Privacy Preserving Policy-Based Content Sharing in Public Clouds

Mohamed Nabeel, *Member*, *IEEE*, Ning Shang, and Elisa Bertino, *Fellow*, *IEEE*

Abstract—An important problem in public clouds is how to selectively share documents based on fine-grained attribute-based access control policies (acps). An approach is to encrypt documents satisfying different policies with different keys using a public key cryptosystem such as attribute-based encryption, and/or proxy re-encryption. However, such an approach has some weaknesses: it cannot efficiently handle adding/revoking users or identity attributes, and policy changes; it requires to keep multiple encrypted copies of the same documents; it incurs high computational costs. A direct application of a symmetric key cryptosystem, where users are grouped based on the policies they satisfy and unique keys are assigned to each group, also has similar weaknesses. We observe that, without utilizing public key cryptography and by allowing users to dynamically derive the symmetric keys at the time of decryption, one can address the above weaknesses. Based on this idea, we formalize a new key management scheme, called broadcast group key management (BGKM), and then give a secure construction of a BGKM scheme called ACV-BGKM. The idea is to give some secrets to users based on the identity attributes they have and later allow them to derive actual symmetric keys based on their secrets and some public information. A key advantage of the BGKM scheme is that adding users/revoking users or updating acps can be performed efficiently by updating only some public information. Using our BGKM construct, we propose an efficient approach for fine-grained encryption-based access control for documents stored in an untrusted cloud file storage.

Index Terms—Group key management, privacy, identity, cloud computing, policy, encryption, access control

1 Introduction

With the advent of technologies such as cloud computing, sharing data through a third-party cloud service provider has never been more economical and easier than now. However, such cloud providers cannot be trusted to protect the confidentiality of the data. In fact, data privacy and security issues have been major concerns for many organizations utilizing such services. Data often encode sensitive information and should be protected as mandated by various organizational policies and legal regulations. Encryption is a commonly adopted approach to protect the confidentiality of the data. Encryption alone, however, is not sufficient as organizations often have to enforce fine-grained access control on the data. Such control is often based on the attributes of users, referred to as *identity attributes*, such as the roles of users in the organization, projects on which users are working and so forth. These systems, in general, are called *attribute-based systems*. Therefore, an important requirement is to support fine-grained access control, based on policies specified using identity attributes, over encrypted data.

With the involvement of the third-party cloud services, a crucial issue is that the identity attributes in the access control policies (acps) often reveal privacy-sensitive information about users and leak confidential information about the content. The confidentiality of the content and the privacy of the users are, thus, not fully protected if the identity attributes are not protected. Further, privacy, both individual as well as organizational, is considered a key requirement in all solutions, including cloud services, for digital identity management [2], [3], [4], [5]. Further, as insider threats [6] are one of the major sources of data theft and privacy breaches, identity attributes must be strongly protected even from accesses within organizations. With initiatives such as cloud computing the scope of insider threats is no longer limited to the organizational perimeter. Therefore, protecting the identity attributes of the users while enforcing attribute-based access control both within the organization as well as in the cloud is crucial.

An approach to support fine-grained selective attribute-based access control is to encrypt each content portion to which the same access control policy (or set of policies) applies with the same key, and then upload the encrypted content to the cloud. One approach to deliver the correct keys to the users based on the policies they satisfy is to use a hybrid solution where the keys are encrypted using a public key cryptosystem such as attribute-based encryption (ABE) and/or proxy re-encryption (PRE). However, such an approach has several weaknesses: it cannot efficiently handle adding/revoking users or identity attributes, and policy changes; it requires to keep multiple encrypted copies of the same key; it incurs high computational cost. Therefore, a different approach is required.

It is worth noting that a simplistic group key management (GKM) scheme in which the content publisher directly delivers the symmetric keys to corresponding users has some major drawbacks with respect to user privacy and key management. On the one hand, user private information encoded in the user identity attributes is not protected in the simplistic approach. On the other

- M. Nabeel and E. Bertino are with the Department of Computer Science, Purdue University, 305 N. University Street, West Lafaytte, IN 47907. E-mail: {nabeel, bertino}@cs.purdue.edu.
- N. Shang is with the Qualcomm Inc., 5775 Morehouse Dr., San Diego, California 92121. E-mail: nshang@qti.qualcomm.com.

Manuscript received 2 May 2012; revised 7 Aug. 2012; accepted 8 Aug. 2012; published online 10 Sept. 2012.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2012-05-0303.
Digital Object Identifier no. 10.1109/TKDE.2012.180.

hand, such a simplistic key management scheme does not scale well as the number of users becomes large and when multiple keys need to be distributed to multiple users. The goal of this paper is to develop an approach which does not have these shortcomings.

We observe that, without utilizing public key cryptography and by allowing users to dynamically derive the symmetric keys at the time of decryption, one can address the above weaknesses. Based on this idea, we first formalize a new GKM scheme, called broadcast GKM (BGKM), and then give a secure construction of the BGKM scheme and formally prove its security. The idea is to give secrets to users based on the identity attributes they have and later allow them to derive actual symmetric keys based on their secrets and some public information. A key advantage of the BGKM scheme is that adding users/revoking users or updating acps can be performed efficiently and only requires updating the public information. Our BGKM scheme satisfies the requirements of *minimal trust*, *key indistinguishability*, *key independence*, *forward secrecy*, *backward secrecy*, and *collusion resistance* [7] with minimal computational, space, and communication cost.

Using our BGKM scheme, we develop an attribute-based access control mechanism whereby a user is able to decrypt the contents if and only if its identity attributes satisfy the content provider's policies, whereas the content provider and the cloud learn nothing about user's identity attributes. The mechanism is fine-grained in that different policies can be associated with different content portions. A user can derive only the encryption keys associated with the portions the user is entitled to access.

The remainder of the paper is organized as follows: Section 2 formalizes the notion of BGKM and provides our construction access control vector BGKM (ACV-BGKM) and security proofs. Section 3 provides two optimizations to the basic ACV-BGKM scheme. Section 4 provides an overview of our overall scheme that utilizes our optimized ACV-BGKM scheme presented in Sections 2 and 3 as a key building block. Section 5 shows how to preserve the privacy of identity attributes. Section 6 provides a detailed description of our scheme. Section 7 presents experimental results on the basic and the optimized ACV-BGKM construction which is the key component in our scheme. Section 8 discusses related work. Section 9 concludes the paper and outlines future research directions.

2 Broadcast Group Key Management (BGKM)

In this section, we first list the requirements for an effective GKM, then give an overview of BGKM schemes and finally present our construction along with security proofs.

