BGKM: An Efficient Secure Broadcasting Group Key Management Scheme

Zhibin Zhou
Arizona State University
zhibin.zhou@asu.edu

Dijiang Huang
Arizona State University
dijiang@asu.edu

***Abstract*— Broadcasting Group Key Management (BGKM) scheme is designed to reduce communication, storage, and computation overhead of existing Broadcasting Encryption Schemes (BES). To this end, BGKM proposes a new broadcasting group management scheme by utilizing the basic construction of ciphertext policy attribute based encryption and flat table identity management. Compared to previous approaches, our performance evaluation shows that BGKM greatly improved performance in communication ($O(\log n)$ for single revocation and $O((\log n)(\log m))$ for bulk revocations), storage ($O(\log n)$ for each group member), and computation ($O((\log n)(\log m))$ for both encryption and decryption), where n is the ID space size and m is the number of GMs in the group. Moreover, BGKM scheme provides group forward/backward secrecy, and it is resilience to colluding attacks.**

I. Introduction

We propose a novel secure Broadcasting Group Key Management (BGKM) scheme to reduce communication, storage, and computation overhead compared to existing Broadcasting Encryption Schemes (BES). Our solution extends the basic construction of Ciphertext Policy Attribute Based Encryption (CP-ABE) [2], which is based on identity-based cryptography [5] and threshold secret sharing scheme [23]. As in BES, our scheme uses a trusted key server (or Group Controller – GC) to manage group formations and distribute keys. When joining the group, each group member (GM) is distributed a set of secrets to decrypt a broadcasted Data Encryption Key (DEK) in later communication phase. For GMs addition and deletion, a new encrypted DEK is broadcasted, and only legitimated group members can decrypt the DEK based on their pre-distributed secrets.

Typical applications of BGKM include: digital-TV, content distributions and so on, where only registered (or legitimated) subscribers to the service providers can reveal broadcasted data content. In addition, pay-per-view subgroups of subscribers can be formed dynamically based on users' ad hoc selections, e.g., sports, movies, games, and so on. Moreover, some of the communication groups may overlap or inclusive and have hierarchical relations. For example, the subscribers of premium package can watch programs for basic package subscribers. We can observe that subscribers from the same service providers form an overall communication group; pay-per-view subscribers form multiple dynamic conferences; subscribers to different packages belong to different groups that can be overlapped.

Various BES solutions [3], [4], [6], [14], [38] were proposed to address the key management issues in the above described applications. Previous BES solutions are mostly based on manipulations of intersections or unions of pre-distributed keys among GMs. The most efficient fully collusion resistant BES solutions for arbitrary receiver sets is BGW scheme as described in [3], [4], [6]. However, the overhead of these solutions is still prohibitive when the group size is large. For example, the communication overhead (size of updating messages due to group members' additions and deletions) is $O(m^{\frac{1}{2}})$, where m is the number of group members; storage overhead (the number of pre-distributed keys) is $O(m^{\frac{1}{2}})$; computation overhead (the number of cryptographic operations) is $O(m)$. Flat Table (FT) key management schemes [11], [36] map the set of pre-distributed secret keys for each GM to bit positions in the GM's ID, in order to reduce communication overheads to $O(\log m)$. However, FT solutions simply adopt the shared key solutions and thus are subject to collusion attacks. For example, GMs 001 and 010 can decrypt ciphertexts destined to other GMs, e.g., 011, 000, by combining their secret keys that are mapped to their bit positions.

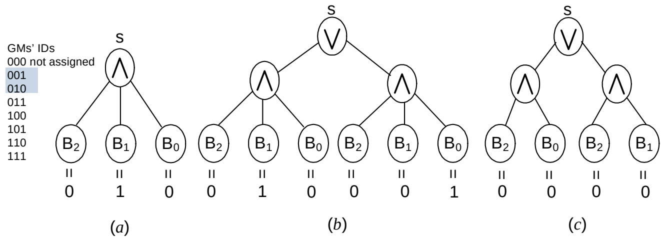

Fig. 1. Examples of access trees.

__Overview of BGKM__: BGKM is designed to improve the performance of existing BES solutions in communication, storage, and computation aspects. To this end, BGKM extends CP-ABE solutions and combines the group management techniques used by FT solutions. CP-ABE allows any user to construct an access control tree (e.g. trees in Figure 1) and perform encryption, so that only receivers who satisfy the access control tree can decrypt the message. However, our extension will enable the GC as the only source to encrypt a message.

To demonstrate the basic construction of BGKM, we present a toy example for a group containing 7 GMs. Each GM is assigned a unique binary ID, from 001 to 111 (suppose 000 is not assigned in this example). For each GM, GC generates a set of secrets, i.e., a private key consists of multiple private key components, and a unique private key component is mapped to each bit position in ID. For example, if a GM's ID $B_2B_1B_0$ is 010, three private key components are mapped to bits: $B_2 = 0$, $B_1 = 1$, $B_0 = 0$, respectively. Note that, 1) although GM 001 has the same bit assignment as GM 010 at the leftmost bit B_2, the private key components mapped to the first bit are different (please see section II-B for details); 2) GC does not need to store any GMs' private keys. Based on assigned ID and distributed private keys, each GM can be uniquely identified by an access control tree, e.g., the access control tree $B_2 = 0 \wedge B_1 = 1 \wedge B_0 = 0$ as shown in Figure 1(a) is only satisfied by GM 010. Now, suppose we want to communicate with a group of GMs such as 010 and 001, the GC can simply combine two access control trees by adding a $\vee$ operator as shown in Figure 1(b). However, the complexity of combined access control tree is tightly bounded to $\Theta(l \cdot log(n))$, where l is the size of group and n is the size of ID space. To reduce the complexity of the access tree, we can further reduce the access control tree from 6 leaves to 4 leaves as shown in Figure 1(c) using the Boolean Function Minimization (BFM) techniques [22], if we consider the 000 as *do not care* (i.e., no group member has been assigned as 000). In this way, only 010 and 001 can satisfy the access tree and use extended CP-ABE to decrypt the DEK s at the root. Through this example, we showed that the GC can construct an access tree based on GMs' IDs. Thus, the research focus is really how to construct access trees based on known GMs' IDs and how to integrate the GM identity management and the extended CP-ABE schemes.

