CERIAS Tech Report 2010-30
Attribute Based Group Key Management
by Mohamed Nabeel, Elisa Bertino
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

Attribute Based Group Key Management

Mohamed Nabeel, Elisa Bertino
Purdue University,
West Lafayette, Indiana, USA
{nabeel, bertino}@cs.purdue.edu

Abstract

Attribute based systems enable fine-grained access control among a group of users each identified by a set of attributes. Secure collaborative applications need such flexible attribute based systems for managing and distributing group keys. However, current group key management schemes are not well designed to manage group keys based on the attributes of the group members. In this paper, we propose novel key management schemes that allow users whose attributes satisfy a certain access control policy to derive the group key. Our schemes efficiently support rekeying operations when the group changes due to joins or leaves of group members. During a rekey operation, the private information issued to existing members remains unaffected and only the public information is updated to change the group key. Our schemes are expressive; are able to support any monotonic access control policy over a set of attributes. Our schemes are resistant to collusion attacks; group members are unable to pool their attributes and derive the group key which they cannot derive individually.

Index Terms

Broadcast group key management; attribute based policies

I. Introduction

Current technological innovations and new application domains have pushed novel paradigms and tools for supporting collaboration among (possibly very dynamic) user groups (see for example the novel notion of group-centric information sharing [1]). An important requirement in collaborative applications is to support operations for user group memberships, like join and leave, based on identity attributes (attributes, for short) of users; we refer to this requirement as *attribute-based group dynamics*. As today enterprises and applications are adopting identity management solutions, it is crucial that these solutions be leveraged on for managing groups. Typically, a user would be automatically assigned (de-assigned) a group membership based on whether his/her attributes satisfy (cease to satisfy) certain group membership conditions. Another critical requirement is to provide mechanisms for group key management (GKM), as very often the goal of a group is to share data. Thus data must be encrypted with keys made available only to the members of the group. The management of these keys, which includes selecting, distributing, storing and updating keys, should directly and effectively support the attribute-based group dynamics and thus requires an *attribute-based group key management* (AB-GKM) scheme, by which group keys are assigned (or de-assigned) to users in a group based on their

identity attributes. This scheme recalls the notion of attribute-based encryption (ABE) [2], [3], [4]; however, as we discuss later on, ABE has several shortcomings when applied to GKM. Therefore, a different approach is needed.

A challenging well known problem in GKM is how to efficiently handle group dynamics, i.e., a new user joining or an existing group member leaving. When the group changes, a new group key must be shared with the existing members, so that a new group member cannot access the data transmitted before she joined (backward secrecy) and a user who left the group cannot access the data transmitted after she left (forward secrecy). The process of issuing a new key is called *rekeying* or *update*. Another challenging problem is to defend against collusion attacks by which a set of colluding fraudulent users are able to obtain group keys which they are not allowed to obtain individually.

In a traditional GKM scheme, when the group changes, the private information given to all or some existing group members must be changed which requires establishing private communication channels. Establishing such channels is a major shortcoming especially for highly dynamic groups. Recently proposed broadcast GKM (BGKM) schemes [5], [6] have addressed such shortcoming. BGKM schemes allow one to perform rekeying operations by only updating some public information without affecting private information existing group members possess. However, BGKM schemes are not designed to support group membership policies over a set of attributes. In their basic form, they can only support 1-*out-of*-n threshold policies by which a group member possessing 1 attribute out of the possible n attributes is able to derive the group key. In this paper we develop novel expressive AB-GKM schemes which allow one to express any threshold or monotonic [1] conditions over a set of identity attributes.

A possible approach to construct an AB-GKM scheme is to utilize attribute-based encryption (ABE) primitives [2], [3], [4]. Such an approach would work as follows. A key generation server issues each group member a private key (a set of secret values) based on the attributes and the group membership policies. The group key, typically a symmetric key, is then encrypted under a set of attributes using the ABE encryption algorithm and broadcast to all the group members. The group members whose attributes satisfy the group membership policy can obtain the group key by using the ABE decryption primitive. One can use such an approach to implement an

[1]Monotone formulas are Boolean formulas that contain only conjunction and disjunction connectives, but no negation.

expressive collusion-resistant AB-GKM scheme. However, such an approach suffers from some major drawbacks. Whenever the group dynamic changes, the rekeying operation requires to update the private keys given to existing members in order to provide backward/forward secrecy. This in turn requires establishing private communication channels with each group member which is not desirable in a large group setting. Further, in applications involving stateless members where it is not possible to update the initially given private keys and the only way to revoke a member is to exclude it from the public information, an ABE based approach does not work. Another limitation is that whenever the group membership policy changes, new private keys must be re-issued to members of the group. Our constructions address these shortcomings.

Our AB-GKM schemes are able to support a large variety of conditions over a set of attributes. When the group changes, the rekeying operations do not affect the private information of existing group members and thus our schemes eliminate the need of establishing expensive private communication channels. Our schemes provide the same advantage when the group membership conditions change. Furthermore, the group key derivation is very efficient as it only requires a simple vector inner product and/or polynomial interpolation. Additionally, our schemes are resistant to collusion attacks. Multiple group members are unable to combine their private information in a useful way to derive a group key which they cannot derive individually.

Our AB-GKM constructions are based on the ACV-BGKM (Access Control Vector BGKM) scheme [6], a provably secure BGKM scheme, and Shamir's threshold scheme [7]. In this paper, we construct three AB-GKM schemes each of which is more suitable over others under different scenarios. The first construction, inline AB-GKM, is based on the ACV-BGKM scheme. Inline AB-GKM supports arbitrary monotonic policies over a set of attributes. In other words, a user whose attributes satisfy the group policies is able to derive the symmetric group key. However, inline AB-GKM does not efficiently support d-*out-of*-m ($d \leq m$) attribute threshold policies over m attributes. The second construction, threshold AB-GKM, addresses this weakness. The third construction, access tree AB-GKM, is an extension of threshold AB-GKM and is the most expressive scheme. It efficiently supports arbitrary policies. The second and third schemes constructed by using a modified version of ACV-BGKM, also proposed in this paper.

The reminder of the paper is organized as follows: Section II discusses related work on GKM, attribute based encryption, GKM in selective dissemination/broadcast encryption, and secret sharing. Section III provides a summary of the ACV-BGKM scheme [6]. Sections IV, V, VI

TABLE I

ACRONYMS

Acronym	**Description**
GKM	Group Key Management
BGKM	Broadcast GKM
ABE	Attribute Based Encryption
ACV	Access Control Vector
ABAC	Attribute Based Access Control
AB-GKM	Attribute Based GKM
PI	Public Information tuple
UA	User-Attribute matrix

show the construction of the inline AB-GKM, threshold AB-GKM, and access tree AB-GKM schemes, respectively, and analyze their security and performance. Section VII shows an example application of the three AB-GKM schemes. Section VIII concludes the paper. Appendix **??** proves the security of the modified ACV-BGKM scheme. Table I list, for the convenience of the reader, the acronyms used in the paper.

II. RELATED WORK

Group Key Management: Group Key Management (GKM) is a widely investigated topic in the context of group-oriented multicast applications [8], [5]. Early approaches to GKM rely on a key server to share a secret with users to distribute decryption keys [9], [10]. Such approaches do not efficiently handle join and leave operations, as in order to achieve forward and backward security, they require sending $O(n)$ private rekey information, where n is the number of users.Hierarchical key management schemes [11], [12], where the key server hierarchically establishes secure channels with different sub-groups instead of individual users, were introduced to reduce this overhead. However, they only reduce the size of the rekey information to $O(\log n)$, and furthermore each user needs to manage at worst $O(\log n)$ hierarchically organized redundant keys.

Broadcast Group Key Management (BGKM) schemes perform the rekey operation with only one broadcast without affecting the secret information issued to existing users. Approaches have also been proposed to make the rekey operation a one-off process [13], [5]. However,

these schemes are not formally proven to be secure. Recently Shang et. al. introduced the first provably secure BGKM scheme ACV-BGKM [6]. Existing BGKM schemes require sending $O(n)$ public information when rekeying. We improve the complexity by utilizing subset-cover techniques [14], [15]. The improved BGKM schemes can efficiently handle group dynamics and lay the foundation for AB-GKM. However such schemes cannot directly handle expressive conditions against attributes.

Attribute-Based Encryption and GKM: The concept of attribute-based encryption (ABE), introduced by Sahai and Waters [2], can be considered as a generalization of identity based encryption [16], [17] (IBE). In an ABE system, the plaintext is encrypted with a set of attributes. The key generation server, which possesses the master key, issues different private keys to users after authenticating the attributes they possess. Thus, these private keys are associated with the set of attributes each user possesses. In its basic form, a user can decrypt a ciphertext if and only if there is a match between the attributes of the ciphertext and the user's key. The initial ABE system is limited to only threshold policies by which there should be at least k out of n attributes common between the attributes used to encrypt the plaintext and the attributes users possess. Pirretti et al. [18] gave an implementation of such a threshold ABE system using a variant of the Sahai-Waters Large Universe construction [2]. Since the definition of the initial threshold scheme, a few variants have been introduced to provide more expressive ABE systems. Goyal et al. [3] introduced the idea of key-policy ABE (KP-ABE) systems and Bethencourt et al. [4] introduced the idea of ciphertext-policy ABE (CP-ABE) systems. Even though these constructs are expressive and provably secure, they are unable to efficiently support group management, and especially to provide forward security when a user leaves the group (i.e. attribute revocation) and to provide backward security when a new user joins the group. These schemes require sending $O(n)$ private rekey messages in order to handle group management operations. The proposers of some of these schemes have suggested using an expiration attribute along with other attributes for attribute revocation. However, such a solution is not suitable for highly dynamic groups where joins and leaves are frequent. Traynor et. al. [19] proposes to improve the performance of ABE by grouping users and assigning a unique group attribute to each group. However, their approach only considers one attribute per user and does not support membership policy based group key management.

Despite the limitations of ABE schemes with respect to revocation, flat table based GKM

schemes [2] based on ABE have been proposed [20], [21]. These schemes further suffer from the inherent limited expressibility and scalability of flat table based GKM [22], [23].