2.1 Requirements for a Secure and Effective GKM

Several requirements are identified and discussed by Challel and Seba [7] and others for effective GKM. Generally speaking, an efficient and practical GKM should address the following requirements:

- *Minimal trust* requires the GKM scheme to place trust on a small number of entities.
- *Key hiding* requires that with given public information, it is hard for anyone outside the group to gain the shared group key. Ideally, every element in the keyspace should have the same probability of being the real key.
- *Key independence* requires that the leak of one key does not compromise other keys.
- *Backward secrecy* means that a member who has left the group cannot access any future group keys.
- *Forward secrecy* means that a newly joining group member cannot access any old keys.
- *Collusion resistance* requires that a set of colluding fraudulent users should not obtain keys which they are not allowed to obtain individually.
- *Low bandwidth overhead* requires that the rekeying should not incur a high volume of messages.
- *Computational costs* should be acceptable at both the server and the group member.
- *Storage requirements* for keys and other relevant information should be minimal.
- *Ease of maintenance* requires that a single change of membership in the group does not need many changes to take place for the other group members.
- *Other requirements* include service availability, minimal packet delays, and so on. These factors are sometimes more affected by real-world settings and implementation, and less related to the high-level design of the GKM.

2.2 Broadcast GKM

To provide forward and backward secrecy, rekey operations should be performed whenever the users in the group change. Typical GKM schemes require $O(n)$ [8], [9] or at least $O(\log n)$ [10], [11] private communication channels to perform the rekey operation. In comparison, BGKM schemes make rekey a one-off process [12], [13], [14]. In such schemes, rekeying is performed with a single broadcast without using private communication channels. It should be noted that even though BGKM schemes have some similarity with secret sharing (SS) schemes, they are constructed for different purposes. "*k out of n*" SS schemes [15], [16] are constructed to split a secret among n users and allow users to recover the secret by combining at least k secret shares. On the contrary, BGKM schemes allow each valid user to recover the secret by using only their secret share. Also, colluding users, who individually cannot recover the secret, are not able to recover the secret collectively. 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 SS. The subsequent rekeying operations are performed using a single broadcast message. Further, such schemes assure forward and backward security by only changing the public information and without affecting secret shares given to existing users. Based on our preliminary work [17], we propose a provably secure BGKM scheme, called ACV-BGKM, and formalize the notion of BGKM. Further we prove the security of ACV-BGKM.

Definition 1 (BGKM). *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* $\mathbf{S}$, *the secret space* $\mathcal{SS}$ *and the key space* $\mathcal{KS}$. *All the parameters are collectively denoted as* Param.

- ***SecGen()***: *It selects a random bit string* $s \notin \mathbf{S}$ *uniformly at random from the secret space* $\mathcal{SS}$, *adds* s *to* $\mathbf{S}$ *and outputs* s.
- ***KeyGen(S)***: *It chooses a group key* K *uniformly at random from the key space* $\mathcal{KS}$ *and outputs the public information* PI *computed from the secrets in* $\mathbf{S}$ *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* $\mathbf{S}$ *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.*

Now, we provide some basic notions and formally define security.

Negligible functions. We call a function $f : \mathbb{N} \rightarrow \mathbb{R}$ *negligible* if for every positive polynomial $p(\cdot)$ there exists an N such that for all $n > N$, we have $f(n) < 1/p(n)$ [18].

Random oracle model. The random oracle model is a paradigm introduced by Bellare and Rogaway [19] for design and analysis of certain cryptographic protocols. Intuitively, a random oracle is a mathematical function that can be queried by anyone, and maps every query to a uniformly randomly chosen response from its output domain. In practice, random oracles can be used to model cryptographic hash functions in many cryptographic schemes.

A BGKM scheme should allow a valid group member to derive the shared group key, and prohibit anyone outside the group from doing so. Formally speaking, a BGKM scheme should satisfy the following security properties. It must be correct, sound, key hiding, and forward/backward key protecting. Let Svr be the group controller.

Definition 2 (Correctness). *Let* Usr[1] *be a current group member with a secret. Let* K *and* PI *be* Svr*'s output of the* ***KeyGen*** *algorithm. Let* K' *be* Usr*'s output of the* ***KeyDer*** *algorithm. A BGKM scheme is correct if* Usr *can derive the correct group key* K *with overwhelming probability, i.e.,*

$$\Pr[K = K'] \geq 1 - f(k),$$

where f *is a negligible function in* k.

Definition 3 (Soundness). *Let* Usr *be an individual without a valid secret. A BGKM scheme is* sound *if the probability that* Usr *can obtain the correct group key* K *by substituting the secret with a value* val *that is not one of the valid secrets and then following the key derivation phase* ***KeyDer*** *is negligible.*

We define the following security game to define the key hiding requirement.

Definition 4 ($KeyHide_{\mathcal{A},\Pi}$).

1. *The* Svr, *as the challenger, runs the* ***KeyGen*** *algorithm of the BGKM scheme* Π *and gives the parameters* Param *to the adversary* $\mathcal{A}$.
2. $\mathcal{A}$ *selects two random keys* $K_0, K_1 \in \mathcal{KS}$ *and give them to the* Svr.
3. *The* Svr *flips a random coin* $b \in \{0, 1\}$ *and selects* K_b *as the group key and runs the* ***KeyGen*** *algorithm.*
4. *The* Svr *gives the public information* PI *of the output of the* ***KeyGen*** *algorithm to* $\mathcal{A}$.
5. $\mathcal{A}$ *outputs a guess* b' *of* b.
6. *The output of the game is defined to be 1 if* $b' = b$, *and 0 otherwise. We write* $KeyHide_{\mathcal{A},\Pi} = 1$ *if the output is 1 and in this case we say that* $\mathcal{A}$ *wins the game.*

The advantage of $\mathcal{A}$ *in this game is defined as* $Pr[KeyHide_{\mathcal{A},\Pi} = 1] - 1/2$.

Definition 5 (Key Hiding). *A BGKM scheme is* key hiding *if given* PI, *any party which does not have a valid secret cannot distinguish the real group key from a randomly chosen value in the keyspace* $\mathcal{KS}$ *with nonnegligible probability. More specifically, a BGKM scheme,* Π, *is* key hiding *if any adversary* $\mathcal{A}$ *as a probabilistic interactive Turing machine [20], has a negligible advantage in the key hiding security game in Definition 4:*

$$\mathbf{Pr}[KeyHide_{\mathcal{A},\Pi} = 1] \leq 1/2 + f(k),$$

where f *is a negligible function in* k.

