TRANSACTIONS ON DATA PRIVACY 7 (2014) 309–336

Attribute Based Group Key Management

Mohamed Nabeel, Elisa Bertino

305 N. University Street, West Lafayette, IN, 47907.

Email: {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; they 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. Experimental results show that our underlying constructs are efficient and practical.

Keywords. Broadcast group key management, attribute based policies, secret sharing

1 Introduction

New application domains have pushed novel paradigms and tools for supporting collaboration among (possibly very dynamic) user groups (see for example the notion of group-centric information sharing [20]). 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) [29, 16, 5]; 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 (forward secrecy) and a user who left the group cannot access the data transmitted after she left (backward 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 [39, 31] 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 do not 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 [29, 16, 5]. 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 expressive collusion-resistant AB-GKM scheme. However, such an approach suffers from some drawbacks. Except for a few recent approaches [7, 28] and their variants, many popular ABE schemes do not support efficient revocation of group members. Among the notable work, Boldyreva et al.'s revocation scheme [7] supports only a limited set of ABE functionality, and Sahai et al.'s revocation scheme [28], which is based on the notion of re-encrypting from one policy to another more restrictive policy by utilizing the delegation capabilities of the underlying ABE construct, requires maintaining additional attributes and relatively expensive operations even though the complexity is reduced to the polylogarithm number of group members. 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 readily work. 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 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

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

Table 1: Acronyms

Acronym	Description
GKM	Group Key Management
BGKM	Broadcast GKM
ABE	Attribute Based Encryption
CP-ABE	Ciphertext Policy ABE
KP-ABE	Key Policy ABE
ACV	Access Control Vector
KEV	Key Extraction Vector
ABAC	Attribute Based Access Control
AB-GKM	Attribute Based GKM
PI	Public Information tuple
UA	User-Attribute matrix

they cannot derive individually.

Our AB-GKM constructions are based on an optimized version of the ACV-BGKM (Access Control Vector BGKM) scheme [31], a provably secure BGKM scheme, and Shamir's threshold scheme [30]. 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 requirement. 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 are constructed by using a modified version of ACV-BGKM, also proposed in this paper.

The reminder of the paper is organized as follows: Section 2 describes related work. Section 3 summarizes the ACV-BGKM scheme [31]. Sections 4, 5, 6 present the construction of the inline AB-GKM, threshold AB-GKM, and access tree AB-GKM schemes, respectively, and analyze their security and performance. Section 7 shows an example application using the access tree AB-GKM scheme. Section 8 provides experimental results of our underlying optimized ACV-BGKM scheme used in all three schemes against the CP-ABE (ciphertext policy attribute based encryption) scheme. Section 9 concludes the paper. Table 1 lists, for the convenience of the reader, the acronyms used in the paper.

2 Related Work

Group Key Management (GKM): Early approaches to GKM rely on a key server to share a secret with users to distribute decryption keys [18, 11]. 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 [36, 32] 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 GKM (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 [2, 39]. However, these schemes are not formally proven to be secure. Recently Shang et. al. introduced the first provably secure BGKM scheme called ACV-BGKM [31]. Existing BGKM schemes require sending $O(n)$ public information when rekeying. We improve the complexity by utilizing subset-cover techniques [25, 17]. Such improved BGKM schemes efficiently handle group dynamics and lay the foundation for AB-GKM. However such schemes cannot directly handle expressive conditions against attributes.

Group Key Exchange/Agreement: Unlike the GKM schemes presented above, in group key exchange (GKE) schemes, group members incrementally decide the group key by exchanging messages among the peer group members usually in a few rounds [9, 9, 19, 10]. Recently, more expressive and secure GKE schemes such as attribute based GKE [1, 15] and predicate based GKE [6], have been proposed. GKE schemes can efficiently support secure communication in dynamic peer groups. However, compared to our key management schemes, such approaches suffer from one or two of the following shortcomings: (1) high communication overhead in handling dynamic membership under one-to-many communication scenarios where a central authority similar to our approach specifies the access control policies and enforces them [35]; (2) high computation cost due to expensive pairing operations [1, 15, 6]; (3) inefficient member revocation which is inherent in the underlying ABE schemes [29, 5] utilized [1, 15, 6].

Attribute-Based Encryption (ABE) and GKM: In an ABE system [29], 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 was 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. Since the definition of the initial threshold scheme, a few variants have been introduced to provide more expressive ABE systems. Goyal et al. [16] introduced the idea of key-policy ABE (KP-ABE) systems and Bethencourt et al. [5] introduced the idea of CP-ABE systems. We have already mentioned in Section 1 the drawbacks of even the recent versions of ABE that support some form of revocation [7, 28]. Our work addresses such drawbacks. The proposers of some of these ABE 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. [34] propose 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.

GKM Schemes for Selective Dissemination Systems: Selective dissemination or broadcast encryption systems allow one to encrypt a message once and broadcast it 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 [14, 3, 22, 17, 8]. 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 [3] or some [22] 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.

3 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) [31, 24] in order for readers to better understand our constructions. It should be noted that we use ACV-BGKM in Section 4 and a modified version of ACV-BGKM in our constructions in Sections 5, and 6.

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 secrecy 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 picks a random bit string $s \notin \mathbf{S}$ uniformly at random from $\mathcal{SS}$, adds s to **S** and outputs s.

