} \cdot g_1^{\gamma_2 \cdot \frac{-b}{a_{2j-1+\overline{ID}_j}}} \cdot k_0^{r'} \cdot g_1^{-v_{2l+2i-1+RL_i^*} r} \\
 y_{2i-1} &= g_1^{\alpha_w} k_{i, \overline{ID}_i}^r = g_1^{\alpha_w} \cdot g_1^{-b \cdot \frac{a_{2l+2i-1+\overline{ID}_i}}{a_{2j-1+\overline{ID}_j}}} \cdot (g_1^{a_{2l+2i-1+\overline{ID}_i}})^{r'} \cdot g_1^{v_{2l+2i-1+\overline{ID}_i} r}. \\
 y_{2i} &= k_{i, ID_i}^r = g_1^{-b \cdot \frac{a_{2l+2i-1+ID_i}}{a_{2j-1+\overline{ID}_j}}} \cdot (g_1^{a_{2l+2i-1+ID_i}})^{r'} \cdot g_1^{v_{2l+2i-1+ID_i} r}. \\
 z &= g_2^r = g_2^{\frac{-b}{a_{2j-1+\overline{ID}_j}} + r'} = g_2^{\frac{-b}{a_{2j-1+\overline{ID}_j}}} \cdot g_2^{r'}.
 \end{aligned}$$

(ii)  $ID \in CL^*$  and  $ID \in RL^*$  :

Algorithm  $\mathcal{B}$  selects random exponents  $r'$  and  $u \in \mathbb{Z}_p$  and sets  $r = \sum_{j=1}^l \frac{-b}{a_{2l+2j-1+\overline{ID}_j}} + r'$ . It sets  $\alpha_w = b - u$ .

Algorithm  $\mathcal{B}$  can compute  $g^{v_i r}$ , since  $g_1^{v_i r} = g_1^{\sum_{j=1}^l \frac{-b}{a_{2l+2j-1+\overline{ID}_j}} \cdot v_i} \cdot g_1^{v_i r'}$ .

Algorithm  $\mathcal{B}$  computes  $x_0$  as follows:

$$\begin{aligned}
 \prod_{i=1}^l h_{i, ID_i}^r &= \prod_{i=1}^l (g_1^{a_{2i-1+ID_i}})^{\sum_{j=1}^l \frac{-b}{a_{2l+2j-1+\overline{ID}_j}} + r'} \cdot g_1^{v_{2i-1+ID_i} r} \\
 &= \prod_{i=1}^l \prod_{j=1}^l g_1^{-b \cdot \frac{a_{2i-1+ID_i}}{a_{2l+2j-1+\overline{ID}_j}}} \cdot \prod_{i=1}^l (g_1^{a_{2i-1+ID_i}})^{r'} \cdot g_1^{v_{2i-1+ID_i} r}. \\
 h_0^r &= \left( \left( \prod_{i=1}^l h_{i, CL_i^*} \right)^{-1} \cdot g_1^{\gamma_1} \right)^r \\
 &= \left( \prod_{i=1}^l g_1^{-a_{2i-1+CL_i^*}} \cdot g_1^{\gamma_1} \right)^{\sum_{j=1}^l \frac{-b}{a_{2l+2j-1+\overline{ID}_j}} + r'} \cdot \prod_{i=1}^l g_1^{-v_{2i-1+CL_i^*} r} \\
 &= \prod_{i=1}^l \prod_{j=1}^l g_1^{b \cdot \frac{a_{2i-1+CL_i^*}}{a_{2l+2j-1+\overline{ID}_j}}} \cdot \prod_{j=1}^l g_1^{\frac{-b}{a_{2l+2j-1+\overline{ID}_j}} \gamma_1} \cdot h_0^{r'} \cdot \prod_{i=1}^l g_1^{-v_{2i-1+CL_i^*} r}.
 \end{aligned}$$

$$x_0 = g_1^{\alpha - \alpha w} H(ID)^r = g_1^u h_0^r \prod_{i=1}^l h_{i, ID_i}^r.$$

Algorithm  $\mathcal{B}$  computes  $x_i$ ,  $y_0$ ,  $y_i$ , and  $z$  as follows:

$$\begin{aligned} x_i &= h_{i, ID_i}^r \\ &= g_1^{a_{2i-1+ID_i} \cdot (\sum_{j=1}^l \frac{-b}{a_{2l+2j-1+ID_j}} + r')} \cdot g_1^{v_{2i-1+ID_i} r} \\ &= \prod_{j=1}^l g_1^{-b \cdot \frac{a_{2i-1+ID_i}}{a_{2l+2j-1+ID_j}}} \cdot (g_1^{a_{2i-1+ID_i}})^{r'} \cdot g_1^{v_{2i-1+ID_i} r}. \end{aligned}$$

$$\begin{aligned} y_0 &= k_0^r = \left( \left( \prod_{i=1}^l k_{i, RL_i^*} \right)^{-1} \cdot g_1^{\gamma_2} \right)^{\sum_{j=1}^l \frac{-b}{a_{2l+2j-1+ID_j}} + r'} \\ &= \prod_{i=1}^l \prod_{j=1}^l g_1^{b \cdot \frac{a_{2l+2i-1+RL_i^*}}{a_{2l+2j-1+ID_j}}} \cdot \prod_{j=1}^l g_1^{\frac{-b}{a_{2l+2j-1+ID_j}} \cdot \gamma_2} \cdot k_0^{r'} \cdot \prod_{i=1}^l g_1^{-v_{2i-1+RL_i^*} r}. \end{aligned}$$

Note that  $RL_i^* = ID_i$  if  $RL_i^* \neq *$ .