Definition 6 (Forward/Backward Key Protecting). *Suppose* Svr *runs an* ***Update*** *algorithm to generate* Param *for a new shared group key* K', *and a previous member* Usr *is no longer a group member after the* ***Update*** *algorithm. Let* K *be a previous shared group key which can be derived by* Usr *with a secret. A BGKM scheme is* backward key protecting *if an adversary with knowledge of the secret,* K, *and the new* PI *cannot distinguish the new key* K' *from a random value in the keyspace* $\mathcal{KS}$ *with nonnegligible probability. Similarly, a BGKM scheme is* forward key protecting *if a new group member* Usr *after running the* ***Update*** *algorithm cannot learn anything about the previous group keys.*

2.3 Our Construction

We now provide our construction of BGKM, the ACV-BGKM scheme, under a client-server architecture. The ACV-BGKM scheme satisfies the requirements of *minimal trust*, *key indistinguishability*, *key independence*, *forward secrecy*, *backward secrecy*, and *collusion resistance* as described in Section 2.1.

The ACV-BGKM algorithms are executed with a trusted key server, Svr, and a group of users, $\mathrm{Usr}_i, i = 1, 2, \ldots, n$.

Setup(ℓ): 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 $\mathbf{S} = \emptyset$.

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

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

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

1. In what follows we use the term Usr; however, in practice, the steps are carried out by the client software transparently to the actual end user.

where

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

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

$$X = K \cdot e_1^T + Y,$$

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 X is called an ACV, *access control vector*. Svr lets $PI = \langle X, (z_1, z_2, \ldots, z_n)\rangle$, and outputs public PI and private K.

KeyDer(s_i, PI): Using its secret s_i and the public information PI, Usr_i computes $a_{i,j}, 1 \le j \le n$, as in (1) and sets an $(n+1)$-dimensional row $\mathbb{F}_q$-vector $v_i = (1, a_{i,1}, a_{i,2}, \ldots, a_{i,n})$. Usr_i derives the group key as $K' = v_i \cdot X$.

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

2.4 Security Analysis

In the security analysis of ACV-BGKM, we will model the cryptographic hash function H as a random oracle. We further assume $q = O(2^k)$ to be a sufficiently large prime. We first present two lemmas with their proofs and then prove the correctness, soundness and key hiding properties of ACV-BGKM. We also briefly analyze backward/forward key protection, minimal trust, key independence and collusion resistance properties of ACV-BGKM.

The following lemmas are useful for the security analysis of ACV-BGKM. Lemma 1 says that in a vector space V over a large finite field, the probability that a randomly chosen vector is in a preselected subspace, strictly smaller than V, is very small. Lemma 2 will be used in the proof of Theorem 2.

Lemma 1. *Let $F = \mathbb{F}_q$ be a finite field of q elements. Let V be an n-dimensional F-vector space, and W be an m-dimensional F-subspace of V, where $m \le n$. 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

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

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

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

□

Lemma 2. *Let $F = \mathbb{F}_q$ be a finite field of q elements. Let $v_i = (1, v_i^{(2)}, \ldots, v_i^{(n)}), i = 1, \ldots, m$, and $1 \le m < n$, be n-dimensional F-vectors. Let $v = (1, v^{(2)}, \ldots, v^{(n)})$ be an n-dimensional F-vector with $v^{(j)}, j \ge 2$ independently and uniformly randomly chosen from F. Then the probability that v is linearly dependent on $\{v_i, 1 \le i \le m\}$ is no more than $1/q^{n-m}$.*

Proof. Let $w_i = (v_i^{(2)}, \ldots, v_i^{(n)}), 1 \le i \le m$, and $w = (v^{(2)}, \ldots, v^{(n)})$. All w_i span an F-subspace W whose dimension is at most m in an $(n-1)$-dimensional F-vector space. w is a uniformly randomly chosen $(n-1)$-dimensional F-vector. By Lemma 1,

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

It follows that

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

□

Theorem 1. *ACV-BGKM is correct.*

Proof. The correctness of ACV-BGKM can be easily seen. Knowing its secret s_i and the public values $z_1, z_2, \ldots, z_n$, a group member Usr_i can compute one row of matrix A as

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

where $a_{i,j}, 1 \le j \le n$ are as in (1). Therefore, $v_i \cdot Y = 0$ for ACV Y, and thus the group key can be derived with probability 1 as

$$v_i \cdot X = v_i \cdot \left(K \cdot e_1^T + Y\right) = K \cdot v_i \cdot e_1^T = K.$$

□

Theorem 2. *ACV-BGKM is sound.*

Proof. Let Y be a given access control vector. Let $\{v_i, 1 \le i \le n\}$ be a basis of the nullspace of Y. Let $v = (1, v^{(2)}, \ldots, v^{(n+1)})$, where $v^{(i+1)} = H(\text{val} \| z_i), 1 \le i \le n$.

Usr can derive the group key using v by following the **KeyDer** phase if and only if v is linearly dependent on $v_i, 1 \le i \le n$. When val is not a valid secret and H is a random oracle, v is indistinguishable from a vector whose first entry is 1 and the other entries are independently and uniformly chosen from $\mathbb{F}_q$. By Lemma 2, the probability that v is linearly dependent on $\{v_i, 1 \le i \le n\}$ is no more than $1/q^{n+1-n} = 1/q$, which is negligible. This proves the soundness of ACV-BGKM. □

Theorem 3. *ACV-BGKM is key hiding.*

Proof. Let $\text{PI} = \langle X, (z_1, \ldots, z_n)\rangle$ be the public information broadcast by Svr. This is the only piece of information seen by the adversary that is related to the group key. By construction, X must be linearly independent from the standard basis vector e_1^T, i.e., X has a nonzero entry after the first position. For any $K \in \mathcal{KS} = \mathbb{F}_q$, let

$$Y = X - K \cdot e_1^T.$$

Then it is clear that all $\mathbb{F}_q$-vectors v such that $v \cdot Y = 0$ form an n-dimensional $\mathbb{F}_q$-vector space, say W. It follows that the n basis vectors of W can be chosen in such a way that they all have nonvanishing first entries. Therefore, the number of vectors v with 1 as their first entry such that $v \cdot X = K$ is q^{n-1}, for all $K \in \mathcal{KS}$. When the cryptographic hash function $H(\cdot)$ is modeled as a random oracle and a valid IST is unknown, every such a vector v assumes the same probability when computed as specified in the **KeyDer** algorithm. This implies that every $K \in \mathcal{KS}$ has the same probability, $1/q$, to be the designated group key in the view of the adversary. The key hiding property of ACV-BGKM follows as a direct consequence. Note that ACV-BGKM is key hiding against a computationally unbounded adversary. □

It is clear that "forward/backward key protection" is a stronger condition than "key hiding." However, we will use the proof of the latter to show the former.