BGKM scheme achieves high efficiency in that: (1) The communication overhead is $O(\log(n))$ for revoking single GM, and $O(\log(n) \cdot \log(m))$ for revoking multiple GMs contrasting to previous broadcasting solutions, which is bounded by $O(m^{\frac{1}{2}})$ [3], [4], [6], where n is the ID space size and m is the number of GMs in the group. Moreover, we proposed a modified CP-ABE algorithm, which further reduce a key update message by approximately 50%; (2) The storage overhead for GC can be as low as $\Theta(1)$ contrasting to previous solution $O(m^{\frac{1}{2}})$ [3], [4], [6]. The storage overhead for a GM is $O(log(n))$; (3) The computation overhead for encryption and decryption is $O(\log(n) \cdot \log(m))$. Moreover, BGKM scheme fulfills the group forward/backward secrecy, and it is resilience to colluding attacks. In addition to the performance and security gain, BGKM is flexible in that it does not just be restricted for broadcast encryption applications. By slightly twisting the protocol, it also can support secure many-to-many group communications, and hierarchical data access control.

We must note that we utilize the basic concept and construction of CP-ABE and borrow some definitions from CP-ABE, such as attributes and access control tree. However, our solution is quite different from CP-ABE in several aspects. First, BGKM only allows the GC to encrypt a message, while every member, even a non-group member can encrypt a message by using CP-ABE. Thus, BGKM is suitable for the control purpose of group formations and is resilient to impersonation attacks and can prevent malicious users from impersonating the GC. Second, we proposed a modified construction of CP-ABE which reduces the size of a ciphertext by approximately 50%.

A. *Paper Organization*

The rest of this paper is organized as follows. Section II presents notations, and a simplified but sufficient version of CP-ABE and the attack model. We present detailed BKGM in Section III. In Section IV, we show the construction for message compression in BGKM scheme and discuss the performance of BGKM scheme. Finally, we conclude our work in Section VI.

II. System and Models

A. *Notations*

Notations used in the rest of paper are presented in the following table.

Symbols	Descriptions
U	the ID space
G	the broadcasting group includes all GMs
J	the set includes all joining GMs
L	the set includes all leaving GMs
C_S	the complementary set of S.
a	one GM
n	Size of ID space
b	number of bits in ID, i.e., $b = log(n)$
m	number of GMs in G
A	attribute
S	set of attributes possessed by a GM

B. Basics of Ciphertext Policy Attribute Based Encryption

In CP-ABE [2], a trusted Private Key Generator (PKG) computes a unique set of private key components for each user with respect to their publicly known attributes. For example, attributes are meaningful, such as faculty, student, computer science department, etc. Although two users may share the same attributes, the private key component associated with each attribute are distinct. Moreover, private key components belongs to different user is incompatible and thus cannot be used together. Thus, collusion attacks are prevented. For encryption, the encrypter must specify an access tree $\mathcal{T} = (\mathcal{A}, \mathcal{O})$ composted by a set of attributes $\mathcal{A} = \{A_1, ..., A_k\}$ as leaves and a set of logical operators $\mathcal{O} = \{\vee, \wedge\}$ as internal nodes. Every internal operator represents a secret sharing threshold with respect to the number of attributes as children of that particular operator. The encryption operation is to split a DEK from the root of a given $\mathcal{T}$ in a top-down fashion using secret sharing scheme. The decryption is just the opposite to the encryption operations in a bottom-up fashion by using shares to recover the secret at the root position. A user can decrypt if and only if his attributes can satisfy the given access tree and reveal the DEK at the root. Here, we present the basic construction of CP-ABE presented in [2]. Particularly, we use $index(x)$ to return the index of x as a child of its parent and use $parent(x)$ to return the parent node of x. CP-ABE is constructed based on bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ of prime order p with a generator g. This bilinear map has the following properties: $e(P^a, Q^b) = \hat{e}(P, Q)^{ab}, \forall P, Q \in \mathbb{G}_0, \forall a, b \in \mathbb{Z}_p^*$. In addition, hash function $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_0$ is also required. CP-ABE scheme consists of four fundamental algorithms: **Setup**, **Encrypt**, **KeyGen** and **Decrypt**. We describe these algorithms at an abstract level, for detailed description of CP-ABE, please refer to [2].

Setup The PKG generates two random numbers α, $\beta \in \mathbb{Z}_p$ and master key $MK = \{\beta, g^\alpha\}$, which are kept privately. It then publishes the following system parameters:
$Para = \{e, H_1, \mathbf{h} = \mathbf{g}^\beta, \mathbf{u} = \mathbf{e}(\mathbf{g}, \mathbf{g})^\alpha\}$.

KeyGen$(Para, MK, S)$ The key generation algorithm will take a set of attributes S of a user as inputs and output a private key SK including multiple private key components associated with each attribute.

- Chooses random values $r \in \mathbb{Z}_p$ and $r_j \in \mathbb{Z}_p$ for each attribute $j \in S$.
- Then computes the key as $SK = (D = g^{(\alpha+r)/\beta}; \forall j \in S : D_j = g^r H_1(j)^{r_j}; D'_j = g^{r_j})$.

Note that in the $Keygen$ algorithm, the random number r is unique for each user. Thus it can guarantee that: 1) even if two users share some common attributes, their corresponding private key components are different; 2) private key components from different users are embedded with different rs, and thus they cannot be used together in the Decrypt algorithm.

Encrypt$(Para, M, \mathcal{T})$ The encryption algorithm encrypts a message M using the access tree $\mathcal{T}$ and $Para$.

- Chooses a polynomial q_x for each node $x \in \mathcal{T}$ (including leaves).
  - Starts from the root node R, for each node x in the tree, sets the degree d_x of the polynomial q_x to be one less than the threshold value k_x of x, thus $d_x = k_x - 1$; chooses a random $s \in \mathbb{Z}_p$ as the top-level secret and sets $q_R(0) = s$, then chooses d_R points as secrets $(q_R(0))$ for lower level polynomials.
  - For any other node x, it sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses d_x other points randomly to completely define q_x.
- Let Y be the set of leaf nodes in $\mathcal{T}$. The ciphertext is then constructed by giving the tree access structure $\mathcal{T}$ and computing: $CT = \{\mathcal{T}; \widetilde{C} = Mu^s; C = h^s; C_Y = \{\forall y \in Y : C_y = g^{q_y(0)}\}; C'_Y = \{\forall y \in Y : C'_y = H_1(y)^{q_y(0)}\}\}$.