KeyGen(S): It picks a group key k uniformly at random from $\mathcal{KS}$ 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 $s \in \mathbf{S}$.

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 BGKM algorithms, we now provide an overview of the construction of the ACV-BGKM scheme under a client-server architecture. The ACV-BGKM scheme is only one approach to construct a secure BGKM scheme. One may construct a secure BGKM scheme using other constructs such as ACP-BGKM (Access Control Polynomial BGKM) [39]. The ACV-BGKM algorithms are executed by a trusted key server Svr and a group of users $\mathsf{Usr}_i, i = 1, 2, \ldots, n$. We refer the reader to the pioneering work on the ACV-BGKM scheme [31] for a simplified example and a practical application of the ACV-BGKM scheme.

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 **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

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

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., $s_i \in \mathsf{S}$.

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 numbers of users since the complexity of the matrix and the public information is $O(n)$. We proprose an improved scheme in Section 4.4 while keeping the underlying scheme unchanged.

4 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.

4.1 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(γ_i): For each attribute $\mathsf{attr}_j \in \gamma_i$, where $\gamma_i \subset \mathcal{A}$ and γ_i is the attribute set of Usr_i, 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 $\mathbf{S}$, 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 secrets exist 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

$$\mathsf{P} = \bigvee_{i=1}^{\alpha} \mathsf{conjunct}_i \text{ where there are } \alpha \text{ conjuncts and}$$

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

where each $\mathsf{conjunct}_i$ has ϕ_i conditions.

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

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

where N_{U_i} is the number of users satisfying $\mathsf{conjunct}_i$ [2]. Svr creates N_U $\widehat{s}_i$'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 $1 \leq i \leq N_U$

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

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

$$ACV = k \cdot e_1^T + Y, \text{ and}$$
$$PI = \langle ACV, (z_1, z_2, \ldots, z_N) \rangle$$

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 formula 2 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 $\widehat{s} \in \mathbf{AS}$.

Update(S): The composition of the user group changes when one of the following occurs:

- Identity attributes are added or removed resulting in the change in S and UA [3].
- The underlying policy P changes.

When such a change occurs, a new symmetric key k' is selected and **KeyGen(P)** 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'.

4.2 Security

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

Definition 1 (Security model for AB-GKM)**.**
Setup The challenger runs the Setup algorithm of AB-GKM and gives the public parameters to the adversary.
Phase 1 The adversary is allowed to request secrets for any set of attributes γ_i and the public information tuples for a policy satisfying these attributes. The public information along with the secrets allows the adversary to derive the private key.
Challenge The adversary declares the set of attributes γ that it wishes to challenged upon. γ is different from any of the attribute sets γ_i that the adversary queried earlier. The adversary submits two keys k_0 and k_1. The challenger flips a random coin b and chooses k_b. The

[2] It should be noted that N_U can be reduced to n, the number of users in the group, by exploiting the relationships between conjuncts and letting the users know the conjunct, out of the many they satisfy, they have to use to derive the key. We leave this optimization to keep the scheme simple.

[3] A change in a user attribute is viewed as two events; removing the existing attribute and adding a new attribute.

challenger generates public information for a policy P satisfying γ, but not any γ_i, using the **KeyGen** algorithm and give it to the adversary. The public information hides the group key k_b.
Phase 2 Phase 1 is repeated as many times provided that the adversary's attribute set does not satisfy P.
Guess The adversary outputs a guess b' of b.
The advantage of an adversary $\mathcal{A}$ in this game is defined as $Pr[b' = b] - 1/2$.

Now we define security under the security model for AB-GKM defined above.

Definition 2 (Security under the security model for AB-GKM)**.** An AB-GKM scheme is secure under the security model for AB-GKM if all adversaries have at most a negligible advantage in the above game defined in the security model.

We [31, 24] 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**. The intuition is that an adversary is required to construct a valid vector in the row space of the access control matrix in order to break ACV-BGKM and the probability of constructing such a vector is $1/q$. We capture the hardness of the ACV-BGKM scheme in the following assumption:

Definition 3 (ACV-BGKM Assumption)**.** No adversary without any valid secrets can break the ACV-BGKM scheme with more than a negligible probability assuming that the hash function is modeled as a random oracle. As shown in [24], the negligible probability is $1/q$.

Theorem 4. *If an adversary can break the inline AB-GKM scheme in the security model for AB-GKM, then a simulator can be constructed to break the ACV-BGKM scheme with non-negligible advantage* $1/2 + \epsilon$ *where* $\epsilon = 1/q$.