Theorem 4. *ACV-BGKM is forward/backward key protecting.*

Proof (Sketch). We first consider the backward key protection property of ACV-BGKM. Suppose that after the **Update** algorithm, an adversary has one secret s from the previous session S_0 which does not propagate to the new session S_1. As the choices of s and the nullspace of the ACV in session S_0 can be viewed as (statistically) jointly independent from the determination of the nullspace of the ACV in session S_1, when H is modeled as a random oracle and by design of the **Update** algorithm, Usr cannot learn the group key for session S_1 with nonnegligible probability due to the key hiding property of ACV-BGKM.

Similarly, ACV-BGKM is forward key protecting. □

Other related GKM security aspects mentioned in Section 1 are briefly discussed in what follows.

Minimal trust. To protect the shared group key from an adversary outside of the group, ACV-BGKM only requires to use a private channel once between Svr and each Usr, during the **SecGen** algorithm. The security of the ephemeral private channels needs to be guaranteed. Any other communications, including the ones for key issuance and rekeying, are executed via an open broadcast channel.

Key independence. It is clear that the group keys (of different sessions) are independent by the ACV-BGKM construction. Furthermore, the secrets are also independent from each other, because they are randomly generated.

Collusion resistance. For BGKM, it only makes sense to consider collusion attacks from outside the group. The case that a valid group member passes its secret or the derived group key to others is not addressed by BGKM. Similar to the analysis for ACV-BGKM's forward/backward key protection property, ACV-BGKM is resistant to polynomially computationally bounded adversaries. In particular, colluding group members are not able to get the secrets of other members to derive group keys of earlier or later sessions.

3 Improvements to Basic ACV-BGKM

In this section, we improve the performance of our basic ACV-BGKM scheme using two techniques: bucketization and subset cover.

3.1 Bucketization

The proposed key management scheme works efficiently even when there are thousands of Usrs. However, as the upper bound n of the number of involved Usrs gets large, solving the linear system $AY = 0$ over a large finite field $\mathbb{F}_q$ becomes the most computationally expensive operation in our scheme. Solving this linear system with the method of Gaussian-Jordan elimination [21] takes $O(n^3)$ time. Although this computation is executed at the Svr, which is usually capable of carrying on computationally expensive operations, when n is very large, for example, $n = 100{,}000$, the resulting costs may be too high for the Svr. Due to the nonlinear cost associated with solving a linear system, we can reduce the overall computational cost by breaking the linear system in to a set of smaller linear systems. We follow a two-level approach. In this case, the Svr divides all the involved Usrs into multiple "buckets" (say m) of a suitable size (e.g., 1,000 each), computes an intermediate key for each bucket by executing the **KeyGen** algorithm, and then computes the actual group key for all the Usrs by executing the **KeyGen** algorithm with the intermediate keys as the secrets. Note that the intermediate key generation can be parallelized as each bucket is independent. The Svr executes $m + 1$ **KeyGen** algorithms of smaller size. The complexity of the **KeyGen** algorithm is proportional to $O(n^3/m^2 + m^3)$. It can be shown that the optimal solution is achieved when m reaches close to $n^{3/5}$.

To derive the group key, Usrs first derive the intermediate key from the m intermediate ACV-BGKM instances. Each intermediate key is associated with a marker so that Usrs can identify if they have derived a valid intermediate key using the **KeyDer** algorithm. One approach to implement the marker is to create a unique polynomial f of degree $m - 1$ such that $f(k_i) = 0$ for each intermediate key k_i. As Usrs do not know which intermediate ACV-BGKM instance they satisfy, Usrs are required to execute $m + 1$ **KeyDer** algorithms in the worst case, where Usrs derive m intermediate keys, and 2 in the best case, where Usrs derive only one intermediate key. Since the **KeyDer** algorithm is linear in n, in general, the bucketization optimization still improves the performance of the **KeyDer** algorithm. The complexity of the **KeyGen** algorithm is proportional to $O(n/m + m)$, but the average case, where Usrs derive $m/2$ intermediate keys, runs faster than the worst case.

3.2 Subset Cover

The bucketization approach becomes inefficient as the bucket size increases. The issue is that the bucketization still utilizes the basic ACV-BGKM scheme. In our basic ACV-BGKM scheme, as each Usr is given a single secret, it makes the complexity of PI and all algorithms proportional to n, the number of Usrs in the group. We utilize the result from previous research on broadcast encryption [22], [23] to improve the complexity to sublinear in n. Based on that, one can make the complexity sublinear in the number of Usrs by giving more than one secret during **SecGen** for each attribute Usrs possess. The secrets given to each Usr overlaps with different subsets of Usrs. During the **KeyGen**, Svr identifies the minimum number of subsets to which all the Usrs belong and uses one secret per the identified subset. During **KeyDer**, a Usr 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 Usrs invalid.

We give a high-level description of the basic *subset-cover* approach and how we integrate it into ACV-BGKM. In the basic subset-cover scheme, n Usrs are organized as the leaves of a balanced binary tree of height $\log n$. The construction of the binary tree results in a predefined user grouping as each Usr is assigned to a unique leaf node at the beginning. A unique secret is assigned to each vertex in the tree. The **SecGen** algorithm gives each Usr $\log n$ secrets that correspond to the vertices along the path from its leaf node to the root node of the tree. To provide backward secrecy when a single Usr is revoked, the updated tree is described by $\log n$ subtrees formed after removing all the vertices along the path from the Usr leaf node to the root node. To execute **KeyGen** or **Update** algorithms, Svr first identifies the set of secrets used as input as follows: Svr identifies the unique set of subtrees that covers the existing Usrs. Svr then selects the secret values assigned to the root nodes of the selected subtrees as the set of secrets. Using these secrets, Svr executes **KeyGen** or **Update** algorithms. These algorithms become sublinear in the number of existing Usrs as they are executed with a logarithmic number of secrets to cover all existing Usrs. To derive the group key, each Usr first identifies the subtree it belongs to and selects the secret value assigned to the root of the subtree as the input secret to the **KeyDer** algorithm. Note that each valid Usr has access to the secret assigned to the root of the subtree it belongs to due to how the above **SecGen** algorithm issues secrets. We implement Naor et al.'s complete subset scheme [22], which is an extension of the above mentioned basic subset-cover approach, in our experiments.