GKM Schemes for Selective Dissemination Systems: Selective dissemination or broadcast encryption systems allow one to encrypt a message once and broadcast to all the users in a group, but only a subset of users who have the correct key can decrypt the message. The database and security communities have carried out extensive research concerning techniques for the selective dissemination of documents based on access control policies with their own GKM schemes [24], [25], [26], [27], [28]. In such approaches, users are able to decrypt the subdocuments, that is, portions of documents, for which they have the keys. However, such approaches require all [25] or some [26] keys be distributed in advance during user registration phase. This requirement makes it difficult to assure forward and backward key secrecy when user groups are dynamic with frequent join and leave operations. Further, the rekey operation is not transparent, thus shifting the burden of acquiring new keys on existing users when others leave or join. Thus the proposed GKM schemes are not efficient. In contrast, our GKM schemes make rekey transparent to users by not distributing actual keys. We remark that the efficiency of the existing selective dissemination systems can be vastly improved by utilizing our GKM schemes.

Secret Sharing Schemes: These schemes split a shared secret among a group of users by giving secret shares to users and allow them to combine their secrets in a specific way and obtain the shared secret. Shamir [7] proposed the first secret sharing scheme, (n, k)-threshold scheme, where k users out of n can construct a unique polynomial $f(x)$ of degree $k-1$ and recover the shared secret $f(0)$. Since the definition of this initial scheme, there have been numerous extensions [29], [30], [31]. A major difference between GKM protocols and secret sharing schemes is that the former are designed to allow any individual group member to obtain a shared secret by itself, and no persistent secure communication channel is assumed between valid group members, whereas the latter are to prevent a single group member from gaining the secret alone, and require a secure communication channel, when group members combine the secret shares, to protect the shared secret from being learned by parties outside the group.

[2]A flat table GKM scheme assigns each member a unique n-bit string. The group key is managed through a set of auxiliary keys which are tied to the unique strings given to the group members.

III. BACKGROUND

In this section, we provide an overview of the Broadcast Group Key Management (BGKM) scheme in general and a description of a provably secure BGKM scheme called ACV-BGKM (Access Control Vector BGKM) proposed by Shang et al. [6], [32] in order for readers to better understand our constructions. It should be noted that we use ACV-BGKM in Section IV and a modified version of ACV-BGKM in our constructions in Sections V, VI.

BGKM schemes are a special type of GKM scheme where the rekey operation is performed with a single broadcast without using private communication channels. Unlike conventional GKM schemes, BGKM schemes do not give users the private keys. Instead users are given a secret which is combined with public information to obtain the actual private keys. Such schemes have the advantage to require a private communication only once for the initial secret sharing. The subsequent rekeying operations are performed using one broadcast message. Further, in such schemes achieving forward and backward security requires only to change the public information and does not affect the secret shares given to existing users. In general, a BGKM scheme consists of the following five algorithms:

Setup(ℓ): It initializes the BGKM scheme using a security parameter ℓ. It also initializes the set of used secrets **S**, the secret space $\mathcal{SS}$, and the key space $\mathcal{KS}$.

SecGen(): It selects a random bit string $s \notin$ **S** uniformly at random from the secret space $\mathcal{SS}$, adds s to **S** and outputs s.

KeyGen(S): It chooses a group key k uniformly at random from the key space $\mathcal{KS}$ and outputs the public information tuple PI computed from the secrets in **S** and the group key k.

KeyDer(s, PI): It takes the user's secret s and the public information PI to output the group key. The derived group key is equal to k if and only if $s \in$ **S**.

Update(S): Whenever the set **S** changes, a new group key k' is generated. Depending on the construction, it either executes the **KeyGen** algorithm again or incrementally updates the output of the last **KeyGen** algorithm.

Using the above abstract algorithms, we now provide an overview of the construction of the ACV-BGKM scheme under a client-server architecture. The ACV-BGKM scheme satisfies the requirements of *minimal trust*, *key indistinguishability*, *key independence*, *forward secrecy*, *backward secrecy* and *collusion resistance* [8]. The ACV-BGKM algorithms are executed by a trusted key server Svr and a group of users $\mathsf{Usr}_i, i = 1, 2, \ldots, n$.

Setup(ℓ): Svr initializes the following parameters: an ℓ-bit prime number q, the maximum group size N ($\geq n$ and N is usually set to $n+1$), a cryptographic hash function $H(\cdot) : \{0,1\}^* \rightarrow \mathbb{F}_q$, where $\mathbb{F}_q$ is a finite field with q elements, the keyspace $\mathcal{KS} = \mathbb{F}_q$, the secret space $\mathcal{SS} = \{0,1\}^\ell$ and the set of issued secrets $\mathbf{S} = \emptyset$.

SecGen(): Svr chooses the secret $s_i \in \mathcal{SS}$ uniformly at random for Usr_i such that $s_i \notin \mathbf{S}$, adds s_i to $\mathbf{S}$ and finally outputs s_i.

KeyGen(S): Svr picks a random $k \in \mathcal{KS}$ as the group key. Svr chooses N random bit strings $z_1, z_2, \ldots, z_N \in \{0,1\}^\ell$. Svr creates an $n \times (N+1)$ $\mathbb{F}_q$-matrix

$$A = \begin{pmatrix} 1 & a_{1,1} & a_{1,2} & \ldots & a_{1,N} \\ 1 & a_{2,1} & a_{2,2} & \ldots & a_{2,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_{n,1} & a_{n,2} & \ldots & a_{n,N} \end{pmatrix},$$

where

$$a_{i,j} = H(s_i||z_j), 1 \leq i \leq n, 1 \leq j \leq N, s_i \in \mathcal{S}. \tag{1}$$

Svr then solves for a nonzero $(N+1)$-dimensional column $\mathbb{F}_q$-vector Y such that $AY = 0$. Note that such a nonzero Y always exists as the nullspace of matrix A is nontrivial by construction. Here we require that Svr chooses Y from the nullspace of A uniformly at random. Svr constructs an $(N+1)$-dimensional $\mathbb{F}_q$-vector

$$ACV = k \cdot e_1^T + Y,$$

where $e_1 = (1, 0, \ldots, 0)$ is a standard basis vector of $\mathbb{F}_q^{N+1}$, v^T denotes the transpose of vector v, and k is the chosen group key. The vector ACV controls the access to the group key k and is called an *access control vector*. Svr lets

$$PI = \langle ACV, (z_1, z_2, \ldots, z_N) \rangle,$$

and outputs public PI and private k.

KeyDer(s_i, PI): Using its secret s_i and the public information tuple PI, Usr_i computes $a_{i,j}, 1 \leq j \leq N$, as in formula (1) and sets an $(N+1)$-dimensional row $\mathbb{F}_q$-vector

$$v_i = (1, a_{i,1}, a_{i,2}, \ldots, a_{i,N}).$$

v_i is called a Key Extraction Vector (KEV) and corresponds to a unique row in the access control matrix A. Usr_i derives the key k' from the inner product of v_i and ACV:

$$k' = v_i \cdot ACV.$$

The derived key k' is equal to the actual group key k if and only if s_i is a valid secret used in the computation of PI, i.e., $s_i \in \mathbf{S}$.

Update(S): It runs the **KeyGen(S)** algorithm and outputs the new public information PI' and the new group key k'.

The above construction becomes impractical with large number of users since the complexity of the matrix and the public information is $O(n)$. We proprose two approaches to improve the complexity in Section IV-C without keeping the underlying scheme unchanged.

IV. Scheme 1: Inline AB-GKM

Recall that in its basic form, a BGKM scheme can be considered as a 1-*out-of-*m AB-GKM scheme. If Usr_i possesses the attribute attr_j, Svr shares a unique secret $s_{i,j}$ with Usr_i. Usr_i is thus able to derive the symmetric group key if and only if Usr_i shares at least one secret with Svr and that secret is included in the computation of the public information tuple PI. In order for Svr to revoke Usr_j, it only needs to remove the secrets it shares with Usr_j from the computation of PI; the secrets issued to other group members are not affected. We extend this scheme to support arbitrary monotonic policies, Ps, over a set of attributes. A user is able to derive the symmetric group key if and only if the set of attributes the user possesses satisfy P.

As in the basic BGKM scheme, Usr_i having attr_j is associated with a unique secret value $s_{i,j}$. However, unlike the basic BGKM scheme, PI is generated by using the aggregated secrets that are generated combining the secrets issued to users according to P. For example, if P is a conjunction of two attributes, that is $\mathsf{attr}_r \wedge \mathsf{attr}_s$, the corresponding secrets $s_{i,r}$ and $s_{i,s}$ for each Usr_i are combined as one aggregated secret $s_{i,r}||s_{i,s}$ and PI is computed using these aggregated secrets. By construction, the aggregated secrets are unique since the constituent secrets are unique. Any Usr_i is able to derive the symmetric group key if and only if Usr_i has at least one aggregated secret used to compute PI. Notice that multiple users cannot collude to create an aggregated secret which they cannot individually create since $s_{i,j}$'s are unique and each aggregated secret is tied to one specific user. Hence, colluding users cannot derive the group

symmetric key. Now we give a detailed description of our first AB-GKM scheme, inline AB-GKM.

A. Our construction

Inline AB-GKM consists of the following five algorithms:

Setup(ℓ): The Svr initializes the following parameters: an ℓ-bit prime number q, a cryptographic hash function $H(\cdot) : \{0,1\}^* \rightarrow \mathbb{F}_q$, where $\mathbb{F}_q$ is a finite field with q elements, the keyspace $\mathcal{KS} = \mathbb{F}_q$, the secret space $\mathcal{SS} = \{0,1\}^\ell$, and the set of issued secrets $\mathbf{S} = \emptyset$. The user-attribute matrix UA is initialized with empty elements and the maximum group size N is decided in the **KeyGen**. It defines the universe of attributes $\mathcal{A} = \{\mathsf{attr}_1, \mathsf{attr}_2, \cdots, \mathsf{attr}_m\}$.

SecGen(Usr$_i$, attr$_j$): The Svr chooses the secret $s_{i,j} \in \mathcal{SS}$ uniformly at random for Usr_i such that $s_{i,j} \notin \mathbf{S}$, adds $s_{i,j}$ to **S**, sets $UA(i,j) = s_{i,j}$, where $UA(i,j)$ is the $(i,j)^{th}$ element of the user-attribute matrix UA, and finally outputs $s_{i,j}$.