If  $RL_i^* = *$  then  $a_{2l+2i-1+RL_i^*} = 0$  since  $k_{i,*} = 1$ .

$$\begin{aligned} y_{2i-1} &= g_1^{\alpha w} \cdot k_{i, ID_i}^r \\ &= g_1^{b-u} \cdot g_1^{\sum_{j=1}^l -b \cdot \frac{a_{2l+2i-1+ID_i}}{a_{2l+2j-1+ID_j}}} \cdot (g_1^{a_{2l+2i-1+ID_i}})^{r'} \cdot g_1^{v_{2l+2i-1+ID_i} r} \\ &= g_1^{b-u} \prod_{j=1, j \neq i}^l g_1^{-b \cdot \frac{a_{2l+2i-1+ID_i}}{a_{2l+2j-1+ID_j}}} \cdot g_1^{-b} \cdot (g_1^{a_{2l+2i-1+ID_i}})^{r'} \cdot g_1^{v_{2l+2i-1+ID_i} r} \\ &= g_1^{-u} \prod_{j=1, j \neq i}^l g_1^{-b \cdot \frac{a_{2l+2j-1+ID_j}}{a_{2l+2j-1+ID_j}}} \cdot (g_1^{a_{2l+2i-1+ID_i}})^{r'} \cdot g_1^{v_{2l+2i-1+ID_i} r}. \end{aligned}$$

$$y_{2i} = k_{i, ID_i}^r = g_1^{\sum_{j=1}^l -b \cdot \frac{a_{2l+2i-1+ID_i}}{a_{2l+2j-1+ID_j}}} \cdot (g_1^{a_{2l+2i-1+ID_i}})^{r'} \cdot g_1^{v_{2l+2i-1+ID_i} r}$$

$$z = g_2^r = g_2^{\sum_{j=1}^l \frac{-b}{a_{2l+2j-1+ID_j}} + r'} = \prod_{j=1}^l g_2^{\frac{-b}{a_{2l+2j-1+ID_j}}} \cdot g_2^{r'}.$$

(iii)  $ID \in CL^*$  and  $ID \notin RL^*$ :

Algorithm  $\mathcal{B}$  does not require the  $SK_{ID}$ , since  $ID \in S^*$ .

**Challenge:** Algorithm  $\mathcal{A}$  submits challenge labels  $(CL', RL')$  and two messages  $M_0^*, M_1^*$ . If  $(CL' \neq CL^*) \vee (RL' \neq RL^*)$ , then algorithm  $\mathcal{B}$  aborts the simulation since it failed to guess the challenge labels. Otherwise,  $\mathcal{B}$  flips a random coin  $\xi \in \{0, 1\}$  internally.  $\mathcal{B}$  implicitly sets  $t = c$  and creates a challenge ciphertext as

$$(C_0, C_1, C_2, C_3) = (T \cdot M_\xi^*, g_2^c, (g_1^c)^{\gamma_1}, (g_1^c)^{\gamma_2}).$$

**Guess:** Finally,  $\mathcal{A}$  outputs a guess  $\xi' \in \{0, 1\}$ . Algorithm  $\mathcal{B}$  concludes its own game by producing a guess as follows. If  $\xi = \xi'$  then  $\mathcal{B}$  outputs 1 meaning  $T = e(g_1, g_2)^{bc}$ . Otherwise, it outputs 0 meaning that  $T$  is random in  $\mathbb{G}_T$ .

To complete the proof, we show that public keys, private keys, and the challenge ciphertext are correctly distributed. The public keys are correctly distributed since new random elements  $v_i$  are chosen from  $\mathbb{Z}_p$ . The private keys are correctly distributed as shown in the query phase. The challenge ciphertext is correctly distributed since it satisfies the following equation

$$\begin{aligned} C_0 &= e(g_1, g_2)^{\alpha t} M_\xi^* = e(g_1, g_2)^{bc} M_\xi^*, \\ C_1 &= g_2^t = g_2^c, \\ C_2^{t^{-1}} &= H(CL^*) = h_0 \prod_{i=1}^l h_{i, CL_i^*} \\ &= \left( \prod_{i=1}^l h_{i, CL_i^*} \right)^{-1} g_1^{\gamma_1} \prod_{i=1}^l h_{i, CL_i^*} = g_1^{\gamma_1}, \\ C_3^{t^{-1}} &= K(RL^*) = k_0 \prod_{i=1}^l k_{i, RL_i^*} \\ &= \left( \prod_{i=1}^l k_{i, RL_i^*} \right)^{-1} g_1^{\gamma_2} \prod_{i=1}^l k_{i, RL_i^*} = g_1^{\gamma_2} \end{aligned}$$