Proof. Suppose that there exists an adversary $\mathcal{A}$ that can break our scheme in the security model for AB-GKM with advantage ϵ. We build a simulator $\mathcal{B}$ that can break the ACV-BGKM scheme with the advantage at most ϵ. The simulation proceeds as follows:
The challenger runs the setup algorithm of ACV-BGKM and generates secrets for each attributes per user outside of $\mathcal{B}$'s view. The simulator $\mathcal{B}$ runs $\mathcal{A}$. $\mathcal{B}$ is given an instance of ACV-BGKM and gives the public parameters to $\mathcal{A}$. We assume that all policies are in DNF such that each conjunctive term has one or more attributes. The secret for a conjunctive term is derived by simply concatenating secrets for the constituent attributes. Notice that each aggregate secret is unique. The intuition behind the assumption is that inline AB-GKM is an extension of ACV-BGKM to support aggregate secrets and, therefore, with unique aggregate secrets, inline AB-GKM is equivalent to ACV-BGKM.
Phase 1 $\mathcal{A}$ submits sets of attributes γ_i to $\mathcal{B}$ and $\mathcal{B}$ sends the secrets corresponding to these attributes using the ACV-BGKM instance.
Challenge $\mathcal{A}$ submits the attribute set $\gamma \neq \gamma_i$ as the challenge and two keys k_0 and k_b. $\mathcal{B}$ flips a random coin b and chooses k_b and then using the ACV-BGKM instance, it generates the public information for a policy P that only γ satisfies and any of γ_i does not satisfy. The public information hides the group key k_b.
Phase 2 $\mathcal{A}$ and $\mathcal{B}$ repeats Phase 1 as many times provided $\mathcal{A}$'s attribute sets do not satisfy P.
Guess Using the public information and the information gathered from the two phases, $\mathcal{A}$ outputs a guess b' of b. Notice that the view of $\mathcal{A}$ when it is run as a subroutine of $\mathcal{B}$

and when it is run directly with the inline AB-GKM scheme is identical. In other words, $\mathcal{B}$ simulates an instance of the inline AB-GKM for $\mathcal{A}$ using an instance of the ACV-BGKM scheme. The simulation is trivial as the aggregate secrets in AB-GKM are unique as the secrets in ACV-BGKM. It should be noted that $\mathcal{A}$ does not have an advantage more than ϵ from the information gathered from the repeated execution of Phase 1 due to the key indistinguishability and key independence properties [24] of the ACV-BGKM scheme since the security of these two properties only depend on the uniqueness of the input secrets to the ACV-BGKM scheme. Notice that this forms the basis of collusion resistance as well. For different rounds of interactions between $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{B}$ produces different unique sets of secrets which results in different unique aggregate secrets. Therefore, the probability of guessing the aggregate secret for an attribute set satisfying P is as same as that of guessing the secret for a single attribute satisfying P in ACV-BGKM scheme.

It can easily be seen that $\mathcal{B}$ has the same advantage of breaking the ACV-BGKM scheme as $\mathcal{A}$ has on the inline AB-GKM scheme. As per the definitions, $\mathcal{B}$ breaks the ACV-BGKM with $Pr[b' = b] = 1/2 + \epsilon$. According to the assumption on the hardness of the ACV-BGKM scheme in Theorem 4, it follows that ϵ must be negligible. □

4.3 Performance

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 $O(n)$.

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 [12] is applied. Experimental results about the ACV-BGKM scheme [31] have shown that this can be performed in a short time when N is small.

When a rekeying process takes place, the new information to be broadcast is $PI = \langle ACV, (z_1, \ldots, z_N) \rangle$, 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 of fixed length. This gives an overall communication complexity $O(n)$. An advantage of inline AB-GKM is that no peer-to-peer private channel is needed for any persisting group members when rekeying is executed.

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.

4.4 Improving the Complexity using Subset-Cover

The above approach becomes inefficient if each N_{U_i} values are large as the computational and communication complexities are still proportional to N_{U_i} values. We utilize the result from previous research on broadcast encryption [25, 17] to improve the complexity. Based on that, one can make the complexity sub-linear in the number of users by giving more than one secret during **SecGen** for each attribute users 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 the 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. 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 backward 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** using the $\log n$ secrets corresponding to the roots of these subtrees. Naor et. al. [25] improve this technique to simultaneously revoke r users and describe the exiting users using $r \log (n/r)$ subtrees. Since then, there have been many improvements to the basic scheme. In the remainder of the paper, in order to maintain the simplicity the **SecGen** algorithm generates only one secret per attribute but the schemes can be trivially modified to use any subset-cover approach.

5 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 4, with such threshold policies the size of the access control matrix (A) increases exponentially if users are not informed which attributes to use. 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, multiple instances of ACV-BGKM allow 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.

5.1 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, the **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 its 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 the ACV-BGKM scheme, 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'.

5.1.1 Security of the Modified ACV-BGKM Scheme

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 5 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 6, which uses Lemma 5, is used to prove the soundness of the modified ACV-BGKM scheme.

Lemma 5. *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 6. *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 5,

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

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 7 (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 $j \neq r$ and $1 \leq r \leq N$, by substituting s_j and then running the key derivation algorithm **KeyDer** is negligible.

Theorem 8. *The modified ACV-BGKM scheme is sound.*

Proof. Let $PI = \langle ACV, (z_1, \ldots, z_N) \rangle$ 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 6, 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 $r \neq j$ 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.

□

5.2 Our Construction

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

Recall that in this scheme, 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 its 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) Svr initializes the parameters of the underlying modified ACV-BGKM scheme: the ℓ-bit prime number q, the maximum group size N ($\geq n$), 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 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 $\gamma_i \subset \mathcal{A}$ and γ_i is the attribute set of Usr_i, 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, d) Taking α, a subset of attributes from the attribute universe $\mathcal{A}$ and d, the threshold value, 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(α, d) The Update algorithm is invoked whenever α, the attribute set considered, or d, the threshold value, or the group members satisfying the threshold policy change. The group membership changes due to similar reasons mentioned under the **Update** algorithm in Section 4.1. In such a situation, a new symmetric group key k' is selected and **KeyGen(α, d)** is invoked to generate the set of new public information tuples **PI'**. Notice that the secrets shared with existing users are not affected by the group change.