In our experimental results in Section 7, we show that combining the bucketization and the subset cover techniques, we can very efficiently execute the ACV-BGKM algorithms and can support very large user groups.

4 Overview of Our Scheme

As shown in Fig. 1, our scheme for policy-based content sharing in the cloud involves four main entities: the *data owner* (Owner), the *users* (Usrs) , the *identity providers* (IdPs), and the *cloud storage service* (Cloud). The interactions are numbered in the figure. Our approach is based on three main phases: *identity token issuance*, *identity token registration*, and *document management*.

1. *Identity token issuance.* IdPs issue *identity tokens* for certified identity attributes to Usrs. An identity token is a Usr's identity in a specified electronic format in which the involved identity attribute value is represented by a semantically secure cryptographic *commitment*.[2] We use the Pedersen commitment scheme and it is described in Section 5.1. Identity tokens are used by Usrs during the registration phase.
2. *Identity token registration.* To be able to decrypt the documents that will be downloaded from the Cloud, Usrs have to register at the Owner. During the registration, each Usr presents its identity tokens and receives from the Owner a set of secrets for each identity attribute based on the **SecGen** algorithm of the ACV-BGKM scheme. These secrets are later used by Usrs to derive the keys to decrypt the subdocuments for which they satisfy the access control policy using the **KeyDer** algorithm of the ACV-BGKM scheme. The Owner delivers the secrets to the Usrs using a privacy-preserving approach based on the oblivious commitment-based envelope (OCBE) protocols [24]. The OCBE protocols ensure that a Usr can obtain secrets if and only if the Usr's committed identity attribute value (within Usr's identity token) satisfies the matching condition in the Owner's access control policy, while the Owner learns nothing about the identity attribute value. Note that not only the Owner does not learn anything about the actual value of Usrs' identity attributes but it also does not learn which policy conditions are verified by which Usrs, thus the Owner cannot infer the values of Usrs' identity attributes. Thus, Usrs' privacy is preserved in our scheme. We give more details about the OCBE protocols in Section 5.2.
3. *Document Management.* The Owner groups the acps into *policy configurations* (PCs). The documents are divided into subdocuments based on the PCs. The Owner generates the keys based on the acps in each PC using the **KeyGen** algorithm of the ACV-BGKM scheme and selectively encrypts the subdocuments. These encrypted subdocuments are then uploaded to the Cloud. Usrs download encrypted subdocuments from the Cloud. The **KeyDer** algorithm of the ACV-BGKM scheme allows Usrs to derive the key K for a given PC using their secrets in an efficient and secure manner. With this scheme, our approach efficiently handles new users and revocations to provide forward and backward secrecy. The system design also ensures that acps can be flexibly updated and enforced by the Owner without changing any information given to Usrs.

2. A cryptographic commitment allows a user to commit to a value while keeping it hidden and preserving the user's ability to reveal the committed value later.

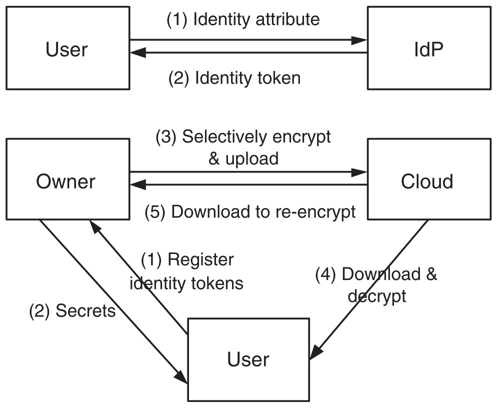


Fig. 1. Overall system architecture.

5 Preserving Privacy

As mentioned in Section 4, we utilize cryptographic techniques to protect the privacy of the identity attributes of the users from the Svr while executing the **SecGen** algorithm. Our technique makes sure that Usrs receive secrets only for valid identity attributes while the Svr does not learn the actual identity attribute values. We now give you an overview of the two cryptographic constructs, Pedersen commitments and OCBE protocols, that we use in this regard.

5.1 Pedersen Commitment

First introduced in [25], the Pedersen commitment scheme is an unconditionally hiding and computationally binding commitment scheme which is based on the intractability of the discrete logarithm problem. We describe how it works as follows.

Setup. A trusted third-party T chooses a finite cyclic group G of large prime order p so that the computational Diffie-Hellman problem is hard in G. Write the group operation in G as multiplication. T chooses two generators g and h of G such that it is hard to find the discrete logarithm of h with respect to g, i.e., an integer α such that $h = g^{\alpha}$. Note that T may or may not know the number α. T publishes (G, p, g, h) as the system's parameters.

Commit. The domain of committed values is the finite field $\mathbb{F}_p$ of p elements, which can be implemented as the set of integers $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$. For a party U to commit a value $x \in \mathbb{F}_p$, U chooses $r \in \mathbb{F}_p$ at random, and computes the commitment $c = g^x h^r \in G$.

Open. U shows the values x and r to open a commitment c. The verifier checks whether $c = g^x h^r$.

5.2 OCBE Protocols

The OCBE protocols, proposed by Li and Li [24], provide the capability of delivering information to qualified users in an oblivious way. There are three communications parties involved in OCBE protocols: a receiver R, a sender S, and a trusted third-party T. The OCBE protocols make sure that the receiver R can decrypt a message sent by S if and only if R's committed value satisfies a condition given by a predicate in S's acp, while S learns nothing about the committed value. Note that S does not even learn whether R is able to correctly decrypt the message or not. The supported predicates by OCBE are comparison predicates $>, \geq, <, \leq, =$ and $\neq$.

The OCBE protocols are built with several cryptographic primitives:

1. The Pedersen commitment scheme.
2. A semantically secure symmetric-key encryption algorithm $\mathcal{E}$, for example, AES, with key length k-bits. Let $\mathcal{E}_{\text{Key}}[M]$ denote the encrypted message M under the encryption algorithm $\mathcal{E}$ with symmetric encryption key Key.
3. A cryptographic hash function $H(\cdot)$. When we write $H(\alpha)$ for an input α in a certain set, we adopt the convention that there is a canonical encoding which encodes α as a bit string, i.e., an element in $\{0,1\}^*$, without explicitly specifying the encoding.

Given the notations as above, we summarize the OCBE protocol for only $\geq$ (GE-OCBE) predicate (due to space limitation) as follows: The OCBE protocols for other predicates can be derived and described in a similar fashion. The protocols' description is tailored to our work, and is stated in a slightly different way than in [24].