When the input tuple is sampled from  $P_{SMEBDH}$  (where  $T = e(g_1, g_2)^{bc}$ ) then  $\mathcal{A}$ 's view is identical to its view in a real attack game and therefore  $\mathcal{A}$  satisfies  $|Pr[\xi = \xi'] - 1/2| \geq \epsilon$ . When the input tuple is sampled from  $R_{SMEBDH}$  (where  $T$  is uniform in  $\mathbb{G}_T$ ) then  $Pr[\xi = \xi'] = 1/2$ . Therefore, with  $g_1$  uniform in  $\mathbb{G}_1$ ,  $g_2$  uniform in  $\mathbb{G}_2$ ,  $b$  and  $c$  uniform in  $\mathbb{Z}_p$ , and  $T$  uniform in  $\mathbb{G}_T$  we have that

$$\begin{aligned} |Pr[B(P, e(g_1, g_2)^{bc}) = 0] - Pr[B(P, T) = 0]| \\ \geq |(1/2 + \epsilon) - 1/2| = \epsilon \end{aligned}$$

as required, which completes the proof of the theorem.

## 5 CCA-secure Broadcast Encryption

In this section, we extend our proposed BESTIE to the chosen-ciphertext-secure broadcast encryption, similar to [KLLO17, BCHK07] by attaching an unforgeable one-time signature scheme to the semantically secure PKBE scheme. To

utilize the CCA extension in [KLLO17, BCHK07], we require our broadcast encryption to support general IDs such that wildcards (\*) can be used in IDs for key generation. Thus, we first describe a general ID scheme as a building block. Then we represent a CCA-secure scheme with a security proof.

### 5.1 General ID Scheme

In this section, we explain a general ID scheme as described in [KLLO17], which can include wildcards (\*) in the IDs for key generation. Similar to section 4.2, we denote  $ID_i$ ,  $CL_i$ , and  $RL_i$  the  $i$ th bit of bit-strings  $ID$ ,  $CL$ , and  $RL$ , respectively. In addition, we denote  $H(ID) = h_0 \prod_{i=1}^l h_{i, ID_i}$ ,  $K(ID) = k_0 \prod_{i=1}^l k_{i, ID_i}$ ,  $h_{i,*} = h_{i,0} \cdot h_{i,1}$  and  $k_{i,*} = 1$ .

**Setup**( $l, \lambda$ ): The setup is equivalent to the main scheme in section 4.2.

$$\begin{aligned} MK &= g_1^\alpha, \\ PK &= ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), g, h_0, h_{1,0}, h_{1,1}, \dots, h_{l,0}, h_{l,1}, \\ &\quad k_0, k_{1,0}, k_{1,1}, \dots, k_{l,0}, k_{l,1}, \Omega = e(g_1, g_2)^\alpha). \end{aligned}$$

**KeyGen**( $ID, MK, PK$ ): Private key generation is similar to the main scheme, except for the wildcards (\*). We set  $h_{i,*} = 1$  and populate  $h_{i,0}^r$  and  $h_{i,1}^r$  for  $h_{i,*}^r$ . Similarly, the interpretation of  $k_{i,*}$  covers both  $k_{i,0}$  and  $k_{i,1}$ . Therefore, if  $ID_i = *$ ,  $SK_{ID}$  includes both  $g_1^{\alpha w} k_{i,0}^r$  and  $g_1^{\alpha w} k_{i,1}^r$  as well as  $h_{i,0}^r$  and  $h_{i,1}^r$ . The key generation is summarized as follows:

$$\begin{aligned} SK_{ID} &= (x_0, x_1, \dots, x_l, y_0, y_1, \dots, y_{2l}, z), \text{ where} \\ x_0 &= g_1^{\alpha - \alpha w} H(ID)^r, \\ x_i &= h_{i, ID_i}^r \quad (1 \leq i \leq l) \\ y_0 &= k_0^r \\ &\left\{ \begin{array}{l} y_{2i-1} = g_1^{\alpha w} k_{i, ID_i}^r \\ y_{2i} = k_{i, ID_i}^r \end{array} \right\} \quad (1 \leq i \leq l), \quad \text{if } ID_i \neq * \\ &\left\{ \begin{array}{l} y_{2i-1} = g_1^{\alpha w} k_{i,0}^r \\ y_{2i} = g_1^{\alpha w} k_{i,1}^r \end{array} \right\} \quad (1 \leq i \leq l), \quad \text{if } ID_i = * \\ z &= g_2^r \end{aligned}$$

**Encrypt**( $S, PK, M$ ): The encryption is equivalent to the main scheme in section 4.2.

$$Hdr_S = \{C_0 = \Omega^t \cdot M, C_1 = g_2^t, C_2 = H(CL)^t, C_3 = K(RL)^t\}.$$

**Decrypt**( $S, ID, SK_{ID}, Hdr_S$ ): Similar to the decryption in section 4.2, for  $ID_i \neq *$ , let  $P = \{i | ID_i \neq * \wedge ID_i \neq RL_i \wedge RL_i \neq *\}$  and  $Q = \{i | ID_i \neq * \wedge ID_i =$

$RL_i \wedge RL_i \neq *$ . We define new sets for wildcards as  $P^* = \{ID_i = * \wedge RL_i = 1\}$  and  $Q^* = \{ID_i = * \wedge RL_i = 0\}$ .