Using CP-ABE, any user can encrypt a message because $Param$ and $\mathcal{T}$ is publicly known. However, we need to restrict that only the GC can encrypt a message in BGKM. To this end, we changed the CP-ABE and make $\mathbf{h} = \mathbf{g}^\beta$ and $\mathbf{u} = \mathbf{e}(\mathbf{g}, \mathbf{g})^\alpha$ in the setup procedure only known by the PKG (or GC in BGKM). This modification is viable and will not downgrade the security strength of our solution, since these factors are only used in Encryption, but not in Decryption.

Decrypt(CT, SK) For simplicity, we just define a function $DecryptNode(CT, SK, x)$ that takes as input a ciphertext CT, a private key SK, which is associated with a set S of attributes, and a leaf node x from $\mathcal{T}$. If $i \in S$, then $DecryptNode(CT, SK, x) = \frac{e(D_x, C_x)}{e(D'_x, C'_x)} = e(g, g)^{rq_x(0)}$; else, $DecryptNode(CT; SK; x) = \perp$. For the complete description of Decryption algorithm, please refer to Appendix.

C. Attack Models

We assume an attacker 1) can be a GM or a non-GM (i.e., one to be revoked or one to join a group); 2) can receives and stores all transmitted messages. We assume that no attackers can break the security of CP-ABE with any reasonable probability by solving hard problems of elliptical curves based cryptography, cryptographic hash functions, and symmetric encryption/decryption functions.

III. Broadcasting Group Key Management (BGKM)

BGKM requires that each GM is identified by a unique binary string ID: $B_{b-1}B_{b-2}...B_0$, where $B \in \{0,1\}$. The ID is assigned by the GC when a GM joins the group. Each GM is also uniquely identified by a set of attributes $S = \{A_{i,B_i} | i \in \{0,1,\dots,b-1\}\}$. An attribute A_{i,B_i} denotes "the i^{th} bit of ID is B_i" as illustrated in Figure 2. For a group, the total number of attributes is $2\log n$ when the length of an ID is b; that is, one bit position maps two attributes (one for value 0 and one for value 1). For example, $B_2 = 1$ maps to attribute $A_{2,1}$ and $B_2 = 0$ maps to attribute $A_{2,0}$. As shown in the figure, the attributes belongs to a GM can be represented in a tree structure (note that this tree is different from the access tree presented in Figure 1). The attributes of a GM can be represented by links from the root down to the leave that represents a GM. Thus, each GM will have at least one attribute which is different from other GMs.

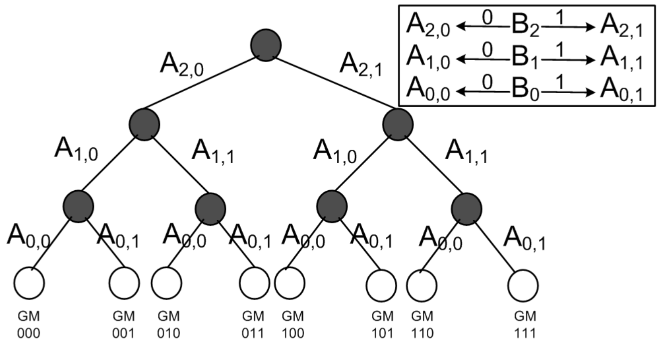

Fig. 2. An illustration of attribute allocation in a 3-bit ID space.

In addition to the hash function H_1, BGKM defines another hash function $H_2 = \{0,1\}^x$, where x is the key length of a DEK. We denote the encryption of message M using a shared key K as $\{M\}_K$; additionally, the encryption of M using CP-ABE with an access policy $\mathcal{T}$ is denoted as $\{M\}_{\mathcal{T}}$.

A. System Setup

The GC is a trusted authority and it generates the following parameters:

1) *Group Public Parameter*: $GP = \{\mathbb{G}_0, \mathbb{G}_1, e, g, H_1, H_2\}$.
2) *Group Encryption Parameter*: $EK = \{h = g^\beta, e(g,g)^\alpha\}$. EK is only known to the GC.
3) *Group Master Key*: $MK = \{\beta, g^\alpha\}$. MK is only known to the GC.

The parameters $h = g^\beta, e(g,g)^\alpha$ are publicly known in CP-ABE. However, in BGKM is in EK and only known to the GC. We denote the encryption as:

$$Encrypt(GP, EK, M, \mathcal{T}). \tag{1}$$

B. Join

When a set of GMs, denoted by J, *join* the group, they need to setup a secure channel with GC, who checks whether each GM is authorized to join. For each accepted GM $a \in J$, GC assigns a unique ID $B^a_{b-1}B^a_{b-2}...B^a_0$ and a set of attributes $S_a = \{A_{i,B^a_i} | i \in \{0,1,\dots,b-1\}\}$.

To preserve group backward secrecy, i.e., a new GM should not have access to data that were transmitted before joining the group, GC renews the DEK to K' and broadcast $\{K'\}_K$, where K is the current DEK. Then the GC runs $SK_a = KeyGen(MK, S_a)$ for each GM $a \in J$. Finally, GC sends the private key SK_a and K' to GM a through a secure channel. In the *join* operation, GC only needs to broadcast one message, i.e., $\{K'\}_K$, for multiple joining GMs.

C. Leave

We present a key update scheme that updates all the remaining GMs' private keys. Then we present BGKM *leave* scheme and show how it works with Boolean function minimization (BFM) [22].

1) Key Update: For a set of leaving GMs, denoted by L, GC needs to update the $\{MK, EK\}$ as well as the private key of each remaining GM $a \in G \setminus L$. To perform key updates, GC first changes $MK' = \{\beta, g^{\alpha'}\}$ and $EK = \{h = g^\beta, e(g,g)^{\alpha'}\}$, where α' is randomly selected in $\mathbb{Z}_p$. Then, GC broadcasts an encrypted key update factor $g^{\frac{\alpha'-\alpha}{\beta}}$. Note that the key update factor is encrypted and should NOT be decrypted by any $a \in L$. How to encrypt the key update factor is presented in Sections III-C.2 and III-C.3.