The GE-OCBE protocol works in a bit-by-bit fashion, for attribute values of at most ℓ bits long, where ℓ is a system parameter which specifies an upper bound for the bit length of attribute values such that $2^{\ell} < p/2$. The GE-OCBE protocol is more complex in terms of description and computation compared to EQ-OCBE (=). It works as follows:

Parameter generation. T runs a Pedersen commitment setup protocol to generate system parameters $\text{Param} = \langle G, g, h \rangle$, and outputs the order of G, p. In addition, T chooses another parameter ℓ, which specifies an upper bound for the length of attribute values, such that $2^{\ell} < p/2$. T outputs $\mathcal{V} = \{0, 1, \ldots, 2^{\ell} - 1\} \subset \mathbb{F}_p$, and $\mathcal{P} = \{\text{GE}_{x_0} : x_0 \in \mathcal{V}\}$, where

$$GE_{x_0} : \mathcal{V} \rightarrow \{\text{true}, \text{false}\},$$

is a predicate such that $\text{GE}_{x_0}(x)$ is true if and only if $x \geq x_0$.

Commitment. T chooses an integer $x \in \mathcal{V}$ for R to commit. T then randomly chooses $r \in \mathbb{F}_p$, and computes the Pedersen commitment $c = g^x h^r$. T sends x, r, c to R, and sends c to S.

Similarly, an offline alternative also works here.

Interaction

- R makes a data request to S.
- Based on the request, S sends to R a predicate $\text{GE}_{x_0} \in \mathcal{P}$.
- Upon receiving this predicate, R sends to S a Pedersen commitment $c = g^x h^r$.
- Let $d = (x - x_0) (\text{mod}\, p)$. R picks $r_1, \ldots, r_{\ell-1} \in \mathbb{F}_p$, and sets $r_0 = r - \sum_{i=1}^{\ell-1} 2^i r_i$. If $\text{GE}_{x_0}(x)$ is true, let $d_{\ell-1} \ldots d_1 d_0$ be d's binary representation, with d_0 the lowest bit. Otherwise if GE_{x_0} is false, R randomly chooses $d_{\ell-1}, \ldots, d_1 \in \{0,1\}$, and sets $d_0 = d - \sum_{i=1}^{\ell-1} 2^i d_i (\text{mod}\, p)$. R computes ℓ commitments $c_i = g^{d_i} h^{r_i}$ for $0 \leq i \leq \ell - 1$, and sends all of them to S.
- S checks that $cg^{-x_0} = \prod_{i=0}^{\ell-1} (c_i)^{2^i}$. S randomly chooses ℓ bit strings $k_0, \ldots, k_{\ell-1}$, and sets $k = H(k_0 \parallel \cdots \parallel k_{\ell-1})$. S picks $y \in \mathbb{F}_p^*$, and computes $\eta = h^y$, $C = \mathcal{E}_k[M]$, where M is the message containing requested data. For each $0 \leq i \leq \ell - 1$ and $j = 0, 1$, S computes $\sigma_i^j = (c_i g^{-j})^y$, $C_i^j = H(\sigma_i^j) \oplus k_i$. S sends to R the tuple

$$\langle \eta, C_0^0, C_0^1, \ldots, C_{\ell-1}^0, C_{\ell-1}^1, C \rangle.$$

Open. After R receives the tuple $\langle \eta, C_0^0, C_0^1, \ldots, C_{\ell-1}^0, C_{\ell-1}^1, C \rangle$ from S as above, R computes $\sigma_i' = \eta^{r_i}$, and $k_i' = H(\sigma_i') \oplus C_i^{d_i}$, for $0 \leq i \leq \ell - 1$. R then computes $k' = H(k_0' \parallel \cdots \parallel k_{\ell-1}')$, and decrypts C using key k'.

The EQ-OCBE protocol is simpler and more efficient compared to the GE-OCBE protocol. The OCBE protocol for the $\leq$ predicates (LE-OCBE) can be constructed in a similar way as GE-OCBE. Other OCBE protocols (for the

$\neq, <, >$ predicates) can be built on EQ-OCBE, GE-OCBE, and LE-OCBE.

All these OCBE protocols guarantee that the receiver R can decrypt the message sent by S if and only if the corresponding predicate is evaluated as true at R's committed value, and that S does not learn the satisfiability of the predicate.

6 Our Scheme

In this section, we describe our scheme in detail. As introduced in Section 4, our scheme has three phases: identity token issuance, identity token registration, and document management. We did not consider the technical details and privacy in Section 4. In this section, we make our scheme privacy preserving using the techniques introduced in Section 5. We explain our approach using the ACV-BGKM scheme with the subset cover optimization as a key building block.

6.1 Identity Token Issuance

The IdP runs a Pedersen commitment setup algorithm to generate system parameters $\mathrm{Param} = \langle G, g, h \rangle$. The IdP publishes Param as well as the order p of the finite group G. The IdP also publishes its public key for the digital signature algorithm it uses. Such parameters are used by the IdP to issue *identity tokens* to Usrs. We assume that the IdP first checks the validity of identity attributes Usrs hold[3]. Usrs present to the IdP their identity attributes to receive *identity tokens* as follows: For each identity attribute shown by a Usr, the IdP encodes the identity attribute value as $x \in \mathbb{F}_p$ in a standard way, and issues the Usr an identity token. An identity token is a tuple

$$\mathcal{IT} = (\mathrm{nym}, \text{id-tag}, c, \sigma),$$

where nym is a pseudonym for uniquely identifying the Usr in the system, id-tag is the tag of the identity attribute under consideration, $c = g^x h^r$ is a Pedersen commitment for the value x, and σ is the IdP's digital signature for $\mathrm{nym}, \text{id-tag}$, and c. The IdP passes values x and r to the Usr for the Usr's private use. We require that all identity tokens of the same Usr have the same nym,[4] so that the Usr and its identity tokens can be uniquely matched with a nym. Once the identity tokens are issued, they are used by Usrs for proving the satisfiability of the Pub's acps; Usrs keep their identity attribute values hidden, and never disclose them in clear during the interactions with other parties.