Then let  $d = |P| + |P^*| + |Q^*|$ , where  $|P|$  denotes the number of bits which in  $ID$  are different from  $RL$ , and  $|P^*| + |Q^*|$  indicates the number of  $*$  in  $ID$ .

If  $d > 0$ , it parses  $SK_{ID} = (x_0, x_1, \dots, x_l, y_0, y_1 \dots, y_{2l}, z)$ .

Then it computes

$$\begin{aligned} x' &= x_0 \cdot \prod_{CL_i = * \wedge ID_i \neq *} x_i \cdot \prod_{CL_i \neq * \wedge ID_i = *} h_{i, CL_i}^r \cdot \prod_{CL_i = * \wedge ID_i = *} h_{i,0}^r h_{i,1}^r \\ y' &= (y_0 \cdot \prod_{i \in P \cup P^*} y_{2i-1} \cdot \prod_{i \in Q \cup Q^*} y_{2i})^{d-1} \end{aligned}$$

and outputs a message as

$$M = C_0 \cdot e(x' \cdot y', C_1)^{-1} \cdot e(C_2 \cdot C_3^{d-1}, z).$$

Otherwise, it outputs  $\perp$ .

## 5.2 CCA-secure Scheme

In the following notation, a vector  $V = (v_1, \dots, v_n)$  is interchangeably presented as  $v_1 \dots v_n$ . With vectors  $V = (v_1, \dots, v_n)$  and  $V' = (v'_1, \dots, v'_m)$ , we denote the concatenation of  $V$  and  $V'$  or  $V||V' = (v_1, \dots, v_n, v'_1, \dots, v'_m)$ .

We extend our semantically secure broadcast encryption scheme using a similar technique presented in [KLL017, BCHK07] to attain the chosen ciphertext security. We can construct an  $l$ -level public key broadcast encryption system  $\Pi = (\text{Setup}, \text{KeyGen}, \text{Encrypt}, \text{Decrypt})$  secure against chosen-ciphertext attacks using the  $(l+z)$ -level  $\Pi' = (\text{Setup}', \text{KeyGen}', \text{Encrypt}', \text{Decrypt}')$  semantically secure broadcast encryption scheme with a strong one-time signature scheme  $(\text{SigKeyGen}, \text{Sign}, \text{Verify})$  with verification keys which are mapped to  $\{0, 1\}^z$ . The main idea is that  $ID = (b_1, \dots, b_l) \in \{1, 0, *\}^l$  in  $\Pi$  is transformed to  $ID' = ID||*^z = (b_1, \dots, b_l, *, \dots, *) \in \{1, 0, *\}^{l+z}$  in  $\Pi'$ . Therefore, the secret key  $SK_{ID}$  for  $ID$  in  $\Pi$  becomes the secret key  $SK_{ID'}$  in  $\Pi'$ . When encrypting a message  $M$  for the  $ID$  in  $\Pi$ , the sender constructs a  $z$ -bit verification key  $V_{sig} = (e_1, \dots, e_z) \in \{0, 1\}^z$  and then encrypts  $M$  to the  $ID' = ID||V_{sig}$  using  $\Pi'$ .

For more detail,  $l$ -level  $\Pi$  is built using  $(l+z)$ -level  $\Pi'$  and a one-time signature scheme as following:

**Setup**( $l, \lambda$ ): Let  $2^l$  be the maximum number of users and  $\lambda$  be the session key length. Assume that the signature verification key space is  $\{0, 1\}^z$ .

Perform a semantically secure broadcast encryption scheme  $\Pi'$  to generate the public key  $PK$  and master secret key  $MK$ , and output  $PK$  and  $MK$ .

$$PK, MK \leftarrow \text{Setup}'(l+z)$$

$\text{KeyGen}(ID, MK, PK)$ : To generate a private key  $SK_{ID}$  for an identity  $ID = b_1 \dots b_l$  utilizing the master secret key, encode  $ID$  to  $ID' = ID \parallel \underbrace{** \dots *}_z$ . The

key generation algorithm in  $\text{KeyGen}'$  of  $\Pi'$  generates the secret key  $SK'_{ID'}$ .

Let  $SK_{ID} = SK'_{ID'} = (SK'_{ID',1}, \dots, SK'_{ID',l+z})$  and output  $\{SK_{ID}\}_{ID \in \{0,1\}^l}$ .

$$\{SK'_{ID'}\}_{ID' \in \{0,1\}^{l+z}} \leftarrow \text{KeyGen}'(ID', MK, PK)$$

$\text{Encrypt}(S, PK, M)$ : Perform  $\text{SigKeyGen}(1^z)$  algorithm to get a signature signing key  $K_{sig}$  and a verification key  $V_{sig}$ . Assume that  $V_{sig} = e_1 \dots e_z$ . For a given  $S = (CL \parallel V_{sig}, RL \parallel V_{sig})$ , run  $\text{Encrypt}'$  to obtain header  $Hdr_S$  and sign the header as

$$Hdr_S \leftarrow \text{Encrypt}'(S, PK, M)$$

$$\sigma \leftarrow \text{Sign}(Hdr_S, K_{sig})$$

and output the tuple  $Hdr$  as  $(Hdr_S, \sigma, V_{sig})$ .