KeyGen(P): We first give a high-level description of the algorithm and then the details. Svr transforms the policy P to disjunctive normal form (DNF). For each disjunctive clause of P in DNF, it creates an aggregated secret ($\widehat{s}$) from the secrets corresponding to each of the attributes in the conjunctive clause. $\widehat{s}$ is formed by concatenation only if there exists secrets for all the attributes in a given row of the user-attribute matrix UA. The construction creates a unique aggregated secret $\widehat{s}$ since the corresponding secrets are unique.

For example, if the conjunctive clause is $\mathsf{attr}_p \wedge \mathsf{attr}_q \wedge \mathsf{attr}_r$, for each row i in UA, the aggregated secret $\widehat{s}_i$ is formed only if all elements $UA(i,p)$, $UA(i,q)$ and $UA(i,r)$ have secrets assigned.

All the aggregated secrets are added to the set **AS**. Finally, Svr invokes algorithm **KeyGen**(**AS**) from the underlying BGKM scheme to output the public information PI and the symmetric group key k.

Now we give the details of the algorithm. Svr converts P to DNF as follows

$$\mathsf{P} = \bigvee_{i=1}^{\alpha} \mathsf{conjunct}_i$$

where

$$\mathsf{conjunct}_i = \bigwedge_{j=1}^{\beta} \mathsf{cond}_j^{(i)}$$

A simple multiplication of clauses ($x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$) and then application of the absorption law ($x \vee (x \wedge y = x)$) are sufficient to convert monotone policies to DNF. Even though there can be an exponential blow up of clauses during multiplication, it has been shown that with the application of the absorption law the number of clauses in the DNF, at the end, is always polynomially bounded. Svr selects N such that

$$N \geq \sum_{i=1}^{\alpha} N_{U_i} = m$$

where N_{U_i} is the number of users satisfying $\mathsf{conjunct}_i$. Svr creates m $\widehat{s}_i$'s and adds them to **AS**. Svr picks a random $k \in \mathcal{KS}$ as the shared group key. Svr chooses N random bit strings $z_1, z_2, \ldots, z_N \in \{0,1\}^{\ell}$. Svr creates an $m \times (N+1)$ $\mathbb{F}_q$-matrix A such that for $1 \leq i \leq m$

$$a_{i,j} = \begin{cases} 1 & \text{if } j = 1 \\ H(\widehat{s}_i || z_j) & \text{if } 2 \leq j \leq N; \widehat{s}_i \in \mathbf{AS} \end{cases}$$

Svr then solves for a nonzero $(N+1)$-dimensional column $\mathbb{F}_q$-vector Y such that $AY = 0$ and sets

$$ACV = k \cdot e_1^T + Y, \text{ and}$$

$$PI = \ ACV, (z_1, z_2, \ldots, z_N)$$

KeyDer(β_i, PI): Given β_i, the set of secrets for Usr_i, it computes the aggregated secret $\widehat{s}$. Using $\widehat{s}$ and the public information PI, it computes $a_{i,j}, 1 \leq j \leq N$, as in IV-A and sets an $(N+1)$-dimensional row $\mathbb{F}_q$-vector $v_i = (1, a_{i,1}, a_{i,2}, \ldots, a_{i,N})$. Usr_i derives the group key k' by the inner product of the vectors v_i and ACV:

$$k' = v_i \cdot ACV.$$

The derived group key k' is equal to the actual group key k if and only if the computed aggregated secret $\widehat{s} \in$ **AS**.

Update(S): If a user leaves or join the group, a new symmetric group key k' is selected. **KeyGen(S)** is invoked to generate the updated public information PI'. Notice that the secrets shared with existing users are not affected by the group change. It outputs the public PI' and private k'.

B. Security and Performance

We can easily show that if an adversary $\mathcal{A}$ can break inline AB-GKM scheme in the random oracle model, a simulator $\mathcal{S}$ can be constructed to break ACV-BGKM scheme.

The intuition behind is that under the random oracle model, the inline AB-GKM scheme is an instance of the ACV-BGKM scheme as follows. The only difference between the ACV-BGKM and inline AB-GKM schemes is that the former is based on unique secrets whereas the latter is based on aggregated secrets. Since each aggregated secret is constructed by concatenating unique secrets, each aggregated secret is unique. For each unique aggregated secret, under the random oracle model, the cryptographic hash function H produces a unique value. It is equivalent to producing unique values using H under the ACV-BGKM scheme using unique secrets. This also implies that the aggregated secrets are as random as individual secrets and knowing a subset of secrets for an aggregated secret does not give an advantage for $\mathcal{A}$ for guessing the aggregated secret. Therefore, an instance of inline AB-GKM is essentially equivalent to an instance of ACV-BGKM. Hence, if $\mathcal{A}$ breaks inline AB-GKM in the random oracle model with advantage ϵ, we can build a trivial simulator $\mathcal{S}$ that breaks ACV-BGKM without knowing any secret used to construct the public information tuple. Shang et al. [6], [32] have shown that the probability of breaking ACV-BGKM is a negligible $1/q$, where q is the ℓ bit large prime number initialized in **Setup**.

Now, we discuss the efficiency of inline AB-GKM with respect to computational costs and required bandwidth for rekeying.

For any Usr_i in the group, deriving the shared group key requires N hashing operations (evaluations of $H(\cdot)$) and an inner product computation $v_i \cdot ACV$ of two $(N+1)$-dimensional $\mathbb{F}_q$-vectors, where N is the maximum group size. Therefore the overall computational complexity is $O(n)$. In practice, this can be done very efficiently.

For every rekeying operation, Svr needs to form a matrix A by performing N^2 hashing operations, and then solve a linear system of size $N \times (N+1)$. Solving the linear system is the most costly operation as N gets large for computation on Svr. It requires $O(n^3)$ field operations in $\mathbb{F}_q$ when the method of Gauss-Jordan elimination [33] is applied. Experimental results about the ACV-BGKM scheme [6] have shown that this can be performed in a short time when N is up to 1000. In Section IV-C, we propose approaches to improve the complexity of the BGKM

scheme.

When a rekeying process takes place, the new information to be broadcast is $PI = \ ACV, (z_1, \ldots, z_N)$, where ACV is a vector consisting of $(N+1)$ elements in $\mathbb{F}_q$, and without loss of generality we can pick z_i to be strings with a fixed length. This gives an overall communication complexity $O(n)$. An advantage of inline AB-GKM is that the no peer-to-peer private channel is needed for any persisting group members when rekeying happens.

Nowadays we generally care less about storage costs on both Svr and Usrs. Nevertheless, for a group of maximum N users, in the worst case, inline AB-GKM only requires each Usr to store ($O(|\mathcal{A}|)$) secrets, one secret per attribute that Usr possesses, and Svr to keep track of all $O(n|\mathcal{A}|)$ secrets.

C. Improving the Complexity

One disadvantage of the above construction is that size of the matrix A created during **KeyGen** becomes very large as the number of users satisfying each conjunctive clause increases and can go up to $N = \alpha n$, where α is the number of conjunctive clauses in P and n the number of users in the system. This increases the computational cost at Svr to solve the linear system associated with A. Further, it requires to solve the complete system every time **Update** operation is invoked and to send $O(n)$ public rekey information. We propose two approaches to improve the complexity: two layer approach and subset-cover approach.

1) Two Layer Approach: We suggest a two layer approach to improve the performance of **KeyGen** and **Update** operations based on the following two observations:

1) Due to the non-linear cost associated with solving a linear system, we can reduce the overall computational cost by breaking the linear system in to a set of smaller linear systems.
2) When the group dynamic changes, in general, it only changes some U_i's, where U_i is the set of users satisfying $\mathsf{conjunct}_i$. Therefore, we can isolate the change to some linear systems without affecting the complete system.

We give a high-level description of the two layer approach. During **KeyGen**, instead of creating one single matrix A, Svr creates a separate matrix A_i for each $\mathsf{conjunct}_i$, $1 \leq i \leq \alpha$. Svr solves each linear system A_i of size $N_{U_i} \times (N_{U_i} + 1)$ and obtain the public information tuple PI_i and an intermediate key k_i. Notice that each linear system is independent and the computation can

easily be parallelized to further improve the performance. Using the set of intermediate keys $\{k_i | 1 \leq i \leq \alpha\}$ as the set of secrets, Svr generates a parent matrix A of much smaller size $\alpha \times (\alpha + 1)$ and computes the corresponding public PI associating the group key k.

During **KeyDer**, Usr first obtains the intermediate key k_j for the $\mathsf{conjunct}_j$ that she satisfies using her secret(s) and PI_j. Then, using k_j and PI, Usr derives the group key k.

During **Update**, Svr first identifies which U_i's, the user groups, have changed, and then update only the corresponding PI_i's as well as PI setting a new group key k'.

2) Subset-Cover Approach: The two layer approach presented above in Section IV-C.1 also becomes inefficient if each N_{U_i} is large. This is due to the fact that the computational and communication complexities are still proportional to N_{U_i} values. We utilize the result from previous research on broadcast encryption [14], [15] to improve the complexities. Based on that, one can make the complexities sub-linear in the number of users by giving more than one secret during **SecGen** for each attribute they possess. The secrets given to each user overlaps with different subsets of users. During the **KeyGen**, Svr identifies the minimum number of subsets to which all the users belong and uses one secret per identified subset. During **KeyDer**, a user identifies the subset it belongs to and uses the corresponding secret to derive the group key. Group dynamics are handled by making some of the secrets given to users invalid.

We give a high-level description of the basic subset-cover approach. More details are available in the technical report. In the basic scheme, n users are organized as the leaves of a balanced binary tree of height $\log n$. A unique secret is assigned to each vertex in the tree. Each user is given $\log n$ secrets that correspond to the vertices along the path from its leaf node to the root node. In order to provide provide forward secrecy when a single user is revoked, the updated tree is described by $\log n$ subtrees formed after removing all the vertices along the path from the user leaf node to the root node. To rekey, Svr executes **Update** including $\log n$ secrets corresponding the roots of these subtrees. Naor et. al. [14] improve this technique to simultaneously revoke r users and describe the exiting users using $r \log (n/r)$ subtrees. Since then, there has been many improvements to the basic scheme. In the remainder of the paper, in order to maintain the simplicity we only provide one secret per attribute but the schemes can be trivially modified to use subset-cover approach.

