POSTER: Towards Attribute Based Group Key Management

Mohamed Nabeel
Dept. of Computer Science
Purdue University
West Lafayette, IN, USA
nabeel@cs.purdue.edu

Elisa Bertino
Dept. of Computer Science
Purdue University
West Lafayette, IN, USA
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 poster, we propose a novel key management scheme that allows users whose attributes satisfy a certain policy to derive the group key. Our scheme efficiently supports 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 scheme is expressive; it is able to support any monotonic policy over a set of attributes. Our scheme is resistant to collusion attacks; group members are unable to pool their attributes and derive the group key which they cannot derive individually.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Security, Algorithms, Design

1. INTRODUCTION

Current technological innovations and new application domains have pushed novel paradigms and tools for supporting collaboration among (possibly very dynamic) user groups. 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) [4, 1]; 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] 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. Further, they become inefficient when the group size is large. In this poster we show a novel expressive AB-GKM scheme which allows one to express any threshold or monotonic [1] conditions over a set of identity attributes. Further, we improve the performance of BGKM schemes by utilizing the concepts from subset-cover techniques [2].

A possible approach to construct an AB-GKM scheme is to utilize attribute-based encryption (ABE) primitives [4,

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

CCS'11, October 17–21, 2011, Chicago, Illinois, USA.
ACM 978-1-4503-0948-6/11/10.

1]. 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 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 construction addresses these shortcomings.

2. BACKGROUND

Our construction is based on the ACV-BGKM (Access Control Vector BGKM) scheme [5], a provably secure BGKM scheme, and Shamir's threshold scheme. We give an overview of ACV-BGKM in this section.

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 of requiring 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**, **SecGen**, **KeyGen**, **KeyDer**, and **Update**. We provide an overview of the construction of the ACV-BGKM scheme under a client-server architecture.

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 where

$$a_{i,j} = \begin{cases} 1 & \text{if } j = 0 \\ H(s_i || z_j) & \text{if } 1 \leq i \leq n, 1 \leq j \leq N, s_i \in \mathcal{S} \end{cases}$$

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 the above formula 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 $O(n)$. In our technical report [3], we propose using subset-cover techniques with BGKM to make the complexity sublinear in n.

3. OUR SCHEME

We use a modified version of ACV-BGKM scheme to construct our AB-GKM scheme. The idea of the modified version is that instead of giving each member in the group same intermediate key, each member is given a unique intermediate key in order to prevent collusion attacks. Our technical report [3] provides the details and security proofs of it. We construct a separate BGKM instance for each attribute and embed the policy P in an access structure $\mathcal{T}$. $\mathcal{T}$ is a tree with the internal nodes representing threshold gates and the leaves representing BGKM instances for attributes. $\mathcal{T}$ can represent any monotonic policy. The goal of the access tree is to allow deriving the group key for only the users whose attributes satisfy the access structure $\mathcal{T}$.

A high-level description of the access tree is as follows. Each threshold gate in the tree is described by its child nodes and a threshold value. The threshold value t_x of a node x specifies the number of child nodes that should be satisfied in order to satisfy the node. Each threshold gate is modeled as a Shamir secret sharing polynomial whose degree equals to one less than the threshold value. The root of the tree contains the group key and all the intermediate values are derived in a top-down fashion. A user who satisfies the access tree derives the group key in a bottom-up fashion.

Our 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 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): 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. We refer the reader to our technical report [3] for the detailed algorithm and security proofs.

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 $q_{parent(x)}(index(x))$, where $parent(x)$ is the parent node of x, $index(x)$ is the index of 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 construction of $\mathcal{T}$, 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 $\mathbf{K}_x = \{k_{i,j} | 1 \leq i \leq N, \mathsf{attr}_j\}$, $j = \mathsf{attr}(x)$, the index of attribute 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 $\mathcal{T}$.

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 $k_r = k$. 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 $s_x \notin \beta_i$, 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 child 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 $\perp$.

4. CONCLUSIONS

We presented a high-level view of a GKM scheme that supports 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 scheme eliminates the need of establishing private communication channels. Our scheme provides 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 scheme is 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.

We plan to implement our scheme and compare the performance of it with an ABE based scheme.

Acknowledgements

The work reported in this paper has been partially supported by the NSF grant 0712846 "IPS: Security Services for Healthcare Applications," and the MURI award FA9550-08-1-0265 from the Air Force Office of Scientific Research.

5. REFERENCES

[1] 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*, pages 89–98, New York, NY, USA, 2006. ACM.

[2] D. Halevy and A. Shamir. The lsd broadcast encryption scheme. In *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '02, pages 47–60, London, UK, 2002. Springer-Verlag.

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

[4] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Eurocrypt 2005, LNCS 3494*, pages 457–473. Springer-Verlag, 2005.

[5] 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.