Each $a \in G \setminus L$ updates its private key SK_a and DEK based on the key update factor $g^{\frac{\alpha'-\alpha}{\beta}}$. The original private key is $SK = (D = g^{(\alpha+r)/\beta}; \forall j \in S : D_j = g^r \times H_1(j)^{r_j}; D'_j = g^{r_j})$. The update factor only affects D. The new D can be updated by the following method:

$$\begin{aligned} D \cdot g^{\frac{\alpha'-\alpha}{\beta}} &= g^{\frac{\alpha+r}{\beta}} \cdot g^{\frac{\alpha'-\alpha}{\beta}} \\ &= g^{\frac{\alpha+r+\alpha'-\alpha}{\beta}} \\ &= g^{\frac{\alpha'+r}{\beta}} \end{aligned}$$

In this way, each $a \in G \setminus L$ update their DEK K' simply by compute $K' = H_2(g^{\frac{\alpha'-\alpha}{\beta}})$.

2) Leave without using BFM: Here, we present the GM *leave* operations without using BFM, which will help us to understand the *leave* operations with BFM presented in the following section. We first consider that only one GM leaves the group. We assume that the leaving GM a's ID is

$B^a_{b-1}B^a_{b-2}...B^a_1B^a_0$. We can define the complementary set of a to be $C_{\{a\}} = \{B'_{b-1}B'_{b-2}...B'_1B'_0| \; \exists b{:}B'_{b-1}{=}\overline{B^a_{b-1}}\}$. Since the ID is represented in binary form, we have $A_{b,\overline{1}} = A_{b,0}$. Then, we can represent the ID of a leaving GM a by

$$\mathcal{T} = (A_{b-1,B^a_{b-1}}) \wedge (A_{b-2,B^a_{b-2}}) \wedge ...(A_{1,B^a_1}) \wedge (A_{0,B^a_0}). \quad (2)$$

$$\overline{\mathcal{T}} = (A_{b-1,\overline{B^a_{b-1}}}) \vee (A_{b-2,\overline{B^a_{b-2}}}) \vee ...(A_{1,\overline{B^a_1}}) \vee (A_{0,\overline{B^a_0}}). \quad (3)$$

Using De Morgan's law we can deduce (3) from (2). Obviously, $G/\{a\} \subseteq C_{\{a\}}$. Thus, to broadcast an encrypted key update factor, GC can perform $Encrypt(GP, EK, , g^{\frac{\alpha'-\alpha}{\beta}}, \overline{\mathcal{T}})$ using (1). This message can be decrypted by $\forall b \in G/\{a\}$. For example, if GM 010 leaves the group, $\overline{\mathcal{T}} = (A_{2,1}) \vee (A_{1,0}) \vee (A_{0,1})$. In this way, we can guarantee that all the remaining GMs can decrypt the key update factor and update the private key as well as DEK, as presented in the section III-C.1.

The operation of *multiple leave* can be performed similar to *single leave*. $\forall a \in L$, GC can derive an access policy that represents the complementary set $C_{\{a\}}$, as mentioned above. Then, GC can represent the intersection of all these complementary sets by constructing a three-level access control tree. The top level is a $\wedge$ gate and the children of this $\wedge$ gate are subtrees representing the complementary set of all leaving GMs. If there are l leaving GMs, we denote the i^{th} GM's ID as $B^i_{b-1}B^i_{b-2}...B^i_1B^i_0$, where $0 \le i \le l-1$. Using (3), we represent $\overline{\mathcal{T}^i}$ as the access policy tree to exclude i^{th} GM, then we can represent the overall access control tree $\overline{\mathcal{T}}$ as:

$$\overline{\mathcal{T}} = \overline{\mathcal{T}^0} \wedge \ldots \wedge \overline{\mathcal{T}^l}. \quad (4)$$

GC then can broadcast $Encrypt(GP, EK, g^{\frac{\alpha'-\alpha}{\beta}}, \overline{\mathcal{T}})$.

3) Leave with BFM: Using (4), the number of attributes equals $l \log n$, where l is the number of leaving GMs. Here, we demonstrate how to use BFM to reduce the total number of attributes used in the access control tree.

We borrow from the results of logical design to our construction of BGKM. First, we define some of the terms we use in subsequent discussions.

- *Literal*: A variable or its complement, e.g. B_1, $\overline{X_1}$, X_2, $\overline{X_2}$, etc.
- *Product Term*: Series of literals related by AND gate, e.g. $\overline{X_2}X_1\overline{X_0}$.
- *Sum Term*: Series of literals related by OR gate, e.g. $X_2 + \overline{X_1} + X_0$.
- *Sum-of-Product Expression (SOPE)*: Series of Product Terms related by OR gate, e.g. $\overline{X_2}X_1X_0 + X_2\overline{X_1}$.

Formally, we define the boolean membership functions $f(X_{b-1}, X_{b-2}, \ldots, X_1, X_0)$ and $f'(X_{b-1}, X_{b-2}, \ldots, X_1, X_0)$, both of which are in the form of SOPE and have b variables. For example, if the set of leaving GMs $L = \{001, 010, 101\}$ and $G \setminus L = \{011, 111\}$, the $f(X_2, X_1, X_0) = \overline{X_2X_1}X_0 + \overline{X_2}X_1\overline{X_0} + X_2\overline{X_1}X_0$ and the $f'(X_2, X_1, X_0) = \overline{X_2}X_1X_0 + X_2X_1X_0$. The following properties of membership functions hold:

$$f(B^a_{b-1}, B^a_{b-2}, \ldots, B^a_1, B^a_0) = \begin{cases} 0 & \text{iff } a \in C_L \\ 1 & \text{iff } a \in L \end{cases}$$

$$f'(B^a_{b-1}, B^a_{b-2}, \ldots, B^a_1, B^a_0) = \begin{cases} 0 & \text{iff } a \in C_{G/L} \\ 1 & \text{iff } a \in G/L \end{cases}$$

The GC runs the Quine-McCluskey algorithm [22] to reduce f and f' to minimal sum-of-product expression $f_{min} = E_0 + \ldots + E_L$ and $f'_{min} = E'_0 + \ldots + E'_{L'}$, respectively, in which the term E uses product term. In each of the computations, we can allow *do not care* values on the U/G, which further reduces the size of both f and f'. After calculating f_{min} and f'_{min}, GC chooses the one with least number of literals, i.e., $\min\{f_{min}, f'_{min}\}$. For example, 011, 101, 100 are leaving GMs, and 001, 010, 110, 111 are remaining GMs, and 000 is never assigned. With *do not care* value 000 considered, f_{min} can be reduced to $\overline{X}_2X_1X_0 + B_2\overline{X}_1$ and f'_{min} can be reduced to $\overline{X}_2\overline{X}_0 + X_2X_1 + \overline{X}_2\overline{X}_1$. We can find that f_{min} contains 5 literals and f'_{min} contains 6 literals. Thus, f_{min} is selected.