V. SCHEME 2: THRESHOLD AB-GKM

Consider now the case of policies by which a user can derive the symmetric group key k, if it possesses at least d attributes out of the m attributes associated with the group. We refer to such policies as *threshold policies*. Under the inline AB-GKM scheme presented in Section IV, with such threshold policies the size of the access control matrix (A) increases exponentially. Specifically, to support d-*out-of*-m, the inline AB-GKM scheme may require creating a matrix of dimension up to $O(nm^d)$ where n is the number of users in the group. Thus, the inline AB-GKM scheme is not suitable for threshold policies. In this section, we construct a new scheme, threshold AB-GKM, which overcomes this shortcoming.

An initial construction to enforce threshold policies is to associate each user with a random $d-1$ degree polynomial, $q(x)$, with the restriction that each polynomial has the same value at $x = 0$ and $q(0) = k$, where k is the symmetric group key. For each attribute users have, they are given a secret value. The secret values given to a user are tied to its random polynomial $q(x)$. A user having d or more secrets can perform a Lagrange interpolation to obtain $q(x)$ and thus the symmetric group key $k = q(0)$. Since the secrets are tied to random polynomials, multiple users are unable to combine their secrets in any way that makes possible collusion attacks. However, revocation is difficult in this simple approach and requires re-issuing all the secrets again.

Our approach to address the revocation problem is to use a layer of indirection between the secrets given to users and the random polynomials such that revocations do not require re-issuing all the secrets again. We use a modified ACV-BGKM construction as the indirection layer. We cannot directly use the ACV-BGKM construction since, when multiple instances of ACV-BGKM are utilized, it does not prevent collusion attacks in which colluding users can recover the group key which they cannot obtain individually. We first show the details of the modified ACV-BGKM scheme and then present the threshold AB-GKM which uses the modified ACV-BGKM scheme and Shamir's secret sharing scheme.

A. Modified ACV-BGKM Scheme

The modified ACV-BGKM works under similar conditions as ACV-BGKM, but instead of giving the same key k to all the users, **KeyDer** algorithm gives each Usr_i a different key k_i when the public information tuple PI is combined with their unique secret s_i.

The algorithms are executed with a trusted key server Svr and a group of users Usr_i, $i = 1, 2, \cdots, n$ with the attribute universe $\mathcal{A} = \{\mathsf{attr}_1, \mathsf{attr}_2, \cdots, \mathsf{attr}_m\}$. The construction is as follows:

Setup(ℓ): Svr initializes the following parameters: an ℓ-bit prime number q, the maximum group size N ($\geq n$), a cryptographic hash function $H(\cdot) : \{0,1\}^* \rightarrow \mathbb{F}_q$, where $\mathbb{F}_q$ is a finite field with q elements, the key space $\mathcal{KS} = \mathbb{F}_q$, the secret space $\mathcal{SS} = \{0,1\}^\ell$ and the set of issued secret tuples $\mathbf{S} = \emptyset$. Each Usr_i is given a unique secret index $1 \leq i \leq N$.

SecGen(): The Svr chooses the secret $s_i \in \mathcal{SS}$ uniformly at random for Usr_i such that s_i is unique among all the users, adds the secret tuple $\langle i, s_i \rangle$ to **S**, and outputs $\langle i, s_i \rangle$.

KeyGen(S, K): Given the set of secret tuples $\mathbf{S} = \{ \langle i, s_i \rangle | 1 \leq i \leq N\}$ and a random set of keys $\mathbf{K} = \{k_i | 1 \leq i \leq N\}$, it outputs the public information tuple PI which allows each Usr_i to derive the key k_i using their secret s_i. The details follow.

Svr chooses N random bit strings $z_1, z_2, \ldots, z_N \in \{0,1\}^\ell$ and creates an $N \times 2N$ $\mathbb{F}_q$-matrix A where for a given row i, $1 \leq i \leq N$

$$a_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } 1 \leq j \leq N \text{ and } i \neq j \\ H(s_i || z_j) & \text{if } N < j \leq 2N \end{cases}$$

Like in ACV-BGKM protocol, Svr computes the null space of A with a set of its N basis vectors, and selects a vector Y as one of the basis vectors. Svr constructs an $2N$-dimensional $\mathbb{F}_q$-vector

$$ACV = (\sum_{i=1}^{N} k_i \cdot e_i^T) + Y,$$

where e_i is the i^{th} standard basis vector of $\mathbb{F}_q^{2N}$. Notice that, unlike ACV-BGKM, a unique key corresponding to Usr_i, $k_i \in \mathbf{K}$ is embedded into each location corresponding to a valid index i. Like, ACV-BGKM, Svr sets $PI = \langle ACV, (z_1, z_2, \ldots, z_N) \rangle$, and outputs PI via the broadcast channel.

KeyDer(s_i, PI): Usr_i, using its secret s_i and public PI, derives the $2N$-dimensional row $\mathbb{F}_q$-vector v_i which corresponds to a row in A. Then Usr_i derives the specific key as $k_i = v_i \cdot ACV$.

Update(S, K'): If a user leaves or join the group, a new set of keys K' is selected. **KeyGen(S, K')** is invoked to generate the updated public information PI'. Notice that the secrets shared with existing users are not affected by the group change. It outputs the public PI'.

1) Security of Modified ACV-BGKM: In this section, we prove the security of the modified ACV-BGKM scheme. Specifically we prove the soundness of the modified ACV-BGKM scheme. We will model the cryptographic hash function H as a random oracle. We further assume that $q = O(2^{\ell})$ is a sufficiently large prime power and N is relatively small. We first present two lemmas with their proofs and then prove that the modified ACV-BGKM scheme is indeed sound.

The following lemmas are useful for proving the security of the modified ACV-BGKM. Lemma 1 says that in a vector space V over a large finite field, the probability that a randomly chosen vector is in a pre-selected subspace, strictly smaller than V, is very small. Lemma 2, which uses Lemma 1, is used to prove the soundness of the modified ACV-BGKM scheme.

Lemma 1: Let $F = \mathbb{F}_q$ be a finite field of q elements. Let V be an n-dimensional F-vector space, and W be an m-dimensional F-subspace of V, where $m \leq n$. Let v be an F-vector uniformly randomly chosen from V. Then the probability that $v \in W$ is $1/q^{n-m}$.

Proof: The proof is straightforward. We show it here for completeness. Let $\{v_1, v_2, \ldots, v_m\}$ be a basis of W. Then it can be extended to a basis of V by adding another $n-m$ basis vector $v_{m+1}, \ldots, v_n$. Any vector $v \in V$ can be written as

$$v = \alpha_1 \cdot v_1 + \ldots + \alpha_n \cdot v_n, \quad \alpha_i \in F, 1 \leq i \leq n,$$

and $v \in W$ if and only if $\alpha_i = 0$ for $m+1 \leq i \leq n$. When v is uniformly randomly chosen from V, it follows

$$\Pr[v \in W] = 1/q^{n-m}.$$

■

Lemma 2: Let $F = \mathbb{F}_q$ be a finite field of q elements. Let $v_i = e_i^T + (0, \ldots, 0, v_i^{(n+1)}, \ldots, v_i^{(2n)})$, e_i is the i^{th} standard basis vector of $\mathbb{F}_q^{2n}, i = 1, \ldots, m$, and $1 \leq m \leq n$, be $2n$-dimensional F-vectors. Let $v = e^T + (0, \ldots, 0, v^{(n+1)}, \ldots, v^{(2n)})$ be a $2n$-dimensional F-vector with $v^{(j)}, j \geq n+1$ chosen independently and uniformly at random from F and e from the $2n$-dimensional standard basis vectors with the position of the non-zero element $\leq m$. Then the probability that v is linearly dependent of $\{v_i,\ 1 \leq i \leq m\}$ is no more than $1/q^{n-m}$.

Proof: Let $w_i = (v_i^{(n+1)}, \ldots, v_i^{(2n)}), 1 \leq i \leq m$, $w = (v^{(n+1)}, \ldots, v^{(2n)})$, and $u_i = (v_i^{(1)}, \ldots, v_i^{(n)})$. All w_i span an F-subspace W whose dimension is at most m in an n-dimensional

F-vector space. w and u are uniformly randomly chosen n-dimensional F-vectors. By Lemma 1,

$$\Pr[w \in W] = 1/q^{n-\dim(W)} \leq 1/q^{n-m}.$$

It follows that

$$\begin{aligned}
&\Pr[v \text{ is linearly dependent of } \{v_i : 1 \leq i \leq m\}] \\
&= \Pr[v = \alpha_1 \cdot v_1 + \ldots + \alpha_m \cdot v_m \text{ for some } \alpha_i \in F] \\
&= \Pr\left[\sum_{i=1}^{m} \alpha_i \cdot u_i = e^T \wedge w = \sum_{i=1}^{m} \alpha_i \cdot v_i \text{ for some } \alpha_i \in F\right] \\
&= \Pr\left[\sum_{i=1}^{m} \alpha_i \cdot u_i = e^T\right] \cdot \Pr[w \in W] \\
&\leq 1/q^n \cdot 1/q^{n-m} = 1/q^{2n-m}.
\end{aligned}$$

■

Definition 1 (Soundness of the modified ACV-BGKM scheme): Let Usr_i be an individual without a valid secret and Usr_j with a valid secret s_j, $1 \leq i, j \leq N$. The modified ACV-BGKM is *sound* if

- The probability that Usr_i can obtain the correct key k_i by substituting the secret with a value val that is not one of the valid secrets and then running the key derivation algorithm **KeyDer** is negligible.
- The probability that Usr_j can obtain a correct key k_r, where $j = r$ and $1 \leq r \leq N$, by substituting s_j and then running the key derivation algorithm **KeyDer** is negligible.

Theorem 1: The modified ACV-BGKM scheme is sound.

Proof: Let $PI = \ ACV, (z_1, \ldots, z_N)$ be the public information broadcast from Svr.

Case 1: Usr_i does not have a valid secret and tries to derive k_i.

Let Y be a vector orthogonal to the access control matrix A.

Let

$$\{v_i, 1 \leq i \leq N\}$$

be a basis of the nullspace of Y.

Let

$$v = e^T + (0, \ldots, 0, v^{(N+1)}, \ldots, v^{(2N)}),$$

where

$$v^{(i+N)} = H(\mathsf{val}||z_i), 1 \leq i \leq N.$$

Usr_i can derive the key using v by running the **KeyDer** algorithm if and only if v is linearly dependent from $v_i, 1 \leq i \leq N$. When val is not a valid secret and H is a random oracle, v is indistinguishable from a vector whose first N entries are from e^T and the rest of the N entries are independently and uniformly chosen from $\mathbb{F}_q$. By Lemma 2, the probability that v is linearly dependent from $\{v_i, 1 \leq i \leq N\}$ is no more than $1/q^{2N-N} = 1/q^N$, which is negligible. This proves that the modified ACV-BGKM scheme is sound in case 1.