$\text{Decrypt}(S, ID, SK_{ID}, Hdr)$ :

Parse  $Hdr = ((C_0, C_1, C_2, C_3), \sigma, V_{sig})$ .

1. Verify if  $\sigma$  is valid against  $(C_0, C_1, C_2, C_3)$  under the key  $V_{sig}$ . If invalid, output  $\perp$ .
2. Otherwise, encode  $ID$  to  $ID' = ID \parallel \underbrace{** \dots *}_z$ , execute

$\text{Decrypt}'(S, ID', SK_{ID}, Hdr)$  and output the message  $M$ .

Correctness can be shown with a similar computation to the one in section 4. It is noted that the user key size is enlarged from  $O(l)$  to  $O(l+z)$  and the header size increases by the size of a signature and a verification key.

### 5.3 Security Proof

**Theorem 3.** *Let  $\mathbb{G}$  be a bilinear group of prime order  $p$ . For any integer  $l$ , the public key broadcast encryption system  $\Pi$  is  $(t, \epsilon_1 + \epsilon_2, l, \lambda, D)$  CCA-secure if the public key broadcast encryption system  $\Pi'$  is  $(t', \epsilon_1, l+z, \lambda, 0)$  semantically secure in  $\mathbb{G}$  and the signature scheme is  $(t'', \epsilon_2, z, 1)$  strongly existentially unforgeable. Moreover,  $t < t' - (2(l+z)\mathbf{a} + 2\mathbf{p}) \cdot D - t_s$ , where  $\mathbf{a}$  is point addition time,  $\mathbf{p}$  is pairing time, and  $t_s$  presents the sum of  $\text{SigKeyGen}$ ,  $\text{Sign}$  and  $\text{Verify}$  computation time.*

*Proof.* Assume that there exists a  $t$ -time adversary  $\mathcal{A}$  such that  $|\text{AdvBr}_{\mathcal{A}, \Pi} - 1/2| > \epsilon_1 + \epsilon_2$ . We construct an algorithm  $\mathcal{B}$  that has advantage  $|\text{AdvBr}_{\mathcal{B}, \Pi'} - 1/2| > \epsilon_1$  in  $\mathbb{G}$ . Algorithm  $\mathcal{B}$  proceeds as following.

**Init:** Algorithm  $\mathcal{B}$  performs  $\mathcal{A}$  and receives set  $S^*$  in which users  $\mathcal{A}$  challenges on.  $\mathcal{B}$  executes the  $\text{SigKeyGen}$  algorithm to obtain a signature signing key  $K_{sig}^*$  and a verification key  $V_{sig}^* \in \{0,1\}^z$ . Let  $V_{sig}^* = e_1 \dots e_z$ , then  $\mathcal{B}$  builds  $S^{**} = \{U \parallel V_{sig}^* \mid U \in S^*\}$  and outputs it.

**Setup:**  $\mathcal{B}$  gets the public key  $PK$  of  $\Pi'$  from challenger  $\mathcal{C}$ .

**KeyGen:**  $\mathcal{B}$  obtains secret keys  $SK_{ID'}$  for revoked  $ID' \notin S^{**}$  from challenger  $\mathcal{C}$ . Note that  $ID' \notin S^{**}$  iff  $\forall X \in S^{**}, \exists i, ID'_i \neq X_i \wedge X_i \neq *$ , and  $ID' \in S^{**}$  iff  $\exists X \in S^{**}, \forall i, ID'_i = X_i \vee X_i = *$ .

Since  $\Pi'$  can generate secret keys using  $*$ ,  $ID'$  can be classified into the following two forms:

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  $k \in \{1, \dots, z\}$ .

Algorithm  $\mathcal{B}$  responds with  $PK$  and secret keys  $SK'_{ID'}$  of the first type of  $ID'$ . Note 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 following.

**Phase 1:** Algorithm  $\mathcal{A}$  issues decryption queries.

Let  $(ID, S, Hdr)$  be a decryption query where  $S \subseteq S^*$  and  $ID \in S$ . Let  $Hdr = (Hdr_S, \sigma, V_{sig})$ . Algorithm  $\mathcal{B}$  responds as following:

1. Perform *Verify* to check the signature  $\sigma$  against  $Hdr_S = (C_0, C_1, C_2, C_3)$  with verification key  $V_{sig}$ . If the signature is invalid then  $\mathcal{B}$  returns  $\perp$ .
2. If  $V_{sig} = V_{sig}^*$ , then a *forge* event happens, and algorithm  $\mathcal{B}$  outputs a random bit  $b \xleftarrow{\$} \{0, 1\}$  and aborts the simulation.
3. Otherwise,  $\mathcal{B}$  decrypts the header using the second type of secret keys. Let  $V = \underbrace{** \cdots *}_{k-1} \bar{e}_k \underbrace{** \cdots *}_{z-k}$  where  $k \in \{1, \dots, z\}$ . Using  $SK_{ID||V}$ ,  $\mathcal{B}$  can obtain  $M \leftarrow \text{Decrypt}'(S, ID, SK_{ID||V}, Hdr_S)$  since  $V_{sig}$  is covered by  $V$ .