5.3 Security

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

Theorem 9. *If an adversary can break the threshold AB-GKM scheme in the security model for AB-GKM, then a simulator can be constructed to break the modified ACV-BGKM scheme with non-negligible advantage* $1/2 + \epsilon$ *where* $\epsilon = 1/q^N$.

Proof. Suppose that an unbounded adversary $\mathcal{A}$ having a set of $d-1$ attributes α can break our scheme in the AB-GKM security model with advantage ϵ. Note that this is the most powerful adversary as it possesses $d-1$ attributes out of the d attributes required to derive the group key. We build a simulator $\mathcal{B}$ that can derive the key k_d from PI_d corresponding to $\mathsf{attr}_d \notin \alpha$ with the same advantage ϵ using $\mathcal{A}$ as subroutine. 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 only has a negligible advantage in obtaining the key for another attribute using the known attributes due to the key indistinguishability and independence properties of the ACV-BGKM scheme.

The challenger creates an instance of the modified ACV-BGKM scheme for each of the n attributes. $\mathcal{A}$ obtains secrets $\{s_i|i = 1, 2, \cdots, d-1\}$ for the attributes α it has from $\mathcal{B}$. The challenger constructs the public information tuples $\{PI_i|i = 1, 2, \cdots, d\}$, each having a random key k_i and gives them to $\mathcal{B}$. $\mathcal{B}$ in turn gives them to $\mathcal{A}$. Notice that the view of $\mathcal{A}$ is identical to that of $\mathcal{A}$ interacting directly with an instance of the threshold AB-GKM scheme, even though it is simulated. The random keys correspond to a random degree $d-1$ polynomial $q(x)$. Notice that $\mathcal{A}$ possesses secrets to obtain the random keys k_i, $1 \leq i \leq d-1$ and can derive the secret key k_d with an advantage ϵ from the public information tuples.

We omit the details of the security game defined in the previous section. As mentioned in the game, $\mathcal{A}$ may execute the threshold AB-GKM scheme for different sets of attributes

that do not satisfy the challenge threshold policy and do not include $attr_d$. As mentioned earlier, $\mathcal{A}$ does not gain any additional advantage by such executions.

After executing the phase 1 of the security game as many times, $\mathcal{A}$ outputs k, which is equal to $q(0)$. This allows $\mathcal{B}$ to fully determine $q(x)$ as it now has d points and derive the key $k_d = q(d)$. In other words, it allows $\mathcal{B}$ 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 Section 5.1.1, 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. Therefore, it follows that ϵ must be negligible. □

5.4 Performance

We now 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 $O(dn + m \log^2 m)$. 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 $O(\sum_{i=1}^{m} n^3)$ field operations in $\mathbb{F}_q$.

When a rekeying process takes place, the new information to be broadcast is $PI_i = \langle ACV_i, (z_1, \ldots, z_{N_i}) \rangle$, $i = 1, 2, \cdots, m$, 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 $O(\sum_{i=1}^{m} n)$.

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.

6 Scheme 3: Access Tree AB-GKM

Before we present the third scheme, called access tree AB-GKM, we provide the rationale behind all three schemes. In the inline AB-GKM scheme, the policy P is embedded into the BGKM scheme itself. As discussed in Section 5, while this approach works for many different types of policies and is quite efficient, such an approach is not able to efficiently support threshold access control policies. Scheme 2, the 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, including threshold policies, we extend the threshold AB-GKM scheme to propose the access tree AB-GKM scheme. The access tree AB-GKM is the most expressive AB-GKM scheme out of the three schemes and can represent any policies supported by inline and threshold AB-GKM schemes. Like threshold AB-GKM, instead of embedding P in the BGKM scheme, we construct a separate BGKM instance for each attribute in access tree AB-GKM. Then, we embed 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. [16].

Table 2: 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

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 $\mathcal{T}$.

6.1 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 2 in order to construct our scheme. All the functions except sat are straightforward to implement. A brief description of sat follows:

The function 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 child nodes of x return 1, then sat($\mathcal{T}_x, \alpha$) returns 1 and 0 otherwise.

6.2 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 ($\geq n$), the cryptographic hash function H, the key space $\mathcal{KS}$, the secret space $\mathcal{SS}$, the set of issued secrets $\mathbf{S}$, 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, $\mathbf{Q}$ 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): Taking γ_i, the attribute set of Usr_i, as input, 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 d_x of the polynomial q_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 derive k bottom-up. Note that a user can obtain the key if and only if its 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 $k_x = v_x \cdot ACV_x$, 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 $\perp$.

Update(P) The group members change due to the similar reasons mentioned for the **Update** algorithm in Section 4.1. In such a situation, a new symmetric group key k' is selected and **KeyGen(P)** is invoked to generate the set of new public information tuples **PI'**. Like the previous two schemes, the secrets shared with existing users are not affected by the group change.

6.3 Security

If an unbounded adversary can break our access tree AB-GKM scheme in the security model for AB-GKM, a simulator can be constructed to break the modified ACV-BGKM scheme. Like the previous scheme, we only give a high-level detail of the reduction based proof as the proof is very similar.