Case 2: Usr_j has a valid secret s_j and tries to derive k_r, where $r = j$ and $1 \leq r \leq N$.

Since Usr_j has a valid secret s_j, it can construct the j^{th} row of A as follows:

$$v_j = e_j^T + (0, \ldots, 0, v_j^{(N+1)}, \ldots, v_j^{(2N)}),$$

where

$$v_j^{(i+N)} = H(s_j||z_i), 1 \leq i \leq N.$$

Usr_j can obtain the key k_j using v_j:

$$k_j = ACV \cdot v_j.$$

In order to obtain the key k_r, Usr_j needs to compute $ACV \cdot v_r$ where v_r is defined as follows.

$$v_r = e_r^T + (0, \ldots, 0, v_r^{(N+1)}, \ldots, v_r^{(2N)}),$$

where

$$v_r^{(i+N)} = H(\mathsf{val}||z_i), 1 \leq i \leq N.$$

By construction, v_r is linearly independent from v_j. When val is not a valid secret and H is a random oracle, v_r is indistinguishable from a vector whose first N entries are from e_r^T and the rest of the N entries are independently and uniformly chosen from $\mathbb{F}_q$. Thus, knowing v_j does not provide an advantage for Usr_j to compute v_r. Therefore, the probability of deriving k_r by running the **KeyDer** algorithm remains the same negligible value $1/q^N$ as in case 1. This proves that the modified ACV-BGKM scheme is sound in case 2.

■

B. Our Construction

Now we provide our construction of the threshold AB-GKM scheme which utilizes the modified ACV-BGKM scheme.

Recall that in this protocol, we wish to allow a user to derive the symmetric group key k if the user possesses at least d attributes out of m. For each user Usr_i we associate a random $d-1$ degree polynomial $q_i(x)$ with the restriction that each polynomial has the same value k, the symmetric group key, at $x=0$, that is, $q_i(0)=k$. We associate a random secret value with each user attribute. For each attribute attr_i, we generate a public information tuple (PI_i) using the modified ACV-BGKM scheme with the restriction that the temporary key that each Usr_j derives is tied to their random polynomial $q_j(x)$, that is $q_j(i)=k_i$. Notice that each user obtains different temporary keys from the same PI. If a user can derive d temporary keys corresponding to d attributes, it can compute its random function $q(x)$ and obtain the group symmetric key k. Notice that, since the temporary keys are tied to a unique polynomial, multiple users are unable to collude and combine their temporary keys in order to obtain the symmetric group key which they are not allowed to obtain individually. Thus, our construction prevents collusion attacks.

A detailed description of our threshold AB-GKM scheme follows.

Setup(ℓ, d) Svr initializes the parameters of the underlying modified ACV-BGKM scheme: the ℓ-bit prime number q, the maximum group size N ($\leq n$), the cryptographic hash function H, the key space $\mathcal{KS}$, the secret space $\mathcal{SS}$, the set of issued secrets **S**, the user-attribute matrix UA and the universe of attributes $\mathcal{A}=\{\mathsf{attr}_1, \mathsf{attr}_2, \cdots, \mathsf{attr}_m\}$. Svr sets the degree of the random polynomial assigned to each user to be $d-1$.

Svr defines the Lagrange coefficient $\Delta_{i,\mathbf{Q}}$ for $i \in \mathbb{F}_q$ and a set, **Q** of elements in $\mathbb{F}_q$:

$$\Delta_{i,\mathbf{Q}}(x) = \prod_{j \in \mathbf{Q}, j \neq i} \frac{x-j}{i-j}.$$

SecGen(γ_i) For each attribute $\mathsf{attr}_j \in \gamma_i$, where $\gamma_i \subset \mathcal{A}$, Svr invokes **SecGen()** of the modified ACV-BGKM scheme in order to obtain the random secret $s_{i,j}$. It returns β_i, the set of secrets for all the attributes in γ_i.

KeyGen(α) For each user Usr_i, Svr assigns a random degree $d-1$ polynomial $q_i(x)$ with $q_i(0)$ set to the group symmetric key k. For each attribute attr_j in the set of attributes α ($\alpha \subset \mathcal{A}$ and $|\alpha| \geq d$), it selects the set of secrets corresponding to attr_j, $\mathbf{S}_j$ and invokes

KeyGen(S_j**,** $\{q_1(j), q_2(j), \cdots, q_N(j)\}$**)** of the modified ACV-BGKM scheme to obtain PI_j, the public information tuple for attr_j. It outputs the private group key k and the set of public information tuples **PI** = $\{PI_j|$ for each $\mathsf{attr}_j \in \alpha\}$.

KeyDer(β_i**, PI)** Using the set of d secrets $\beta_i = \{s_{i,j}|1 \leq j \leq N\}$ for the d attributes attr_j, $1 \leq j \leq N$, and the corresponding d public information tuples $PI_j \in$ **PI**, $1 \leq j \leq N$, it derives the group symmetric key k as follows.

First, it derives the temporary key k_j for each attribute attr_j using the underlying modified ACV-BGKM scheme as **KeyDer(**$s_{i,j}$, PI_j**)**. Then, using the set of d points $\mathbf{Q}_i = \{(j, k_j)|1 \leq j \leq N\}$, it computes $q_i(x)$ as follows:

$$\Delta_{j,\mathbf{Q}_i}(x) = \prod_{j \in \mathbf{Q}_i, j \neq i} \frac{x-j}{i-j}$$

$$q_i(x) = \sum_{j \in \mathbf{Q}_i} k_j \Delta_{j,\mathbf{Q}_i}(x).$$

It outputs the group key $k = q_i(0)$.

Update() If a user leaves or joins the group, a new symmetric group key k' is selected. The **Update** method of the underlying modified ACV-BGKM scheme is invoked to generate the updated public information. Notice that the secrets shared with existing users are not affected by the group change.

C. Security and Performance

If an adversary can break our threshold AB-GKM scheme, a simulator can be constructed to break the modified ACV-BGKM scheme. We give a high-level detail of the reduction based proof.

Suppose an adversary $\mathcal{A}$ having a set of $d-1$ attributes α can break our scheme in the random oracle model with advantage ϵ. We build a simulator $\mathcal{S}$ that can derive the key k_d from PI_d corresponding to $\mathsf{attr}_d \notin \alpha$ with advantage ϵ. In other words, we build a simulator to break the modified ACV-BGKM scheme.

The intuition behind our proof is that, by construction, the modified ACV-BGKM instances corresponding to the attributes are independent. In other words, a user who can access the key

for one attribute does not have any advantage in obtaining the key for another attribute using the known attribute.

$\mathcal{S}$ runs **Setup** of the threshold AB-GKM scheme. $\mathcal{A}$ obtains secrets $\{s_i | i = 1, 2, \cdots, d-1\}$ for the attributes α it has. $\mathcal{S}$ constructs the public information tuples $\{PI_i | i = 1, 2, \cdots, d-1\}$, each having a random key k_i corresponding to a random degree $d-1$ polynomial $q(x)$ and sends them along with PI_d to $\mathcal{A}$. $\mathcal{A}$ outputs k, which is equal to $q(0)$. This allows $\mathcal{S}$ to fully determine $q(x)$ as it now has d points and derive the key $k_d = q(d)$. In other words, it allows $\mathcal{S}$ to break the modified ACV-BGKM scheme to recover the intermediate key k_d from the public information tuple PI_d without the knowledge of the secret s_d. In Appendix I, we show that the probability of breaking the modified ACV-BGKM scheme is a negligible $1/q^N$ where q is the ℓ bit prime number and N is the maximum number of users.

Now, we discuss the efficiency of the threshold AB-GKM with respect to computational costs and required bandwidth for rekeying.

For any Usr_i in the group deriving the shared group key requires: $\sum_{i=1}^{d} N_i$ hashing operations (evaluations of $H(\cdot)$), where N_i is the maximum number of users having attr_i; and d inner product computations $v_i \cdot ACV_i$ of two $(2N_i)$-dimensional $\mathbb{F}_q$-vectors and the Lagrange interpolation $O(m \log^2 m)$, where $m = |\mathcal{A}|$. Therefore, the overall computational complexity is $O(N + m \log^2 m)$ where $N = \sum_{i=1}^{d} N_i$. Notice that the inner product computations are independent and can be parallelized to improve performance.

For every rekeying phase, for each attr_i, Svr needs to form a matrix A_i by performing N_i^2 hashing operations, and then solve a linear system of size $N_i \times (2N_i)$. Solving the linear system is the most costly operation as N_i gets large for computation on Svr; it requires $O(\sum_{i=1}^{m} N_i^3)$ field operations in $\mathbb{F}_q$.

When a rekeying process takes place, the new information to be broadcast is $PI_i = \ ACV_i, (z_1, \ldots, z_{N_i})$, $i = 1, 2, \cdots, m$, where ACV_i is a vector consisting of $(2N_i)$ elements in $\mathbb{F}_q$, and without loss of generality we can pick z_i to be strings with a fixed length. This gives an overall communication complexity $O(\sum_{i=1}^{m} N_i)$.

For a group of maximum N users, in the worst case, the threshold AB-GKM only requires each Usr to store ($O(m)$) secrets, one secret per attribute that Usr possesses and Svr to keep track of all $O(Nm)$ secrets.

VI. Scheme 3: Access Tree AB-GKM

In the inline AB-GKM scheme, the policy P is embedded into the BGKM scheme itself. As discussed in Section V, while this approach works for many different types of policies, such an approach is not able to efficiently support threshold access control policies. Scheme 2, threshold AB-GKM, on the other hand, is able to efficiently support threshold policies, but it is unable to support other policies. In order to support more expressive policies, we extend the threshold AB-GKM. Like the threshold AB-GKM, instead of embedding P in the BGKM scheme, we construct a separate BGKM instance for each attribute. Then, we embed the P in an access structure $\mathcal{T}$. $\mathcal{T}$ is a tree with the internal nodes representing threshold gates and the leaves representing attributes. The construction of $\mathcal{T}$ is similar to that of the approach by Goyal et al. [3]. However, unlike Goyal et al.'s approach, the goal of our construction is to derive the group key for the users whose attributes satisfy the access structure $\mathcal{T}$.