**Challenge:** When  $\mathcal{A}$  outputs  $M_0$  and  $M_1$  for the challenge,  $\mathcal{B}$  bypasses them to  $\mathcal{C}$  and get the challenge  $Hdr_S^*$ . To generate a challenge for  $\mathcal{A}$ ,  $\mathcal{B}$  calculates  $Hdr^*$  as following:

$$\begin{aligned} \sigma^* &\leftarrow \text{Sign}(Hdr_S^*, K_{sig}^*) \\ Hdr^* &\leftarrow (Hdr_S^*, \sigma^*, V_{sig}^*) \end{aligned}$$

$\mathcal{B}$  replies with  $Hdr^*$  to  $\mathcal{A}$ .

**Phase 2:** Same as in query phase 1.

**Guess:** Algorithm  $\mathcal{A}$  outputs a guess  $b \in \{0, 1\}$ . Then  $\mathcal{B}$  outputs 1 if  $b = b'$ , or outputs 0 otherwise.

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

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

It is required to compute the probability of  $\mathcal{B}$  aborting the simulation as a result of a *forge* to conclude the proof of Theorem 3. We argue that  $\Pr[\text{forge}] < \epsilon_2$ . Otherwise one can utilize  $\mathcal{A}$  to forge signatures with a probability of at least  $\epsilon_2$ . Shortly, we can build another simulator that knows the private key, but receives  $K_{sig}^*$  as a challenge in an existential forgery game. In the above experiment,  $\mathcal{A}$  aborts by submitting a query that includes an existential forgery under  $K_{sig}^*$  on some ciphertexts. Our simulator can use this forgery to win the existential forgery game. During the game the adversary makes only one chosen message query to generate the signature for the challenge ciphertext. Hence,  $\Pr[\text{forge}] < \epsilon_2$ . It now follows that  $\mathcal{B}$ 's advantage is at least  $\epsilon_1$  as required.

## 6 Experiment

In this section, we present the experimental results in terms of three main factors - the ciphertext header size, the execution time and the key size - in the proposed scheme (BESTIE) and existing PKBE schemes. To implement the schemes, we utilize the PBC (pairing based cryptography) library using type F parameters ( $f_{param}$ ), and execute them on an Intel Edison with a 32-bit Intel Atom processor 500Mhz and ubuntu 3.10.17.

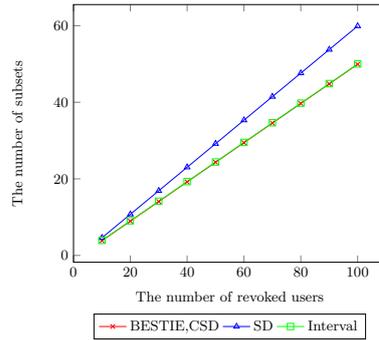


Fig. 2: The number of subsets in the BESTIE, CSD, SD and interval schemes for random revocation ( $2^{128}$  users)

The number of subsets define the ciphertext header sizes in broadcast encryptions. Figure 2 and 3 compares the number of subsets in the bestie, CSD [KLLO17], SD [DF02], and interval schemes [LCL<sup>+</sup>10]. Note that the header sizes are equivalent in BESTIE and CSD since they share the same CSD representation method. In figure 2, the y axis represents the number of subsets as varying the number of randomly chosen revoked users (x axis) varies. The number of total users is  $2^{128}$ . The result shows that the number of subsets is strictly linear to the number of revoked users. In figure 3, instead of a random revocation, we vary the number of randomly chosen secure multicast subsets. Secure multicast subsets

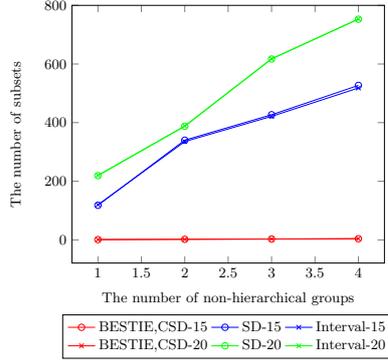


Fig. 3: The number of subsets in the BESTIE, CSD, SD and interval schemes for secure multicast revocation

are non-hierarchical subsets that include wildcards (\*) in the middle of covering labels (e.g.  $1 * *01 - 10 * *1$ ). BESTIE, CSD-15, SD-15, and Interval-15 indicate  $2^{15}$  total users. BESTIE, CSD-20, SD-20, and Interval-20 indicate  $2^{20}$  total users. Since the CSD representation supports a non-hierarchical representation (\* can be placed anywhere), it can cover the non-hierarchical example within a single CSD subset; the number of subsets in BESTIE and CSD is identical to the number of non-hierarchical groups. However, the number of subsets in the SD and interval schemes is large since they only support hierarchical representations.

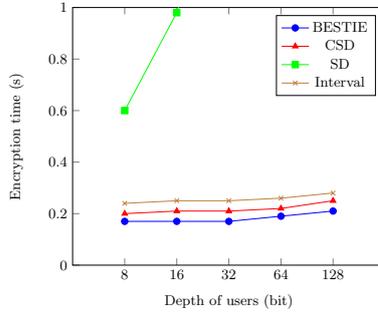


Fig. 4: Encryption time in the BESTIE, CSD, SD, and interval schemes

Figure 4 represents the encryption time in the BESTIE, CSD, and interval schemes. The y-axis represents the encryption time measured in seconds, and the x-axis represents the bit-length of users. The SD scheme follows the encryption of [BBG05], thus it requires point exponentiation for the increasing bit-length. In the figure, the encryption time in SD increases dramatically when the bit-length gets longer. The results show that, other than the SD scheme,

the encryption time remains similar, and BESTIE shows the best encryption performance among the BE schemes.