Theorem 10. *If an adversary can break the access tree AB-GKM scheme in the security model for AB-GKM, then a simulator can be constructed to break the modified ACV-BGKM scheme with non-negligible advantage* $1/2 + \epsilon$ *where* $\epsilon = 1/q^N$.

Proof. Suppose that an unbounded adversary $\mathcal{A}$ using a set of attributes α as the challenge set that does not satisfy the access tree $\mathcal{T}$ breaks our scheme in the AB-GKM security model with advantage at most 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, which represents any type of threshold node, $\mathcal{A}$ 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.

The challenger constructs modified ACV-BGKM instances for each of the attributes and gives them to $\mathcal{B}$. $\mathcal{A}$ obtains secrets for each of the attributes in α. $\mathcal{B}$ sends the public information tuples and the access tree $\mathcal{T}$ to $\mathcal{A}$. Notice that $\mathcal{A}$ can easily derive the keys for any attribute in α, but it can derive the keys for any other attribute only with an advantage of ϵ. According to the assumption, $\mathcal{A}$ does not satisfy at least one attribute required to satisfy $\mathcal{T}$. Let that attribute be $attr_x$. $\mathcal{A}$ derives k_x from PI_x corresponding to one such unsatisfied leaf node with advantage ϵ. Therefore, $\mathcal{A}$ derives the group key k with an advantage of at most ϵ.

Like the proof in Section 5, $\mathcal{A}$ derives the group key k, after executing the phase 1 of the security game as many times and give k to $\mathcal{B}$. Now, $\mathcal{B}$ works downwards $\mathcal{T}$ to recover the keys for nodes originally unsatisfied by $\mathcal{A}$ using Lagrange interpolation. For example, using k and $t_r - 1$, $\mathcal{B}$ obtains the key k_{t_r} for the t_r^{th} child node of r. Finally, $\mathcal{B}$ obtains the key k_x for an unsatisfied leaf node x corresponding to $attr_x$. In other words, it allows $\mathcal{B}$ 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. As mentioned earlier, the probability of breaking the modified ACV-BGKM scheme by applying the **KeyDer** algorithm is a negligible $1/q^N$ where q is the ℓ bit prime number and N is the maximum number of users. Therefore, it follows that ϵ must be negligible. □

7 Example Application

Among other applications, fine-grained access control in a group setting using broadcast encryption is an important application of the AB-GKM schemes. We illustrate the access-tree AB-GKM scheme using a healthcare scenario [27, 31]. We refer the reader to our technical report [23] for more examples. A hospital (Svr) supports fine-grained access control on electronic health records (EHRs) [38, 13] 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. Table 3 shows the insurance plans supported by each doctor and nurse, identified by the pseudonym "Employee ID".

Table 3: Insurance Plans Supported by Doctors/Nurses

EmployeeID	**Role/level**	**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 4 shows the content of the user attribute matrix UA that the hospital maintains. (Small numbers are used for illustrative purposes.)

Table 4: 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

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 5 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 6.

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 chil-

Table 5: 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	$\mathsf{emp}_1, \mathsf{emp}_2, \mathsf{emp}_3, \mathsf{emp}_7, \mathsf{emp}_9$

Table 6: 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	$\mathsf{emp}_4, \mathsf{emp}_5, \mathsf{emp}_8, \mathsf{emp}_9$

dren 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:

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

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.

Table 7: Average Time for CP-ABE algorithms

Algorithm	Time (ms)
Setup	34.395
Key generation	26.725
Encryption	24.453
Decryption	13.415

8 Experimental Results

In this section we provide experimental results for the underlying optimized ACV-BGKM scheme used with all three AB-GKM schemes presented earlier. We compare our results with CP-ABE scheme, which is used in other AB-GKM schemes, with comparable security parameters.

The experiments were performed on a machine running GNU/Linux kernel version 2.6.32 with an Intel® Core™ 2 Duo CPU E8400 3.00GHz and 3.2 Gbytes memory. Only one processor was used for computation. Our prototype system is implemented in C/C++. We use V. Shoup's NTL library [33] version 5.4.2 for finite field arithmetic, and SHA-1 and AES-128 implementations of OpenSSL [26] version 1.0.0d for cryptographic hashing and symmetric key encryption. We use Bethencourt et. al.'s cpabe [4] library to gather experimental results for CP-ABE. The cpabe library uses PBC library [21] for pairing based cryptography.

We implemented the ACV-BGKM scheme with subset cover optimization. We utilized the complete subset algorithm introduced by Naor et al. [25] as the subset cover. All finite field arithmetic operations in ACV-BGKM scheme are performed in an 512-bit prime field. We used comparable and efficient pairing parameters for CP-ABE. The size of the base finite field is set to the 512-bit prime number
$8780710799663312522437781984754049815806883199414208211028653399266475630880222957078625179422662221423155858769582317459277713367317481324925129998224791$
and the group order to the 160-bit number
$730750818665451621361119245571504901405976559617$.

Following the well-known security practice, we generate symmetric keys and use them for encrypting documents. Then we encrypt such encryption keys with either the ACV-BGKM generated symmetric keys or the CP-ABE generated public keys. Therefore, in the experiments we measure the time to encrypt and decrypt the document encryption keys only. For all the ACV-BGKM experiments, we assume that 5% of users have left the group after executing the setup.