A. Access Tree

Let $\mathcal{T}$ be a tree representing an access structure. Each internal node of the tree represents a threshold gate. A threshold gate is described by its child nodes and a threshold value. If n_x is the number of children of a node x and t_x is its threshold value, then $0 < t_x \leq n_x$. Notice that when $t_x = 1$, the threshold gate is an OR gate and when $t_x = n_x$, it is an AND gate. Each leaf node x of the tree is described by an attribute, a corresponding BGKM instance and a threshold value $t_x = 1$. The children of each node x are indexed from 1 to n_x.

We define the functions in Table II in order to construct our scheme. All the functions except sat are straightforward to implement. A brief description of sat follows:

The function $\mathsf{sat}(\mathcal{T}_x, \alpha)$ works as a recursive function. If x is a leaf node, it returns 1, provided that the attribute associated with x is in the set of attributes α and 0 otherwise. If x is an internal node, if at least t_x children nodes of x return 1, then $\mathsf{sat}(\mathcal{T}_x, \alpha)$ returns 1 and 0 otherwise.

B. Our Construction

The access tree AB-GKM scheme consists of five algorithms:

Setup(ℓ): Svr initializes the parameters of the underlying modified ACV-BGKM scheme: the prime number q, the maximum group size N ($\leq n$), the cryptographic hash function H, the key

TABLE II

ACCESS TREE FUNCTIONS

Function	**Description**
index(x)	Returns the index of node x
parent(x)	Returns the parent node of node x
attr(x)	Returns the index of the attribute associated with a leaf node x
q_x	The polynomial assigned to node x
sat($\mathcal{T}_x, \alpha$)	Returns 1 if the set of attributes α satisfies $\mathcal{T}_x$, the subtree rooted at node x, and 0 otherwise

space $\mathcal{KS}$, the secret space $\mathcal{SS}$, the set of issued secrets **S**, the user-attribute matrix UA and the universe of attributes $\mathcal{A} = \{\mathsf{attr}_1, \mathsf{attr}_2, \cdots, \mathsf{attr}_m\}$.

Svr defines the Lagrange coefficient $\Delta_{i,\mathbf{Q}}$ for $i \in \mathbb{F}_q$ and a set, **Q** of elements in $\mathbb{F}_q$:

$$\Delta_{i,\mathbf{Q}}(x) = \prod_{j \in \mathbf{Q}, j \neq i} \frac{x-j}{i-j}.$$

SecGen(γ_i): For each attribute $\mathsf{attr}_j \in \gamma_i$, where $\gamma_i \subset \mathcal{A}$, Svr invokes **SecGen()** of the modified ACV-BGKM scheme to obtain the random secret $s_{i,j}$. It returns β_i, the set of secrets for all the attributes in γ_i.

KeyGen(P): Svr transforms the policy P into an access tree $\mathcal{T}$. The algorithm outputs the public information which a user can use to derive the group key if and only if the user's attributes satisfy the access tree $\mathcal{T}$ built for the policy P. The algorithm constructs the public information as follows.

For each user Usr_i having the intermediate set of keys $\mathbf{K}_i = \{k_{i,j} | 1 \leq j \leq m\}$, where $k_{i,j}$ represents the intermediate key for Usr_i and attr_j, the following construction is performed. For each attribute attr_i, there is a leaf node in $\mathcal{T}$. The construction of the tree is performed top-down. Each node x in the tree is assigned a polynomial q_x. The degree of the polynomial q_x, d_x is set to $t_x - 1$, that is, one less than the threshold value of the node. For the root node r, $q_r(0)$ is set to the group key k and d_r other points are chosen uniformly at random so that q_r is a unique polynomial of degree d_r fully defined through Lagrange interpolation. For any other node x, $q_x(0)$ is set to $q_{parent(x)}(index(x))$ and d_x other points are chosen uniformly at random to uniquely define q_x. For each leaf node x corresponding to a unique attribute attr_j, $q_x(0)$ is set

to $q_{parent(x)}(1)$ and $k_{i,j} = q_x(0)$.

At the end of the above computation, we have all the sets of intermediate keys $\mathbf{K} = \{\mathbf{K}_i | \mathsf{Usr}_i, 1 \leq i \leq N\}$. For each leaf node x, the modified BGKM algorithm **KeyGen($\mathbf{S}_x$, $\mathbf{K}_x$)**, where $\mathbf{S}_x$ is the set of secrets corresponding to the attribute associated with the node x and $\mathbf{K}_x = \{k_{i,j} | 1 \leq i \leq N, \mathsf{attr}_j\}$, $j = \mathsf{attr}(x)$, is invoked to generate public information tuple PI_x. We denote the set of all the public information tuples $\mathbf{PI} = \{PI_j | \mathsf{attr}_j, 1 \leq j \leq m\}$.

KeyDer(β_i, PI): Given β_i, a set of secret values corresponding to the attributes of Usr_i, and the set of public information tuples **PI**, it outputs the group key k.

The key derivation is a recursive procedure that takes β_i and **PI** to bottom-up derive k. Note that a user can obtain the key if and only if her attributes satisfy the access tree $\mathcal{T}$, i.e., $\mathsf{sat}(\mathcal{T}_r, \beta_i)$ = 1. The high-level description of the key derivation is as follows.

For each leaf node x corresponding to the attribute with the user's secret value $s_x \in \beta_i$, the user derives the intermediate key k_x using the underlying modified BGKM scheme **KeyDer**(s_x, PI_x). Using Lagrange interpolation, the user recursively derives the intermediate key k_x for each internal ancestor node x until the root node r is reached and $k_r = k$. Notice that since intermediate keys are tied to unique polynomials, users cannot collude to derive the group key k if they are unable to derive it individually. A detailed description follows.

If x is a leaf node, it returns an empty value $\perp$ if $\mathsf{attr}(x) \notin \beta_i$, otherwise it returns the key k_x as follows:

$$k_x = v_x \cdot ACV_x,$$

where v_x is the key derivation vector corresponding to the attribute $\mathsf{attr}_{\mathsf{attr}(x)}$ and ACV_x the access control vector in PI_x.

If x is an internal node, it returns an empty value $\perp$ if the number of children nodes having a non-empty key is less than t_x, otherwise it returns k_x as follows:

Let the set $\mathbf{Q}_x$ contain the indices of t_x children nodes having non-empty keys $\{k_i | i \in \mathbf{Q}_x\}$.

$$\Delta_{i,\mathbf{Q}_x}(y) = \prod_{i \in \mathbf{Q}_x, i \neq j} \frac{y-i}{j-i}$$

$$q_x(y) = \sum_{i \in \mathbf{Q}_x} k_i \Delta_{i,\mathbf{Q}_x}(y)$$

$$k_x = q_x(0).$$

The above computation is performed recursively until the root node is reached. If Usr_i satisfies $\mathcal{T}$, Usr_i gets $k = q_r(0)$, where r is the root node. Otherwise, Usr_i gets an empty value $\perp$.

C. Security and Performance

If an adversary can break our access tree AB-GKM scheme, a simulator can be constructed to break the modified ACV-BGKM scheme. We give a high-level detail of the reduction based proof.

Suppose that an adversary $\mathcal{A}$ having a set of attributes α that does not satisfy the access tree $\mathcal{T}$ breaks our scheme in the random oracle model with advantage ϵ. Let the root node of $\mathcal{T}$ be r and the group key $k = q_r(0)$. Notice that since $\mathcal{A}$ does not satisfy $\mathcal{T}$ and $q_r(x)$ a t_r-*out-of*-n_r threshold scheme, $\mathcal{A}$ satisfies no more than $t_r - 1$ subtrees rooted at children of r out of the n_r subtrees. By inference, it is easy to see that $\mathcal{A}$ does not satisfy at least one leaf node. We build a simulator $\mathcal{S}$ that can derive k_x from PI_x corresponding to one such unsatisfied leaf node with advantage ϵ. In other words, we build a simulator to break the modified ACV-BGKM scheme.

Like the proof in Section V, using $\mathcal{A}$ as a routine, $\mathcal{S}$ can obtain the group key k with advantage ϵ. Now, $\mathcal{S}$ works downwards $\mathcal{T}$ to recover the keys for nodes originally unsatisfied by $\mathcal{A}$ using Lagrange interpolation. For example, using k and $t_r - 1$, $\mathcal{S}$ obtains the key k_{t_r} for the t_r^{th} child node of r. Finally, $\mathcal{S}$ obtains the key k_x for an unsatisfied leaf node x corresponding to $attr_x$. In other words, it allows $\mathcal{S}$ to break the modified ACV-BGKM scheme to recover the key k_x from the public information tuple PI_x without the knowledge of the secret s_x. We show that the probability of breaking the modified ACV-BGKM scheme following **KeyDer** algorithm is a negligible $1/q^N$ where q is the ℓ bit prime number and N is the maximum number of users.

Now, we discuss the efficiency of threshold AB-GKM with respect to computational costs and required bandwidth for rekeying.

For any Usr_i in the group, deriving the shared group key requires: $\sum_{i=1}^{d} N_i$ hashing operations (evaluations of $H(\cdot)$), where $d = |\beta_i|$, N_i is the maximum number of users having attr_i, and d inner product computations $v_i \cdot ACV_i$ of two $(2N_i)$-dimensional $\mathbb{F}_q$-vectors and M Lagrange interpolations $O(Mm\log^2 m)$, where M = No. of internal nodes in $\mathcal{T}$ and $m = |\mathcal{A}|$. Therefore, the overall computational complexity is $O(N + Mm\log^2 m)$ where $N = \sum_{i=1}^{d} N_i$. Notice that the inner product computations are independent and can be parallelized to improve performance.

The cost of rekeying, communication and storage are comparable to those of the threshold scheme presented in Section V.

VII. Application to a Medical System

Among other applications, fine-grained access control in a group setting using broadcast encryption is one important application of AB-GKM schemes. We illustrate our AB-GKM schemes using a healthcare scenario [18], [6]. A hospital (Svr) supports fine-grained access control on electronic health records (EHRs) [34], [35] by encrypting and making the encrypted records available to hospital employees (Usrs). Typical hospital users include employees playing different roles such as receptionist, cashier, doctor, nurse, pharmacist, system administrator and non-employees such as patients. An EHR document is divided into subdocuments including BillingInfo, ContactInfo, Medication, PhysicalExam, LabReports and so on. In accordance with regulations such as health insurance portability and accountability act (HIPAA), the hospital policies specify which users can access which subdocument(s). A cashier, for example, need not have access to data in EHRs except for the BillingInfo, while a doctor or a nurse need not have access to BillingInfo. These policies can be based on the content of EHRs itself. An example of such policies is that "information about a patient with cancer can only be accessed by the primary doctor of the patient". In addition, patients define their own privacy policies to protect their EHRs. For example, a patient's policy may specify that "only the doctors and nurses who support her insurance plan can view her EHR".