Example 1. Suppose a Usr Bob presents his driver's license to IdP to receive an identity token for his age. IdP assigns Bob a pseudonym pn-1492. IdP deduces from the birth date on Bob's driver's license that Bob's age is $x = 28$. The IdP randomly chooses a value $r = 9{,}270$, and computes a Pedersen commitment $c = g^x h^r$. The IdP then digitally signs the message containing Bob's pseudonym, a tag for "age" and the commitment c. The identity token Bob receives from the IdP may look like this

$$\mathcal{IT} = (\text{pn-}1492, \mathrm{age}, 6267292101, 949148425702313975).$$

6.2 Identity Token Registration

We assume that the Owner defines a set of acps, denoted as $\mathcal{ACPB}$, that specifies which subdocuments Usrs are authorized to access. ACPs are formally defined as follows:

Definition 7 (Attribute Condition). *An attribute condition* cond *is an expression of the form: "*name_A op l,*" where* name_A *is the name of an identity attribute* A*, op is a comparison operator such as* $=, <, >, \leq, \geq, \neq$*, and* l *is a value that can be assumed by attribute* A.

Definition 8 (Access Control Policy). *An access control policy (acp) is a tuple* $(s, o, \mathcal{D})$ *where:* o *denotes a set of portions (subdocuments)* $\{\mathcal{D}_1, \ldots, \mathcal{D}_t\}$ *of document* $\mathcal{D}$*; and* s *is a Boolean formula of attribute conditions* $\mathrm{cond}_1, \ldots, \mathrm{cond}_n$ *that must be satisfied by a Usr to have access to* o.[5]

Example 2. The acp

$$(\text{“level} \geq 58\text{”} \wedge \text{“role} = \text{nurse”}$$
$$\{\text{physical exam, treatment plan}\}, \text{“EHR.xml”})$$

states that a Usr of level no lower than 58 and holding a nurse position has access to the subdocuments "physical exam" and "treatment plan" of document EHR.xml.

Different acps can apply to the same subdocuments because such subdocuments may have to be accessed by different categories of Usrs. We denote the set of acps that apply to a subdocument as PC.

Definition 9 (PC). *A PC for a subdocument* $\mathcal{D}_1$ *of a document* $\mathcal{D}$ *is a set of policies* $\{\mathrm{acp}_1, \ldots, \mathrm{acp}_k\}$ *where* $\mathrm{acp}_i, i = 1, \ldots, k$*, is an acp* $(s, o, \mathcal{D})$ *such that* $\mathcal{D}_1 \in o$.

There can be multiple subdocuments in $\mathcal{D}$ which have the same PC. For each PC of $\mathcal{D}$, the Owner randomly chooses a key K for a symmetric key encryption algorithm (e.g, AES), and uses K to encrypt all subdocuments associated with this PC. Therefore, if a Usr satisfies $\mathrm{acp}_1, \ldots, \mathrm{acp}_m$, the Owner must make sure that the Usr can derive all the symmetric keys to decrypt those subdocuments to which a PC containing at least one $\mathrm{acp}_i (i = 1, \ldots, m)$ applies.

As in our ACV-BGKM-based scheme the actual symmetric keys are not delivered along with the encrypted documents, a Usr has to register its identity tokens at the Owner to derive the symmetric encryption key from the PI stored at the Cloud. The **SecGen** algorithm of the ACV-BGKM scheme and the OCBE techniques are used to register user identity tokens in a privacy preserving manner. During the registration, a Usr receives a set of secrets, based on the identity attribute names corresponding to the attribute names in the identity tokens. Note that secrets are generated by the Owner only based on the names of identity attributes and not on their values. Therefore,

3. The IdP can verify the validity of Usr's identity either in a traditional way, for example, through a on-the-spot registration, or digitally over computer networks. We will not dive into the details of identity validity check in this paper.

4. In practice, this can be achieved by requesting the Usr to present a strong identifier that correlates with the identity being registered. Again, we will not discuss this process in this paper.

5. In what follow we use the dot notation to denote the different components of an acp.

a Usr may receive an encrypted set of secrets corresponding to a condition which has a value that the Usr′ identity attribute does not satisfy. However, in this case, the Usr will not be able to extract the secrets from the message delivering these secrets as shown in Section 5.2. Proper secrets are later used by a Usr to compute symmetric decryption keys for particular subdocuments of the encrypted documents, as discussed in the document management phase. The delivery of secrets is performed in such a way that the Usr can correctly receive secrets if and only if the Usr has an identity token whose committed identity attribute value satisfies an attribute condition in Owner's acp, while the Owner does not learn any information about the Usr's identity attribute value and does not learn whether Usr has been able to obtain the CSS.

To enable Usrs registration, the Owner first chooses the OCBE parameters: an ℓ'-bit prime number q, a cryptographic hash function $H(\cdot)$ whose output bit length is no shorter than ℓ', and a semantically secure symmetric-key encryption algorithm with key length ℓ' bits. The Owner publishes these parameters. The Owner also constructs a subset cover tree with n leaf nodes corresponding to each Usr for each distinct attribute condition in acps. Let SC_j be the subset cover for the attribute condition cond_j. Then for an acp in $\mathcal{ACPB}$ that a subscriber Usr_i under pseudonym nym_i wants to satisfy, Usr_i selects and registers an identity token $\mathcal{IT} = (\mathrm{nym}_i, \mathrm{id\text{-}tag}, c, \sigma)$ with respect to each attribute condition cond_j in acp. Note that Usr_i does not register only for the attribute condition which the Usr_i's identity token satisfies; to assure privacy, Usr_i registers its identity token for more attribute conditions whose identity attribute name matches the $\mathrm{id\text{-}tag}$ contained in the identity token. In this way, the Owner cannot infer from Usr_i's registration which condition Usr_i is actually interested in. Such measures greatly reduce the leaking of identity attributes due to insider threats.

The Owner checks if id-tag matches the name of the identity attribute in cond_j, and verifies the IdP's signature σ using the IdP's public key. If either of the above steps fails, the Owner aborts the interaction. Otherwise, the Owner selects the corresponding secrets from the subset cover SC_j for Usr_i. The Owner then starts an OCBE session as a sender (S) to obliviously transfer these secrets to Usr_i who acts as a receiver (R). The Owner maintains a matrix $\mathcal{T}$ to store if secrets are delivered to each Usr_i for each cond_j. Upon the completion of the OCBE session the Owner performs the following actions:

- If nym_i does not exist in the matrix, it first creates a row for it.
- It sets $r_{i,j}$ cell of $\mathcal{T}$ with respect to nym_i and cond_j.

We remark that all secrets are independent, so the above secret delivery process can be executed in parallel. Matrix $\mathcal{T}$ is used by the Owner to execute the **KeyGen** algorithm of the ACV-BGKM scheme.