First we give experimental results for the most simplest case where a single attribute condition is considered. Then we provide, experimental results for multiple attribute conditions.

Table 7 shows the average time required to execute setup, key generation, encryption and decryption algorithms of CP-ABE scheme for one attribute condition. It should be noted that the time required for the above CP-ABE algorithms linearly increases with the number of attributes as CP-ABE needs to perform similar amount of computation for each attribute. Our previous work [24] provides detailed experimental results for the key generation, encryption and decryption algorithms of ACV-BGKM with the subset cover optimization.

Figure 1 reports the average time required to execute the key generation algorithm of ACV-BGKM and CP-ABE with different group sizes. In both ACV-BGKM and CP-ABE the time increases linearly with the group size. However, ACV-BGKM is much more effi-

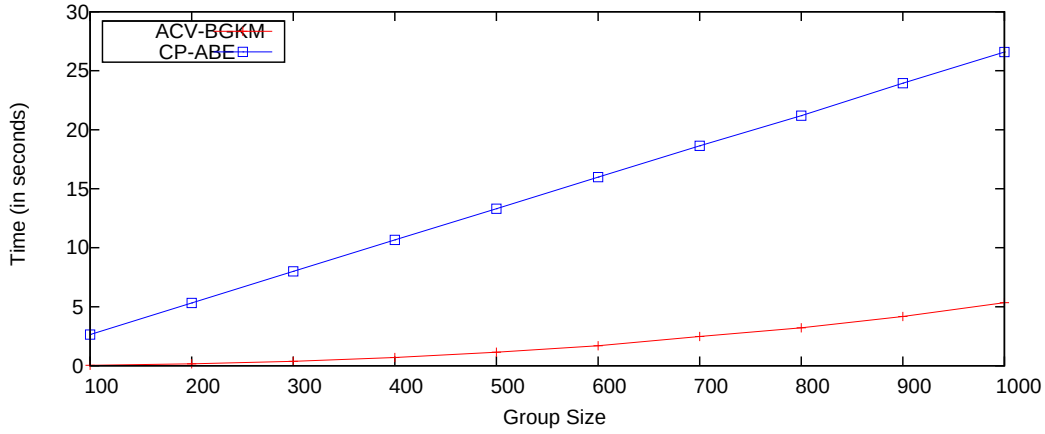

Figure 1: Average Key Generation Time for Different Group Sizes

cient as it does not involve any expensive pairing operations. It only uses efficient hashing and binary operations over a finite field. Further, the subset cover technique applied to ACV-BGKM reduces the computational complexity of the underlying scheme. Without the subset cover optimization, ACV-BGKM has a non-linear computational complexity and becomes inefficient for large groups. It should also be noted that if the number of revoked users are less than the setup value of 5%, ACV-BGKM scheme becomes even more efficient as less shared secrets are utilized for key generation.

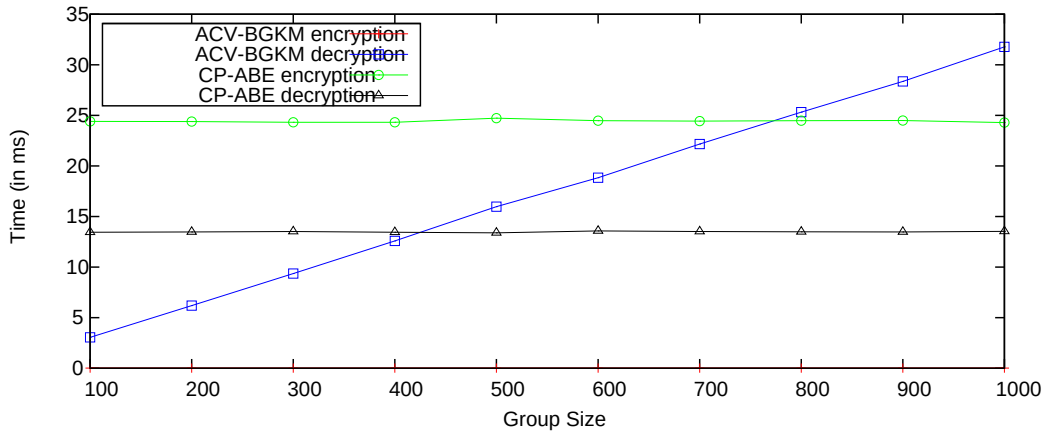

Figure 2: Average Encryption/Decryption Time for Different Group Sizes

Figure 2 reports the average time required to perform encryption and decryption in ACV-BGKM and CP-ABE schemes for one attribute condition with different group sizes. The decryption time of ACV-BGKM is taken as the time to derive the key as well as to decrypt the encryption key. The encryption and decryption times of CP-ABE remain constant whereas the decryption time of ACV-BGKM increases linearly with the group size. As the

group size increases, the key derivation algorithm of ACV-BGKM requires to spend more time to build larger KEVs. The encryption time of ACV-BGKM is negligible and remains constant as it involves an efficient symmetric encryption only. The average encryption time of ACV-BGKM is 8.8 microseconds (as these times are very small, the line plotting them is very close to zero in the graph in Figure 2 and thus overlaps with the x-axis). It should be noted that if one caches the KEVs, the decryption time of ACV-BGKM also becomes negligible as it involves only modular multiplications.