If $f_{min} = E_0 + \ldots + E_L$ contains less numbers of literals, GC calculates $\overline{f_{min}}$, which will be the product-of-sum form according to De Morgan law, i.e., $\overline{f_{min}} = \overline{E}_0 \cdots \overline{E}_L$, in which $\overline{E}$ will be the sum of negated literals in E. Thus, the $\overline{f_{min}}$ represents a three-level access control tree, i.e., a $\wedge$ gate at the top and each child is a two-level subtree connected by a $\vee$ gate. Let T_l denote a two-level access control tree, whose top node is a $\vee$ gate and each leaves are attributes corresponding to the literals occurred in $\overline{E}_l$. For example, if $\overline{E}_l = X_2 + \overline{X}_1 + \overline{X}_0$, then $T_l = A_{2,1} \vee A_{1,0} \vee A_{0,0}$. The three-level access policy is:

$$T_0 \ldots \wedge \ldots T_{L'}.$$

One the other hand, suppose $f' = E'_0 + \ldots + E'_{L'}$ contains less numbers of literals. GC can use a three-level access policy with a $\vee$ gate at the top and each child is a two-level sub-tree connected by a $\wedge$ gate. Let T'_l denote a two-level access control tree, whose top node is a $\wedge$ gate and leaves are attributes corresponding to the literals occurred in E'_l. For example, if $\overline{E}_l = X_2\overline{X}_1\overline{X}_0$, then $T'_l = A_{2,1} \wedge A_{1,0} \wedge A_{0,0}$. The three-level access control tree is:

$$T'_0 \ldots \vee \ldots T'_{L'}.$$

GC finally encrypts and broadcasts the key update factor using the three-level access control tree that has the least number of attributes.

D. Security Properties of BKGM

BGKM scheme provides the following security properties:

Lemma 1 (Group Backward Secrecy): GBKM provides group backward secrecy.

Proof Sketch 1: When new GMs *join* the group, a new random DEK K' is encrypted ($\{K'\}_K$), and then distributed through broadcasting. Also, suppose the private keys of joining GMs are generated under $MK' = \{\beta, g^{\alpha'}\}$. All the previous key update messages are encrypted using ($EK = \{h = g^{\beta}, e(g,g)^{\alpha}\}$). Given 1) randomness of K' and α'; 2) security of symmetric encryption and CP-ABE, the group backward secrecy is satisfied.

□

Lemma 2 (Group Forward Secrecy): GBKM provides group forward secrecy.

Proof Sketch 2: When GMs *leave* the group, GC updates the system parameters to MK' and EK' using a new random α' and broadcast the encrypted $g^{\frac{\alpha'-\alpha}{\beta}}$ to all the GMs in $G \setminus L$. All remaining GMs will update their private keys and a new DEK using the key update factor $g^{\frac{\alpha'-\alpha}{\beta}}$. The leaving GMs cannot decrypt future encrypted messages since the decrypting parameter D is changed. Even if a leaving GM stores all encrypted key update messages and *join* the group again, he or she cannot decrypt any previous key update message, since these messages are encrypted under different system parameters, which is unknown to the GM. Moreover, knowing the key update factor $g^{\frac{\alpha'-\alpha}{\beta}}$ to derive g^{α} and β is considered to solve a hard problem. This property will ensure the whole key management scheme is secure, and thus no GM can compromise the GC's master keys.

□

Lemma 3 (Collusion Resistance): GMs cannot collude to decrypt broadcasted messages targeted to other GMs.

Proof Sketch 3: The proof is the same as shown in [2].

□

Lemma 4 (Key Update Authenticity): Only GC can broadcast a legal key update message.

Proof Sketch 4: GMs cannot perform encryption and produce legitimated ciphertexts, since EK is only known by the GC. If we assume CP-ABE is secure, EK cannot be recovered from ciphertext by any attacker. Without knowing EK, the probability of Non-GCs constructing a legitimated ciphertext is $\frac{1}{p^2}$, where p is a large prime number.

IV. Performance Improvement and Assessments

In this section, we first present how to reduce the ciphertext size and assess the performance of BGKM compared to previous solutions.

A. Reducing the Size of Ciphertext

In this section, we present an improved encryption scheme, which can reduce the ciphertext size by 50% compared to CP-ABE scheme.

The message size of CP-ABE ciphertext linearly depends on the number of attributes in the access control tree. In [12], the authors observed that, empirically, a CP-ABE ciphertext encrypted using an access control tree with one attribute is roughly 630 bytes and each additional attribute adds roughly 250 to 300 bytes. This is due to the fact that, for each attributes in access control tree, two members in $\mathbb{G}_0$ are included in ciphertext, i.e., $C_Y = \{C_y = g^{q_y(0)} | \forall y \in Y\}$ and $C'_Y = \{H(\mathbf{att}(y))^{q_y(0)} | \forall y \in Y\}$. The bytes overhead is non-trivial in the network with limited bandwidth.

Note that the function $H_1 : \{0,1\}* \rightarrow \mathbb{G}_0$ hashes arbitrary strings to a member in $\mathbb{G}_0$. We can denote the hashed member to be g^t. GC and GMs cannot derive t, given the hardness of discrete logarithm problem in $\mathbb{G}_0$ and one-way nature of H_1. In our approach, the attributes in the system are in the fixed set $\{A_{i,B} | i \in \mathbb{Z}_b, B \in \{0,1\}\}$ with size of $2n$. Thus, arbitrary attributes are not required and we can remove the hash function H_1. Instead, GC randomly selects $2n$ large prime numbers in $\mathbb{Z}_p$, where p is the group size of $\mathbb{G}_0$. We can denote the set of random numbers to be $R = \{t_{A_{i,B}} | i \in \mathbb{Z}_b, B \in \{0,1\}\}$ and R can be added to *group master keys*.