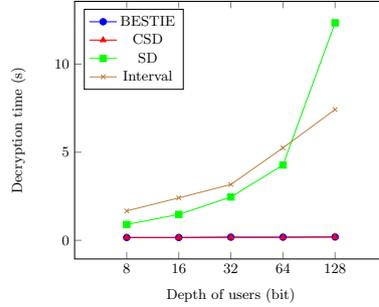


Fig. 5: Decryption time in the BESTIE, CSD, SD, and interval schemes

Figure 5 represents the decryption time in the bestie, csd and interval schemes. Since the decryption is generally performed in a slow IoT (or embedded) system, the decryption performance should be improved. Since the decryption algorithm mostly performs multiplication of secret keys with constant number of pairings and exponentiations in BESTIE and CSD, the decryption time does not increase as the number of users increases. On the other hand, since the interval scheme performs key derivation using a public parameter for decryption, the decryption time is proportional to the depth of users. Hence, BESTIE and CSD are IoT friendly PKBE schemes in decryption.

Table 2: Key size of BESTIE, CSD, SD, and interval schemes

	<i>depth</i> (bits)	<b>BESTIE</b> (ours)	<b>CSD</b> [KLLO17]	<b>SD(HIBE)</b> [DF02, BBG05]	<b>Interval</b> [LCL <sup>+</sup> 10]
<i>PK size</i> (KB)	<i>8bit</i>	0.66	0.66	0.20	0.39
	<i>16bit</i>	1.29	1.29	0.35	0.70
	<i>32bit</i>	2.54	2.54	0.66	1.33
	<i>64bit</i>	5.04	5.04	1.29	2.58
	<i>128bit</i>	10.04	10.04	2.54	5.08
<i>SK size</i> (KB)	<i>8bit</i>	0.53	3.79	10.04	2.58
	<i>16bit</i>	1.00	15.04	80.04	10.08
	<i>32bit</i>	1.93	60.04	640.04	40.08
	<i>64bit</i>	3.81	240.04	5120.04	160.08
	<i>128bit</i>	7.56	960.04	40960.04	640.08

Table 2 shows the public key and secret key size in the bestie, CSD, SD, and interval schemes. When the number of users is  $2^{128}$ , BESTIE requires 7.56KB of

SK storage while CSD, SD, and interval schemes require 960KB, 40960KB, and 640KB, respectively.

## 7 Conclusion

This paper proposes a *broadcast encryption scheme for tiny IoT equipments (BESTIE)*, that reduces the key size suitable for a large scale IoT systems. The proposed BESTIE is a public key broadcast encryption scheme for the combinatorial subset difference (CSD) representation. BESTIE has the most efficient ciphertext header size which is  $2r$  in the worst case, where  $r$  is the number of revoked users. Most importantly, BESTIE is the first scheme to reduce a key size to  $O(\log n)$  from  $O(\log^2 n)$  which was the minimal key size in existing subset difference based approaches, without sacrificing any other factor.

The experimental results show that the BESTIE has the best performance in key generation, encryption and decryption. Furthermore, in BESTIE the SK size is no more than 7KB even for the IPv6 128bit settings (or  $2^{128}$  devices). We prove that the proposed BESTIE is secure under  $q$ -Simplified Multi-Exponent Bilinear Diffie-Hellman ( $q$ -SMEBDH) assumption without the random oracle model.

## References

- AHL<sup>+</sup>12. Nuttapon Attrapadung, Javier Herranz, Fabien Laguillaumie, Benoît Libert, Elie De Panafieu, and Carla Ràfols. Attribute-based encryption schemes with constant-size ciphertexts. *Theoretical computer science*, 422:15–38, 2012.
- BBG05. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2005.
- BCHK07. Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-Ciphertext security from Identity-Based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.
- BF03. Dan Boneh and Matthew K. Franklin. Identity-Based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.
- BGW05. Dan Boneh, Craig Gentry, and Brent Waters. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005.
- BH08. Dan Boneh and Michael Hamburg. Generalized identity based and broadcast encryption schemes. In *Advances in Cryptology-ASIACRYPT 2008*, pages 455–470. Springer, 2008.
- BSW06. Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *Advances in Cryptology-EUROCRYPT 2006*, pages 573–592. Springer, 2006.