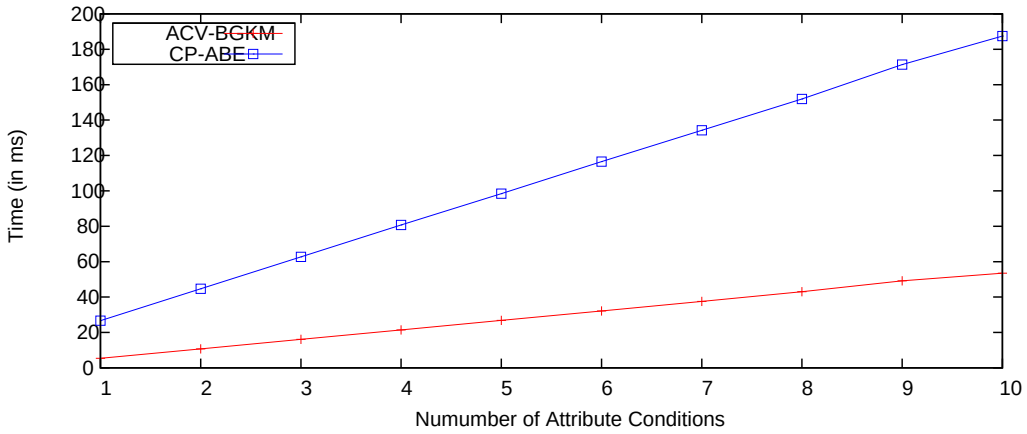

Figure 3: Average Key Generation Time for Varying Attribute Counts

Figure 3 reports the average time required to execute the key generation algorithm with varying number of attribute conditions with the group size set to 1000. The time of both techniques increases linearly with the number of attribute conditions. However, similar to Figure 1, the ACV-BGKM key generation is much more efficient than the CP-ABE key generation. This is due to the fact that, in ACV-BGKM key generation, only an additional hashing operation is performed for each new attribute condition, whereas, in CP-ABE key generation, additional paring operations are performed for each new attribute condition.

Figure 4 reports the average time required to execute the encryption/decryption algorithms with varying number of attribute conditions with the group size set to 1000 and the policy being the conjunction of all attribute conditions. Similar to the graph in Figure 2, ACV-BGKM encryption is negligible and independent of the the number of attributes. This is due to the fact that ACV-BGKM encryption is simply a symmetric key encryption operation. ACV-BGKM decryption time only increases slightly with the number of attribute conditions. This is due to the fact that the KEVs for ACV-BGKM decryption remain the same size and only an additional concatenation operation is performed for each new attribute condition. Unlike ACV-BGKM encryption/decryption, a noticeable increase in encryption/decryption is observed for CP-ABE scheme with each additional attribute condition. This is due to the fact that CP-ABE encryption/decryption operations need to perform same amount of pairing operations for each new attribute.

As can be seen from the experiments, our constructs are more efficient in handling scenarios where the key generation algorithm has to be executed frequently due to changes in user dynamics.

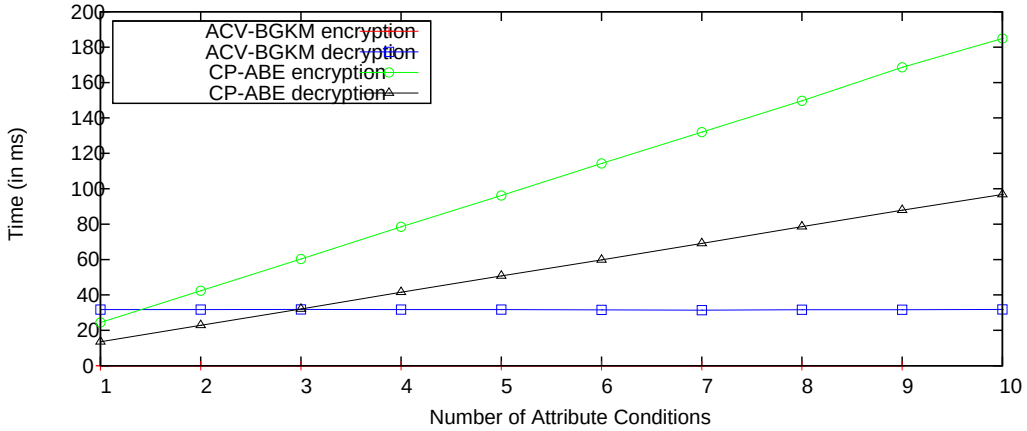

Figure 4: Average Encryption/Decryption Time for Varying Attribute Counts

9 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 private communication channels. Our schemes provide the same advantage when the group membership policies change. We have also shown that our schemes are resistant to collusion attacks. Our constructions are based on a provably secure ACV-BGKM scheme and Shamir's threshold scheme. Our experimental results show that our underlying construction is more efficient than the popular CP-ABE scheme. As future work, we plan to combine proposed schemes with hierarchical group key management schemes in order to further improve performance. The group key management schemes proposed in our work can be adopted to construct group agreement protocols [37]. We also plan to extend our work in this direction.

Acknowledgments

This material is based upon work supported by the Air Force Office of Scientific Research under Awards No. FA9550-08-1- 0260, FA9550-09-0468 and FA9550-08-1-0265.