In particular, We also modify the **KeyGen**, **Encrypt** and **Decrypt** function as follows:

$\mathbf{KeyGen}(MK, S, T)$

The generated private key is $SK = \{D = g^{(\alpha+r)/\beta}; \forall j \in S : D_j = g^{r/t_j}\}$.

$\mathbf{Encrypt}(P, M, \mathcal{T})$

The encrypted ciphertext is $CT = \{\mathcal{T}; \widetilde{C} = Me(g;g)^{\alpha s}; C = h^s; C'_Y = \{C'_y = g^{t_y \cdot q_y(0)} | \forall y \in Y\}\}$. Note that we remove the C_Y from the ciphertext in original CP-ABE scheme.

$\mathbf{Decrypt}(P, CT, SK)$

Then the modified decryption for a leaf node is $\mathbf{DecryptNode}(P, CT, SK, j) = e(D_j, C'_j) = e(g^{r/t_j}, g^{t_j \cdot q_j(0)}) = e(g,g)^{r \cdot q_j(0)}$.

In a summary, the original CP-ABE requires 2 group members in $\mathbb{G}_0$ for each attribute to construct a ciphertext, whereas our solution only requires 1 group member in $\mathbb{G}_0$. Thus, we can reduce the size of ciphertext by roughly 50%.

From security perspective, this modified version of CP-ABE is still collusion-resistant, given the randomness of number r for each user.

B. Performance Assessment

We analyze the performance of our BGKM scheme and compare it with several previous solutions: flat table scheme (FT) [11], [36], subset-difference scheme (Subset-Diff) [14], BGW broadcasting encryption [6], access control polynomial (ACP) scheme [38], and tree based schemes (e.g., OFT [33], LKH [37], and ELK [26]). The performance assessments are assessed in terms of storage overhead (group data to be stored on the GC and GM), communication overhead (number and size of messages to be broadcasted in join and leave operation) and computation overhead (number of cryptographic operations needed in encryption and decryption). In this section, we focus on the comparison between BGKM and another three broadcasting encryption schemes, i.e. Subset-Diff scheme, ACP scheme and BGW scheme. We denote the size of ID space to be n, the number of current GMs to be m, the number of leaving GMs to be l. Also, for the Subset-Diff scheme, t denotes the maximum number of colluding users to compromise the ciphertext. The summary of performance assessment is presented in Table I.

1) Storage Overhead: In BGKM, the storage overhead for GC is $\Theta(m)$ (when GC store the IDs of all current GMs). The storage overhead is $\Theta(\log n)$ for GC, since GM stores a private key component for each bit in its ID.

2) Communication Overhead: In BGKM, the *join* and *leave* operation only require 1 message, regardless of the number of joining or leaving GMs.

The message size is $\Theta(1)$ in the *join* operation. Our discussion focuses on the complexity of *leave* operation. In Subset-Diff scheme, the communication overhead grows linearly with the maximum number of colluding users to compromise the ciphertext. In ACP scheme, the size of message depends on the degree of access control polynomial, which equals to the number of current GMs plus the number of joining GMs or the number of current GMs minus the number leaving GMs. Thus, the message size is $O(m)$. For BGW scheme, the message size is $O(m^{\frac{1}{2}})$ as reported in [6].

In BGKM, the size of message linearly depends on the number of leafs (attributes) in the access control policy tree. We utilize BFM to minimize number of literals in the minimized SOPE, and thus, reduce the number of leaves in the access control tree. In [32], the authors derived an upper bound on the average number of products in the minimized SOPE. According to [12], the number of product expression after the Boolean function minimization is about $O(\log m)$. Given that each product expression contains $\log n$ literals at most, the message size of BGKM is $O(\log n \cdot \log m)$. For single leave, the message is encrypted with access control tree with $\log n$ attributes, i.e., $O(\log n)$.

3) Computation Overhead: For shared key or hash based group key management schemes, the computation overhead is trivial for decryption and encryption. On the other hand, the asymmetric cryptographic operations are non-trivial. Moreover, some systems require large group (e.g. m or n is $2^{16}, 2^{32}, \ldots$). In ACP scheme, the author reports that the encryption needs $O(m^2)$ finite field operations; in the BGW scheme, the encryption and decryption require $O(m)$ operations on the bilinear group, e.g., a group of Elliptic curve points, which is heavier than finite field operations [16], [29]. In BGKM, the encryption requires 1 pairing operation and $\log n$ operations on the bilinear group, and the decryption requires $\log n$ pairing operations and $\log m$ finite field operations. Although the problem of minimizing SOPE is NP-hard, efficient approximations are widely known. Thus, we can conclude that BGKM is much more efficient than ACP and BGW when the size of group goes large.

V. Related Work

Group key management (GKM) has been investigated intensively in centralized group key distribution schemes [9], [20], [21], [25], [27], [37] and decentralized (contributory) key agreement schemes [1], [18], [19], [10], [13], [30], [34], [35]. Due to richness of research publications, we can hardly list all the related researches in this area. We refer to [24], [28] as two excellent surveys.

Tree-based rekey algorithms have gained popularity, including, notably, Logical Key Hierarchy (LKH) [37], One-Way Function Tree (OFT) [33], One-way Function Chain Tree [7], Hierarchical α-ary Tree with Clustering [8] and Efficient Large-Group Key (ELK) [26]. These algorithms provide different tradeoffs among storage, computation and communication overheads. Compared to these schemes, flat table (FT) scheme achieves high efficiency in terms of storage, computation and communication overheads. In FT schemes [11], [36], there are total $2\log n$ KEKs: $\{k_{i,b} | i \in \{0,1,\ldots,2\log n - 1\}, b \in \{1,0\}\}$ and $\log n$ symmetric KEKs are distributed to each GM, with each KEK corresponding to one bit in ID. Despite its efficiency, FT scheme is vulnerable to collusion attacks. To prevent the collusion attacks, Cheung et al. [12] proposed CP-ABE-FT to implement the FT using CP-ABE. CP-ABE-FT utilizes a periodic refreshment mechanism to ensure forward secrecy. However, it has several drawbacks: 1) if the ID of a revoked GM is re-assigned to another GM before the refreshment, the revoked GM can regain the

TABLE I

COMPARISON OF STORAGE OVERHEAD AND NUMBER OF MESSAGES IN DIFFERENT GKM SCHEMES.