In order to support content-based access control, the hospital maintains some associations among users and data. Tables III and IV show some example associations. Table III shows for each patient, identified by the pseudonym "Patient ID", the corresponding insurance plan. Table IV shows the insurance plans supported by each doctor and nurse, identified by the pseudonym "Employee ID".

TABLE III

PATIENT INSURANCE PLANS

Patient ID	**Insurance Plan**
pat_1	Med A
pat_2	Med B
pat_3	ACME
pat_4	Med A

TABLE IV

INSURANCE PLANS SUPPORTED BY DOCTORS/NURSES

EmployeeID	**Attributes**	**Insurance Plan(s)**
emp_1	doctor	MedB, ACME
emp_2	doctor	ACME
emp_3	nurse/junior	ACME
emp_4	nurse/senior	MedA
emp_5	nurse/senior	MedC
emp_6	doctor	MedA
emp_7	doctor	MedB, ACME
emp_8	nurse/senior	MedA
emp_9	nurse/senior	MedA, MedB, ACME

The hospital runs **Setup** algorithm to initialize system parameters and issues secrets to employees by running the **SecGen** algorithm. Table V shows the content of the user attribute matrix UA that the hospital maintains.

In what follows we show how an example access control policy is enforced for each of the three AB-GKM schemes we constructed. Note that the example policies considered in the following sub-sections are not related and used only for the illustrative purposes.

A. Inline AB-GKM

First we illustrate the use of inline AB-GKM scheme. Consider the following policy specification on the Medication subdocument of the EHR: “A senior nurse or a doctor can access Medication” and “Doctors and nurses supporting the insurance plan of a patient can access

TABLE V

USER ATTRIBUTE MATRIX

Emp ID	**doctor**	**nurse**	**senior**	**junior**	**MedA**	**MedB**	**MedC**	**ACME**
emp_1	100	$\perp$	$\perp$	$\perp$	$\perp$	111	$\perp$	102
emp_2	120	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	105
emp_3	$\perp$	106	$\perp$	120	$\perp$	$\perp$	$\perp$	121
emp_4	$\perp$	103	150	$\perp$	175	$\perp$	$\perp$	$\perp$
emp_5	$\perp$	133	151	$\perp$	$\perp$	$\perp$	161	$\perp$
emp_6	129	$\perp$	$\perp$	$\perp$	141	$\perp$	$\perp$	$\perp$
emp_7	119	$\perp$	$\perp$	$\perp$	$\perp$	133	$\perp$	137
emp_8	$\perp$	143	152	$\perp$	115	$\perp$	$\perp$	$\perp$
emp_9	$\perp$	109	156	$\perp$	117	119	$\perp$	124

the patient's EHR". The insurance plan is an attribute of the Medication subdocument. The corresponding access control policy looks like as follows:

P = ((“role = nurse” $\wedge$ “level = senior”) $\vee$ (“role = doctor”)) $\wedge$ (“patient-insurance = employee-insurance”)

The policy is expressed in CNF as follows:

P = ((“role = nurse” $\wedge$ “level = senior” $\wedge$ “patient-insurance = employee-insurance”) $\vee$ (“role = doctor” $\wedge$ “patient-insurance = employee-insurance”)

Let the medication subdocument of pat_i, $1 \leq i \leq 4$ be M_i. Since the above access control policy involves content based access control and the patients are covered under three different insurance plans, we need to perform three encryptions with three different keys. Medication subdocuments M_1 and M_4 are encrypted with the same symmetric group key k_1, M_2 with k_2 and M_3 with k_3. We now show how the hospital executes the **KeyGen** algorithm for k_1. The hospital computes the aggregated secrets for the employees who satisfy the above access control policy with the MedA insurance plan (Table VI).

The hospital then invokes the **KeyGen** algorithm with the group key k_1 and the aggregated secrets in Table VI to generate the corresponding public information tuple PI_1:

TABLE VI

AGGREGATED SECRETS FOR k_1

Emp ID	**Attributes**	**Aggregated Secret**
emp_4	nurse, senior, MedA	$103\|\|150\|\|175$
emp_6	doctor, MedA	$129\|\|141$
emp_8	nurse, senior, MedA	$143\|\|152\|\|115$
emp_9	nurse, senior, MedA	$109\|\|156\|\|117$

$$PI_1 = \ ACV_1, (z_1, z_2, z_3, z_4) \ .$$

The hospital publishes $\mathcal{E}_{k_1}(\mathsf{M}_1)$, $\mathcal{E}_{k_1}(\mathsf{M}_4)$, where $\mathcal{E}$ is a symmetric encryption algorithm, along with PI_1. Similarly, two additional **KeyGen** operations are performed for the Medication subdocuments $\mathbf{M}_2$ and $\mathbf{M}_3$.

Fine grained access control. Notice that only those employees who satisfy the access control policy can derive one or more symmetric group keys k_1, k_2, k_3 using **KeyDer** algorithm and decrypt the Medication subdocuments. Specifically, only emp_4, emp_6, emp_8 and emp_9 can access M_1 and M_4, only emp_1, emp_7 and emp_9 can access M_2, and only emp_1, emp_2, emp_7 and emp_9 can access M_3. Observe that even though emp_5 is a senior nurse, she cannot decrypt any of the Medication subdocuments as the insurance plan that she supports is different. Even though emp_3 is a nurse with an insurance plan overlapping with that of a patient, she cannot decrypt any of the Medication subdocuments since she is a junior nurse.

Collusion resistance. It may appear that if emp_3 and emp_5 collude, they can obtain M_3. However, the inline AB-GKM construction makes it impossible to combine secrets from multiple users to create a valid aggregated secret. Thus, neither emp_3 nor emp_5 can access M_3.

Handling user dynamics. Assume a new doctor who supports the insurance plan MedA joins the hospital. In order to provide backward security, the hospital needs to update only PI_1, the public information tuple corresponding to the Medication subdocuments M_1 and M_4, and re-encrypt M_1 and M_4 with the new group key k_1'. A similar approach is taken to assure forward security when an employee resigns from the hospital or some attributes are revoked. Notice that our scheme can

even handle policy changes with the same approach used for handling changes in user attributes.

B. Threshold AB-GKM

Now we illustrate the use of the threshold AB-GKM scheme. Consider the following policy specification on the Medication subdocument of the EHR. "An employee supporting at least two insurance plans can access the Medication of any patient". We consider each insurance plan as an attribute. Since there are four insurance plans, MedA, MedB, MedC and ACME, the above policy can be implemented by a 2-*out-of*-4 threshold AB-GKM scheme. Table VII shows the list of employees who satisfy each insurance plan attribute.

TABLE VII

LIST OF EMPLOYEES SATISFYING EACH INSURANCE PLAN

Attribute	**Employee IDs**
MedA	$\mathsf{emp}_4, \mathsf{emp}_6, \mathsf{emp}_8, \mathsf{emp}_9$
MedB	$\mathsf{emp}_1, \mathsf{emp}_7, \mathsf{emp}_9$
MedC	emp_5
ACME	$\mathsf{emp}_1, \mathsf{emp}_2, \mathsf{emp}_3, \mathsf{emp}_7, \mathsf{emp}_9$

The hospital executes the **KeyGen** algorithm to generate 4 PI tuples and encrypts the Medication subdocuments with the group symmetric key k:

$$
\begin{aligned}
PI_{MedA} &= ACV_{MedA}, (z_1, z_2, z_3, z_4) \\
PI_{MedB} &= ACV_{MedB}, (z_5, z_6, z_7) \\
PI_{MedC} &= ACV_{MedC}, (z_8) \\
PI_{ACME} &= ACV_{ACME}, (z_9, z_{10}, z_{11}, z_{12}, z_{13})
\end{aligned}
$$

Threshold access control. Notice that only three employees can derive the group key k using **KeyDer** algorithm to decrypt the Medication subdocuments: emp_1, emp_7, and emp_9. Our threshold scheme is very flexible in that an employee supporting more than two insurance plans can use any two insurance plans to derive the key k. For example, emp_9 can use any two of her three insurance plans.

Collusion resistance. Notice that emp_4 supports MedA and emp_5 supports MedC. It may appear that these two employees can collude in order to derive the group key k. Since the threshold AB-GKM scheme associates each user with a unique degree 1 polynomial, combining the intermediate keys derived from PI_{MedA} and PI_{MedB} for emp_4 and emp_5, respectively, does not result in a correct polynomial whose constant is the group key k.

Handling user dynamics. Assume that emp_1 no longer supports the insurance plan ACME. The hospital re-generates the public information by removing emp_1 from the calculation of PI_{ACME} and associating a new group key k'. emp_1 is not able to derive k' since it is associated only with PI_{MedB}. Notice that the revocation does not affect the secret information each existing employee has. A similar approach is taken when an existing employee supports a new insurance plan. It should be noted that our scheme has the added flexibility to support a different threshold policy by requiring only to change the public information.

C. Access Tree AB-GKM

Now we illustrate the use of the access tree AB-GKM scheme. Consider the following policy specification on the Medication subdocument of the EHR. "A senior nurse supporting at least two insurance plans can access Medication of any patient". In order to implement this access control policy, we need to consider attributes role, level and insurance plan. The access control policy looks as follows:

P = ("role = nurse" $\wedge$ "level = senior" $\wedge$ "2-*out-of*-{MedA, MedB, MedC, ACME}")

In addition to Table VII containing the list of employees satisfying insurance plans, the hospital maintains the list of employees satisfying the attributes nurse and senior as shown in Table VIII.

TABLE VIII

LIST OF EMPLOYEES SATISFYING ATTRIBUTES

Attribute	**Employee IDs**
nurse	$\mathsf{emp}_3, \mathsf{emp}_4, \mathsf{emp}_5, \mathsf{emp}_8, \mathsf{emp}_9$
senior	$\mathsf{emp}_4, \mathsf{emp}_5, \mathsf{emp}_8, \mathsf{emp}_9$

The above policy can be represented using an access tree with two internal nodes and six leaf nodes. The root node is an AND gate and has three children. The first and second children of

the root node represent the attributes nurse and senior, respectively, and the third child of the root node is a 2-*out-of*-4 threshold gate which has four children representing the four insurance plans.