References

[1] G. Ateniese, J. Kirsch, and M. Blanton. Secret handshakes with dynamic and fuzzy matching. In *NDSS 2007*.

[2] S. Berkovits. How to broadcast a secret. In *EUROCRYPT 1991*, pages 535–541.

[3] E. Bertino and E. Ferrari. Secure and selective dissemination of XML documents. *ACM Trans. Inf. Syst. Secur.*, 5(3):290–331, 2002.

[4] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext policy attribute based encryption library. http://http://acsc.cs.utexas.edu/cpabe/.

[5] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *SP 2007*, pages 321–334.

[6] J. Birkett and D. Stebila. Predicate-based key exchange. In *Information Security and Privacy*, volume 6168, pages 282–299. 2010.

[7] A. Boldyreva, V. Goyal, and V. Kumar. Identity-based encryption with efficient revocation. In *CCS 2008*, pages 417–426, 2008.

[8] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO 2005*, volume 3621, pages 258–275.

[9] E. Bresson, O. Chevassut, D. Pointcheval, and J.J. Quisquater. Provably authenticated group diffie-hellman key exchange. In *CCS 2001*, pages 255–264.

[10] E. Bresson and M. Manulis. Securing group key exchange against strong corruptions. In *ASI-ACCS 2008*, pages 249–260.

[11] H. Chu, L. Qiao, K. Nahrstedt, H. Wang, and R. Jain. A secure multicast protocol with copyright protection. *SIGCOMM Comput. Commun. Rev.*, 32(2):42–60, 2002.

[12] D. Dummit and R. Foote. Gaussian-Jordan elimination. In *Abstract Algebra*, page 404. Wiley, 2nd edition, 1999.

[13] M. Eichelberg, T. Aden, J. Riesmeier, A. Dogac, and G. B. Laleci. A survey and analysis of electronic healthcare record standards. *ACM Comput. Surv.*, 37(4):277–315, 2005.

[14] A. Fiat and M. Naor. Broadcast encryption. In *CRYPTO 1993*, volume 773, pages 480–491.

[15] M. C. Gorantla, C. Boyd, and G. Nieto. Attribute-based authenticated key exchange. In *ACISP 2010*, pages 300–317.

[16] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS 2006*, pages 89–98.

[17] Dani Halevy and Adi Shamir. The LSD broadcast encryption scheme. In *CRYPTO 2002*, pages 47–60.

[18] H. Harney and C. Muckenhirn. Group key management protocol specification. Technical report, Network Working Group, United States, 1997.

[19] J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In *CRYPTO 2003*, volume 2729, pages 110–125. 2003.

[20] Ram Krishnan, Ravi Sandhu, Jianwei Niu, and William H. Winsborough. Foundations for group-centric secure information sharing models. In *SACMAT 2009*, pages 115–124.

[21] B. Lynn. Pairing based cryptography library. http://crypto.stanford.edu/pbc/.

[22] G. Miklau and D. Suciu. Controlling access to published data using cryptography. In *VLDB 2003*, pages 898–909.

[23] M. Nabeel and E. Bertino. Attribute based group key management. Technical Report CERIAS TR 2010, Purdue University, 2010.

[24] M. Nabeel, N. Shang, and E. Bertino. Privacy preserving policy based content sharing in public clouds. *IEEE Trans. on Know. and Data Eng.*, 2012.

[25] Dalit Naor, Moni Naor, and Jeffrey B. Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO 2001*, pages 41–62.

[26] OpenSSL the open source toolkit for SSL/TLS. http://www.openssl.org/.

[27] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attribute-based systems. In *CCS 2006*, pages 99–112.

[28] A. Sahai, H. Seyalioglu, and B. Waters. Dynamic credentials and ciphertext delegation for

attribute-based encryption. In *CRYPTO 2012*, volume 7417, pages 199–217, 2012.

[29] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT 2005*, pages 457–473.

[30] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[31] N. Shang, M. Nabeel, F. Paci, and E. Bertino. A privacy-preserving approach to policy-based content dissemination. In *ICDE 2010*.

[32] A.T. Sherman and D.A. McGrew. Key establishment in large dynamic groups using one-way function trees. *IEEE Trans. on Soft. Eng.*, 29(5):444–458, May 2003.

[33] V. Shoup. NTL library for doing number theory. `http://www.shoup.net/ntl/`.

[34] Patrick Traynor, Kevin R. B. Butler, William Enck, and Patrick McDaniel. Realizing massive-scale conditional access systems through attribute-based cryptosystems. In *NDSS 2008*.

[35] W.-G. Tzeng and Z.-J. Tzeng. Round-efficient conference key agreement protocols with provable security. In *ASIACRYPT 2000*, ASIACRYPT '00, pages 614–628.

[36] C.K. Wong and S.S. Lam. Keystone: a group key management service. In *ICT 2000*.

[37] Qianhong Wu, Yi Mu, Willy Susilo, Bo Qin, and Josep Domingo-Ferrer. Asymmetric group key agreement. In *EUROCRYPT*, pages 153–170, 2009.

[38] XML in clinical research and healthcare industries. http://xml.coverpages.org/healthcare.html.

[39] X. Zou, Y. Dai, and E. Bertino. A practical and flexible key management mechanism for trusted collaborative computing. *INFOCOM 2008*, pages 538–546.