Scheme	Storage		Communication Overhead			Computation Overhead	
	GC	GM	join	single leave	multiple leave	encryption	decryption
BGKM	$\Theta(m)$	$\Theta(\log n)$	$\Theta(1)$	$O(\log n)$	$\approx O(\log n \cdot \log m)$	$O(\log n)$	$O(\log n)$
FT	$\Theta(m + \log n)$	$\Theta(\log n)$	$\Theta(\log n)$	$O(\log n)$	$\approx O(\log m)$		
Subset-Diff	$\Theta(m)$	$\Theta(log^2(n))$	N/A	$O(t \cdot log^2(t) \cdot \log m)$	$O(t \cdot log^2(t) \cdot \log m)$		
BGW	$O(m^{\frac{1}{2}})$	$O(m^{\frac{1}{2}})$	N/A	$O(m^{\frac{1}{2}})$	$O(m^{\frac{1}{2}})$	$O(m)$	$O(m)$
ACP	$O(m)$	$O(1)$	$O(m)$	$O(m)$	$O(m)$	$O(m^2)$	$\Theta(1)$
Tree based schemes	$\Theta(m)$	$\Theta(\log m)$	$\Theta(1)$	$\Theta(\log m)$	$O(l \cdot \log m)$		

n: the ID space; m: the number of group members; l: the number of leaving members; t: maximum number of colluding users to compromise the ciphertext.

access to group data and then the group forward secrecy is compromised; 2) outsiders can impersonate GC to disturb the rekey process by sending CP-ABE ciphertexts.

In [14], the authors first formally explored the broadcasting encryption. They presented a solution for m users, which is secure against collusion attacks of t users. The communication overhead of this scheme is $O(t \log^2 t \log m)$. In [6], Boneh et al. proposed a collusion-resistant broadcasting encryption scheme. In this approach, the storage, communication and computation overhead grows linearly with the increase of the number of users.

ID based Encryption (IBE) [5] greatly facilitate the management of public key. One improvement to the ID based cryptography is pseudonym-based cryptography [17](PBC). In PBC, each user is free to generate pseudonym and corresponding key for herself. Attribute based encryption was first proposed by A. Sahai and B. Waters in [31]. In recent years, encrypting data using attributes to enforce fine-grained access control has draw a lot research interests. V. Goyal et al. proposed an attribute-based encryption for fine-grained access control in [15]. In this scheme, each ciphertext is attached with certain descriptive attributes and each user's private key is embedded with an access control policy. J. Bethencourt et al. [2] proposed the ciphertext policy attribute based encryption, in which encrypter specifies an access control policy to confine who can decrypt the ciphertext .

VI. CONCLUSION

In this paper, we proposed a novel BGKM scheme. By utilizing the basic construction of ciphertext policy attribute based encryption and flat table identity management, BGKM greatly improved performance in communication ($O(\log n)$ for one addition and $O((\log n)(\log m))$ for bulk additions), storage ($O(\log n)$ for each group member), and computation ($O((\log n)(\log m))$ for both encryption and decryption), where n is the ID space size and m is the number of GMs in the group. Moreover, BGKM scheme provides group forward/backward secrecy, and it is resilience to colluding attacks.

The future work of this paper would be considered in the following directions: 1) since ciphertext is large of the original CP-ABE scheme, reducing the ciphertext size requires more research efforts; 2) this work is subjected to single failure problem when GC fails, we can investigate distributed group management infrastructure to improve the robustness of BGKM.

REFERENCES

[1] Y. Amir, Y. Kim, C. Nita-Rotaru, JL Schultz, J. Stanton, and G. Tsudik. Secure group communication using robust contributory key agreement. *Parallel and Distributed Systems, IEEE Transactions on*, 15(5):468–480, 2004.

[2] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-Policy Attribute-Based Encryption. *Proceedings of the 28th IEEE Symposium on Security and Privacy (Oakland)*, 2007.

[3] D. Boneh, C. Gentry, and B. Waters. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. *Advances in Cryptology-Crypto 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, 2005.

[4] D. Boneh and B. Waters. A fully collusion resistant broadcast, trace, and revoke system. *Proceedings of the 13th ACM conference on Computer and communications security*, pages 211–220, 2006.

[5] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. *SIAM Journal of Computing*, 32(2):586–615, 2003.

[6] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. pages 573–592, 2006.

[7] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, I.B.M.T.J.W.R. Center, and Y. Heights. Multicast security: a taxonomy and some efficient constructions. *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2, 1999.

[8] R. Canetti, T. Malkin, and K. Nissim. Efficient Communication-Storage Tradeoffs for Multicast Encryption, Advances in Cryptology-Eurocrypt99. *Lecture Notes in Computer Science*, 1592:459–474, 1999.

[9] G. Caronni, M. Waldvogel, D. Sun, and B. Plattner. Efficient security for large and dynamic multicast groups. *Proceedings of the IEEE 7th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE98)*, 1998.

[10] H. Chan, V.D. Gligor, A. Perrig, and G. Muralidharan. On the Distribution and Revocation of Cryptographic Keys in Sensor Networks. *IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING*, pages 233–247, 2005.

[11] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, D. Saha, I.B.M.T.J.W.R. Center, and Y. Heights. Key management for secure Internet multicast using Boolean functionminimization techniques. *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2, 1999.

[12] L. Cheung, J. Cooley, R. Khazan, and C. Newport. Collusion-Resistant Group Key Management Using Attribute-Based Encryption. Technical report, Cryptology ePrint Archive Report 2007/161, 2007. http://eprint. iacr. org.

[13] L.R. Dondeti, S. Mukherjee, and A. Samal. Disec: A distributed framework for scalable secure many-to-many communication. *Proceedings of The Fifth IEEE Symposium on Computers and Communications (ISCC 2000)*, 2000.

[14] A. Fiat and M. Naor. Broadcast Encryption, Advances in Cryptology-Crypto93. *Lecture Notes in Computer Science*, 773:480–491, 1994.

[15] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98, 2006.