The hospital executes the **KeyGen** algorithm to generate six PI tuples and encrypts the Medication subdocuments with the group symmetric key k:

$$\begin{aligned}
PI_{MedA} &= ACV_{MedA}, (z_1, z_2, z_3, z_4) \\
PI_{MedB} &= ACV_{MedB}, (z_5, z_6, z_7) \\
PI_{MedC} &= ACV_{MedC}, (z_8) \\
PI_{ACME} &= ACV_{ACME}, (z_9, z_{10}, z_{11}, z_{12}, z_{13}) \\
PI_{nurse} &= ACV_{nurse}, (z_{14}, z_{15}, z_{16}, z_{17}, z_{18}) \\
PI_{senior} &= ACV_{senior}, (z_{19}, z_{20}, z_{21}, z_{22})
\end{aligned}$$

Expressive access control. Notice that only one employee, emp_9, can derive the group key k using **KeyDer** algorithm to decrypt Medication subdocuments.

Collusion resistance. Notice that emp_4 supports MedA and emp_5 supports MedC and both of them are senior nurses. It may appear that these two employees can collude to derive the group key k. Since, in this particular example, the access tree AB-GKM scheme associates each user with two unique polynomials, one for the AND gate and another for the threshold gate, none of them individually satisfies the access tree and **KeyDer** results in an incorrect key.

Handling user dynamics. Assume that emp_4 starts to support the insurance plan ACME in addition to MedA. The hospital re-generates the public information by adding emp_4 to the calculation of PI_{ACME} and associating a new group key k'. Now emp_4 is able to derive k' using **KeyDer** as its attributes satisfy the access tree. Notice that the change in the user attributes does not affect the secret information each existing employees have. A similar approach is taken when one or more of these attributes are revoked from an existing employee. It should be noted that, like the first two schemes, this scheme has the added flexibility to support changes to the access tree by requiring only changes to the public information.

VIII. Conclusion

In this paper, we have presented three attribute based group key management (AB-GKM) schemes: inline AB-GKM, threshold AB-GKM, and access tree AB-GKM. In all our schemes, when the group changes, the rekeying operations do not affect the private information of existing group members and thus our schemes eliminate the need of establishing expensive private communication channels. We have also shown that our schemes are resistant to collusion attacks; multiple users are not able to combine their private information to derive a group key which they cannot derive individually.

Our constructions are based on a provably secure ACV-BGKM scheme and Shamir's threshold scheme. We have introduced a modified ACV-BGKM scheme with security proofs in order to construct threshold and access tree AB-GKM schemes. We have provided high-level proofs of security of the three schemes under the random oracle model. We have also described a practical group scenario where our schemes are utilized to manage the group.

As future work, we plan to implement the proposed schemes and experimentally evaluate their performance.

References

[1] R. Krishnan, R. Sandhu, J. Niu, and W. H. Winsborough, "Foundations for group-centric secure information sharing models," in *Proceedings of the 14th ACM symposium on Access control models and technologies*, ser. SACMAT '09. New York, NY, USA: ACM, 2009, pp. 115–124. [Online]. Available: http://doi.acm.org/10.1145/1542207.1542227

[2] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Eurocrypt 2005, LNCS 3494*. Springer-Verlag, 2005, pp. 457–473.

[3] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2006, pp. 89–98.

[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 321–334.

[5] X. Zou, Y. Dai, and E. Bertino, "A practical and flexible key management mechanism for trusted collaborative computing," *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pp. 538–546, April 2008.

[6] N. Shang, M. Nabeel, F. Paci, and E. Bertino, "A privacy-preserving approach to policy-based content dissemination," in *ICDE '10: Proceedings of the 2010 IEEE 26th International Conference on Data Engineering*, 2010.

[7] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[8] Y. Challal and H. Seba, "Group key management protocols: A novel taxonomy," *International Journal of Information Technology*, vol. 2, no. 2, pp. 105–118, 2006.

[9] H. Harney and C. Muckenhirn, “Group key management protocol (gkmp) specification,” Network Working Group, United States, Tech. Rep., 1997.

[10] H. Chu, L. Qiao, K. Nahrstedt, H. Wang, and R. Jain, “A secure multicast protocol with copyright protection,” *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 2, pp. 42–60, 2002.

[11] C. Wong and S. Lam, “Keystone: a group key management service,” in *International Conference on Telecommunications, ICT*, 2000.

[12] A. Sherman and D. McGrew, “Key establishment in large dynamic groups using one-way function trees,” *Software Engineering, IEEE Transactions on*, vol. 29, no. 5, pp. 444–458, May 2003.

[13] S. Berkovits, “How to broadcast a secret,” in *EUROCRYPT ’91: Proceedings of the 10th annual international conference on Advances in Cryptology*. Berlin, Heidelberg: Springer-Verlag, 1991, pp. 535–541.

[14] D. Naor, M. Naor, and J. B. Lotspiech, “Revocation and tracing schemes for stateless receivers,” in *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO ’01. London, UK: Springer-Verlag, 2001, pp. 41–62. [Online]. Available: http://portal.acm.org/citation.cfm?id=646766.704277

[15] D. Halevy and A. Shamir, “The lsd broadcast encryption scheme,” in *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO ’02. London, UK: Springer-Verlag, 2002, pp. 47–60. [Online]. Available: http://portal.acm.org/citation.cfm?id=646767.704291

[16] D. Boneh and M. Franklin, “Identity-based encryption from the weil pairing,” in *CRYPTO ’01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*. Springer-Verlag, 2001, pp. 213–229.

[17] C. Cocks, “An identity based encryption scheme based on quadratic residues,” in *Proceedings of the 8th IMA International Conference on Cryptography and Coding*. London, UK: Springer-Verlag, 2001, pp. 360–363.

[18] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, “Secure attribute-based systems,” in *CCS ’06: Proceedings of the 13th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2006, pp. 99–112.

[19] P. Traynor, K. R. B. Butler, W. Enck, and P. McDaniel, “Realizing massive-scale conditional access systems through attribute-based cryptosystems,” in *Proceedings of the Network and Distributed System Security Symposium, 2008*, ser. NDSS 2008, 2008.

[20] L. Cheung, J. A. Cooley, R. Khazan, and C. Newport, “Collusion-resistant group key management using attribute-aased encryption. cryptology eprint archive report 2007/161,” 2007.

[21] S. Yu, K. Ren, and W. Lou, “Attribute-based on-demand multicast group setup with membership anonymity,” in *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, ser. SecureComm ’08. New York, NY, USA: ACM, 2008, pp. 18:1–18:6. [Online]. Available: http://doi.acm.org/10.1145/1460877.1460900

[22] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha, “Key management for secure internet multicast using boolean function minimization techniques,” in *INFOCOM 1999. The 18th Conference on Computer Communications. IEEE*, 1999, pp. 689–698.

[23] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, “The versakey framework: versatile group key management,” *Selected Areas in Communications, IEEE Journal on*, vol. 17, no. 9, pp. 1614 –1631, Sep. 1999.

[24] A. Fiat and M. Naor, “Broadcast encryption,” in *Advances in Cryptology - CRYPTO 93*, ser. Lecture Notes in Computer Science, D. Stinson, Ed., vol. 773. Springer Berlin / Heidelberg, 1994, pp. 480–491. [Online]. Available: http://dx.doi.org/10.1007/3-540-48329-2_40

[25] E. Bertino and E. Ferrari, “Secure and selective dissemination of XML documents,” *ACM Trans. Inf. Syst. Secur.*, vol. 5, no. 3, pp. 290–331, 2002.

[26] G. Miklau and D. Suciu, “Controlling access to published data using cryptography,” in *VLDB ’2003: Proceedings of the 29th international conference on Very large data bases*. VLDB Endowment, 2003, pp. 898–909.

[27] D. Halevy and A. Shamir, “The lsd broadcast encryption scheme,” in *Advances in Cryptology CRYPTO 2002*, ser. Lecture Notes in Computer Science, M. Yung, Ed., vol. 2442. Springer Berlin / Heidelberg, 2002, pp. 145–161. [Online]. Available: http://dx.doi.org/10.1007/3-540-45708-9_4

[28] D. Boneh, C. Gentry, and B. Waters, “Collusion resistant broadcast encryption with short ciphertexts and private keys,” in *Advances in Cryptology CRYPTO 2005*, ser. Lecture Notes in Computer Science, V. Shoup, Ed., vol. 3621. Springer Berlin / Heidelberg, 2005, pp. 258–275. [Online]. Available: http://dx.doi.org/10.1007/11535218_16

[29] J. Benaloh and J. Leichter, “Generalized secret sharing and monotone functions,” in *CRYPTO ’88: Proceedings of the 8th Annual International Cryptology Conference on Advances in Cryptology*. London, UK: Springer-Verlag, 1990, pp. 27–35.

[30] E. F. Brickell, “Some ideal secret sharing schemes,” in *EUROCRYPT ’89: Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*. New York, NY, USA: Springer-Verlag New York, Inc., 1990, pp. 468–475.

[31] T. Pedersen, “Non-interactive and information-theoretic secure verifiable secret sharing,” in *Advances in Cryptology CRYPTO 91*, ser. Lecture Notes in Computer Science, J. Feigenbaum, Ed., vol. 576. Springer Berlin / Heidelberg, 1992, pp. 129–140. [Online]. Available: http://dx.doi.org/10.1007/3-540-46766-1_9

[32] N. Shang, M. Nabeel, E. Bertino, and X. Zou, “Broadcast group key management with access control vectors,” Department of Computer Science, Tech. Rep., 4 2010.

[33] D. Dummit and R. Foote, “Gaussian-Jordan elimination,” in *Abstract Algebra*, 2nd ed. Wiley, 1999, p. 404.

[34] “XML in clinical research and healthcare industries,” http://xml.coverpages.org/healthcare.html.

[35] M. Eichelberg, T. Aden, J. Riesmeier, A. Dogac, and G. B. Laleci, “A survey and analysis of electronic healthcare record standards,” *ACM Comput. Surv.*, vol. 37, no. 4, pp. 277–315, 2005.