- BW06. Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 211–220. ACM, 2006.
- CGI<sup>+</sup>99a. Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast security: A taxonomy and some efficient constructions. In *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, volume 2, pages 708–716. IEEE, 1999.
- CGI<sup>+</sup>99b. Ran Canetti, Juan A. Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast security: A taxonomy and some efficient constructions. In *Proceedings IEEE INFOCOM '99*, pages 708–716, 1999.
- CHK04. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-Ciphertext security from Identity-Based encryption. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 207–222, 2004.
- CJKY08. Jung H. Cheon, Nam-Su Jho, Myung-Hwan Kim, and Eun S. Yoo. Skipping, cascade, and combined chain schemes for broadcast encryption. *IEEE Trans. Information Theory*, 54(11):5155–5171, 2008.
- CMM16. Melissa Chase, Mary Maller, and Sarah Meiklejohn. Déjà Q all over again: Tighter and broader reductions of q-type assumptions. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, pages 655–681, 2016.
- CMN99. Ran Canetti, Tal Malkin, and Kobbi Nissim. Efficient communication-storage tradeoffs for multicast encryption. In *International conference on the theory and applications of cryptographic techniques*, pages 459–474. Springer, 1999.
- Del07. Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In *Advances in Cryptology-ASIACRYPT 2007*, pages 200–215. Springer, 2007.
- DF02. Yevgeniy Dodis and Nelly Fazio. Public Key Broadcast Encryption for Stateless Receivers. In Joan Feigenbaum, editor, *Digital Rights Management Workshop*, volume 2696 of *Lecture Notes in Computer Science*, pages 61–80. Springer, 2002.
- DGB13. Renaud Dubois, Aurore Guillevic, and Marine Sengelin Le Breton. Improved broadcast encryption scheme with constant-size ciphertext. In *Proceedings of the 5th International Conference on Pairing-Based Cryptography, Pairing'12*, pages 196–202, Berlin, Heidelberg, 2013. Springer-Verlag.
- EMN<sup>+</sup>09. Keita Emura, Atsuko Miyaji, Akito Nomura, Kazumasa Omote, and Masakazu Soshi. A Ciphertext-Policy Attribute-Based encryption scheme with constant ciphertext length. In *Information Security Practice and Experience, 5th International Conference, ISPEC*, pages 13–23, 2009.
- FGL<sup>+</sup>17. Aya Fukami, Saugata Ghose, Yixin Luo, Yu Cai, and Onur Mutlu. Improving the reliability of chip-off forensic analysis of nand flash memory devices. *Digital Investigation*, 20:S1–S11, 2017.
- FN93. Amos Fiat and Moni Naor. Broadcast Encryption. In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, 1993.

- GPS08. Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
- GPSW06. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS*, pages 89–98, 2006.
- GST04. Michael T. Goodrich, Jonathan Z. Sun, and Roberto Tamassia. Efficient Tree-Based Revocation in Groups of Low-State Devices. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 511–527. Springer, 2004.
- GW09. Craig Gentry and Brent Waters. *Advances in Cryptology - EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, chapter Adaptive Security in Broadcast Encryption Systems (with Short Ciphertexts), pages 171–188. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- HS02. Dani Halevy and Adi Shamir. The LSD Broadcast Encryption Scheme. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 47–60. Springer, 2002.
- Jou04. Antoine Joux. A one round protocol for tripartite Diffie-Hellman. *J. Cryptology*, 17(4):263–276, 2004.
- KLLO17. Jihye Kim, Seunghwa Lee, Jiwon Lee, and Hyunok Oh. Combinatorial subset difference public key broadcast encryption scheme for secure multicast. *IACR Cryptology ePrint Archive*, 2017:408, 2017.
- LCL<sup>+</sup>10. Huang Lin, Zhenfu Cao, Xiaohui Liang, Muxin Zhou, Haojin Zhu, and Dongsheng Xing. How to construct interval encryption from binary tree encryption. In *International Conference on Applied Cryptography and Network Security*, pages 19–34. Springer, 2010.
- LKLP14. Kwangsu Lee, Woo K. Koo, Dong H. Lee, and Jong H. Park. Public-key revocation and tracing schemes with subset difference methods revisited. In *European Symposium on Research in Computer Security*, pages 1–18. Springer, 2014.
- NNL01. Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and Tracing Schemes for Stateless Receivers. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, 2001.
- NP01. Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In *Financial cryptography*, pages 1–20. Springer, 2001.
- NP10. Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. *Int. J. Inf. Sec.*, 9(6):411–424, 2010.
- SHO08. Ying Su, Jeremy Holleman, and Brian P Otis. A digital 1.6 pj/bit chip identification circuit using process variations. *IEEE Journal of Solid-State Circuits*, 43(1):69–77, 2008.
- SM03. Alan T Sherman and David A McGrew. Key establishment in large dynamic groups using one-way function trees. *IEEE transactions on Software Engineering*, 29(5):444–458, 2003.
- WHA99. D. M. Wallner, E. J. Harder, and R. C. Agee. Key management for multicast: Issues and architectures. In *Internet draft*, 1999.
- YFDL04. Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *Proceedings of the ACM Conference*

*on Computer and Communications Security, CCS '04*, pages 354–363, New York, NY, USA, 2004. ACM.

ZH10. Zhibin Zhou and Dijiang Huang. On efficient ciphertext-policy attribute based encryption and broadcast encryption: extended abstract. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS*, pages 753–755, 2010.

ZQY08. Huanguo Zhang, Zhongping Qin, and Qi Yang. Design and implementation of the tpm chip j3210. In *Trusted Infrastructure Technologies Conference, 2008. APTC'08. Third Asia-Pacific*, pages 72–78. IEEE, 2008.