[16] D.R. Hankerson, S.A. Vanstone, and A.J. Menezes. *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[17] Dijiang Huang. A Pseudonym-Based Cryptography for Anonymous Communications in Mobile Ad-hoc Networks. *Special Issue on Cryptography in Networks, International Journal of Security and Networks (IJSN)*, 2007.

[18] Dijiang Huang and Deep Medhi. A key-chain-based keying scheme for many-to-many secure group communication. *ACM Trans. Inf. Syst. Secur.*, 7(4):523–552, 2004.

[19] Y. Kim, A. Perrig, and G. Tsudik. Tree-based group key agreement. *ACM Transactions on Information and System Security (TISSEC)*, 7(1):60–96, 2004.

[20] X.S. Li, Y.R. Yang, M.G. Gouda, and S.S. Lam. Batch rekeying for secure group communications. *Proceedings of the 10th international conference on World Wide Web*, pages 525–534, 2001.

[21] D. Liu, P. Ning, and K. Sun. Efficient self-healing group key distribution with revocation capability. *Proceedings of the 10th ACM conference on Computer and communications security*, pages 231–240, 2003.

[22] E.J. McCluskey. Minimization of Boolean functions. *Bell System Technical Journal*, 35(5):1417–1444, 1956.

[23] A.J. Menezes. *Handbook of Applied Cryptography*. CRC Press, 1997.

[24] MJ Moyer, JR Rao, and P. Rohatgi. A survey of security issues in multicast communications. *Network, IEEE*, 13(6):12–23, 1999.

[25] Wee Hock Desmond Ng, Michael Howarth, Zhili Sun, and Haitham Cruickshank. Dynamic balanced key tree management for secure multicast communications. *IEEE Transactions on Computers*, 56(5):590–605, 2007.

[26] A. Perrig, D. Song, and J. Tygar. ELK, A New Protocol for Efficient Large-Group Key Distribution. *IEEE SYMPOSIUM ON SECURITY AND PRIVACY*, pages 247–262, 2001.

[27] A. Perrig and JD Tygar. *Secure Broadcast Communication in Wired and Wireless Networks*. Springer, 2003.

[28] S. Rafaeli and D. Hutchison. A survey of key management for secure group communication. *ACM Computing Surveys (CSUR)*, 35(3):309–329, 2003.

[29] Archana Ramachandran, Zhibin Zhou, and Dijiang Huang. Computing Cryptographic Algorithms in Portable and Embedded Devices. *Portable Information Devices, 2007. PORTABLE07. IEEE International Conference on*, 25-29:1–7, 2007.

[30] M. Ramkumar and N. Memon. An efficient key predistribution scheme for ad hoc network security. *Selected Areas in Communications, IEEE Journal on*, 23(3):611–621, 2005.

[31] A. Sahai and B. Waters. Fuzzy Identity-Based Encryption. *Advances in Cryptology–Eurocrypt*, 3494:457–473.

[32] T. Sasao. Bounds on the average number of products in the minimum sum-of-products expressions for multiple-value input two-valued output functions. *Computers, IEEE Transactions on*, 40(5):645–651, May 1991.

[33] A.T. Sherman and D.A. McGrew. Key Establishment in Large Dynamic Groups Using One-Way Function Trees. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, pages 444–458, 2003.

[34] M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman key distribution extended to group communication. *Proceedings of the 3rd ACM conference on Computer and communications security*, pages 31–37, 1996.

[35] B. Sun, W. Trappe, Y. Sun, and KJR Liu. A time-efficient contributory key agreement scheme for secure group communications. *Communications, 2002. ICC 2002. IEEE International Conference on*, 2, 2002.

[36] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner. The VersaKey Framework: Versatile Group Key Management. *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, 17(9), 1999.

[37] C.K. Wong, M. Gouda, and SS Lam. Secure group communications using key graphs. *Networking, IEEE/ACM Transactions on*, 8(1):16–30, 2000.

[38] X. Zou, Y.S. Dai, and E. Bertino. A Practical and Flexible Key Management Mechanism For Trusted Collaborative Computing. *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 538–546, 2008.

APPENDIX

We first define a function $DecryptNode(CT, SK, x)$ that takes as input a ciphertext CT, a private key SK, which is associated with a set S of attributes, and a node x from T. If the node x is a leaf node then we let $i = att(x)$ and define as follows: If $i \in S$, then

$$\begin{aligned} & DecryptNode(CT, SK, x) \\ & = \frac{e(D_i, C_x)}{e(D'_i, C'_x)} \\ & = \frac{e(g^r \cdot H(i)^{r_i}, g^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})} \\ & = \frac{e(g^r, g^{q_x(0)}) \cdot e(H(i)^{r_i}, g^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})} \\ & = e(g, g)^{rq_x(0)} \end{aligned}$$

If $i \notin S$, then we define $DecryptNode(CT; SK; x) = \perp$.

We now consider the recursive case when x is a non-leaf node. The algorithm $DecryptNode(CT; SK; x)$ then proceeds as follows: For all nodes z that are children of x, it calls $DecryptNode(CT; SK; z)$ and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$.

Otherwise, we compute

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)} \\ &= \prod_{z \in S_x} (e(g; g)^{r \cdot q_z(0)})^{\Delta_{i;S'_x}(0)} \\ &= \prod_{z \in S_x} (e(g; g)^{r \cdot q_{\mathbf{parent}(z)(\mathbf{index}(z))}})^{\Delta_{i;S'_x}(0)} \\ &= \prod_{z \in S_x} (e(g; g)^{r \cdot qx(i) \cdot \Delta_{i;S'_x}(0)} \\ &= e(g, g)^{rq_x(0)} \end{aligned}$$

where $i = \mathbf{index}(z)$ and $S'_x = \{\mathbf{index}(z) : z \in S_x\}$

Now that we have defined $DecryptNode$, we can define the decryption algorithm, which begins by simply calling the $DecryptNode$ function on the root node r of the tree T . If the tree is satisfied by S we set $A = DecryptNode(CT; SK; r) = e(g; g)^{rq_R(0)} = e(g; g)^{rs}$. The algorithm decrypts by computing $\widetilde{C}/(e(C; D)/A) = \widetilde{C}/(e(h^s; g^{(\alpha+r)/\beta})/e(g; g)^{rs}) = M$