Example 3. Table 1 shows an example of matrix $\mathcal{T}$. A Usr under pseudonym pn-0012 who has an identity token with respect to identity tag role registers for all attribute conditions ("$\mathrm{role} = \mathrm{doc}$" and "$\mathrm{role} = \mathrm{nur}$" are shown in Table 1) involving identity attribute role. This Usr does not register for attribute conditions "$\mathrm{level} \geq 59$," "$\mathrm{YoS} \geq 5$,"[6] and "$\mathrm{YoS} < 5$," either because it does not hold an identity token with identity tag level or YoS, thus cannot register, or because it chooses not to register as it only needs to access subdocuments whose associated acp does not require conditions for these attributes. A drawback of registering only for the conditions required is that it may allow an attacker to infer certain attributes about the Usr with high confidence. To protect against such attacks the Usr may choose to register for more than one condition as explained earlier. Note that the Usr under pn-0829 registers for both conditions $\mathrm{YoS} \geq 5$ and $\mathrm{YoS} < 5$, which are mutually exclusive and thus both cannot be satisfied by any Usr. The registration for both conditions is crucial for privacy in that it prevents the Pub from inferring from the Usr's registration behavior which condition the Usr is actually interested in. A Usr under pn-1492 registers for all five attribute conditions.

TABLE 1
A Table of CSSs Maintained by the Pub

nym	level ≥ 59	YoS ≥ 5	YoS < 5	role $= \mathrm{doc}$	role $= \mathrm{nur}$	$\ldots$
pn-0012	$\perp$	$\perp$	$\perp$	1	1	$\ldots$
pn-0829	1	1	1	$\perp$	$\perp$	$\ldots$
pn-1492	1	1	1	1	1	$\ldots$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$

6.3 Document Management

Recall that the Owner encrypts all subdocuments with the same PC applicable with the same symmetric key. Therefore, the Owner execute the **KeyGen** algorithm of the ACV-BGKM for each PC. For a given PC, the Owner first identifies the secrets to be considered as follows:

- The Owner first converts each acp into disjunctive normal form. For each unique conjunctive term, it executes the remaining steps.
- Let ith conjunctive term be $\bigwedge_{j=1}^{\phi_i} \mathrm{cond}_j$, where the term has ϕ_i conditions. The Owner iterates through the secrets matrix $\mathcal{T}$, and finds the set of users who have been issued secrets to all the conditions in each conjunctive term.
- At the end of the previous step, the Owner has the list of Usrs who satisfy the PC, their association with the subset covers SC_i for each applicable cond_i. The Owner identifies the covers in each SC_i and the secrets corresponding the covers. The Owner aggregates by concatenating secrets in the order of the conditions in the conjunctive terms to produce a single secret for each user satisfying the conjunctive terms. For example, if the conjunctive term is $\mathrm{cond}_1 \wedge \mathrm{cond}_3$ and Usr_5 satisfies the term, the Owner obtains the cover secrets s_1 and s_3 from SC_1 for Usr_5 and SC_3 for Usr_5, respectively. The aggregated secret is $s_1 \| s_3$.

The set of aggregated secrets from the above algorithm is used as the input to the **KeyGen** algorithm which produces the public information PI and the symmetric group key k. The Owner creates an index of the public information tuples and associate with the encrypted subdocuments, and uploads them to the Cloud.

6. YoS means "years of service."

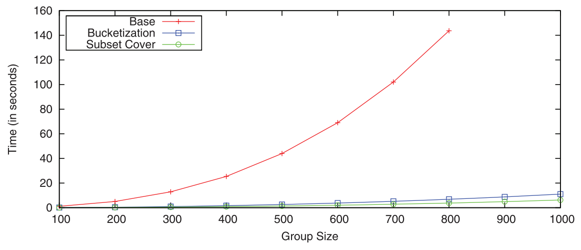


Fig. 2. Average time to generate keys.

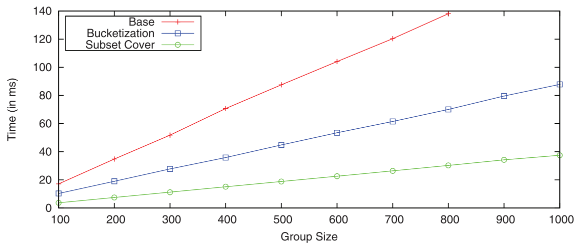


Fig. 3. Average time to derive keys.

If a Usr with a pseudonym nym_i wants to view the subdocument $\mathcal{D}_1$, it first downloads the encrypted subdocument along with the PI. It then picks an acp_k that it satisfies and derive the key using the **KeyDer** algorithm.

Now, we look at how to handle system dynamics such as adding/revoking credentials and acp updates.

When a new user Usr registers at the Owner, the Owner delivers the corresponding secrets to Usr, and updates the matrix $\mathcal{T}$. The Owner then performs a rekey process for all involved subdocuments (or equivalently, PCs) using the **Update** algorithm. When Owner uploads new documents, it also uploads the updated PI index.

During credential revocations, the conditions under which a Usr needs to be revoked is out of the scope of this paper. We assume that the Owner will be notified when a Usr with a pseudonym nym_i is revoked from those who may satisfy cond_j. In this case, the Owner simply reset the value $r_{i,j}$ from matrix $\mathcal{T}$, and performs a rekey process for all involved subdocuments. Allowing particular secrets to be deleted from $\mathcal{T}$ enables a fine-tuned user management.

A Usr's credentials may have to be updated over time for various reasons such as promotions, change of responsibilities, and so on. In this case, the Usr with a pseudonym nym_i submits the updated credential $cond_j$ to the Owner. The Owner simply resets the old $r_{i,j}$ entry and set a new entry in the matrix $\mathcal{T}$, and performs a rekey process only for the subdocuments involved.

When a Usr with a pseudonym nym_i needs to be removed, the Owner removes the row corresponding to nym_i from the matrix $\mathcal{T}$, and performs a rekey process only for the subdocuments involved.

Note that in all cases of new subscription, credential revocation, credential update and subscription revocation, the rekey process does not introduce any cost to Usrs in that except for those whose identity attributes are added, updated or revoked, no Usr needs to directly communicate with the Owner to update secrets. New encryption/decryption keys can be derived by using the original secrets and updated public values stored at the Cloud. The ability to derive the secret encryption/decryption keys using public values is a key point to achieve transparency in subscription handling. Most of the existing GKM schemes fail to achieve this objective.

7 Experimental Results

In this section, we present experimental results for the optimized ACV-BGKM scheme which is the heart of our scheme. We refer the reader to our preliminary work [17] for the experimental results on the OCBE protocols and the basic ACV-BGKM scheme.

The experiments were performed on a machine running GNU/Linux kernel version 2.6.32 with an Intel Core 2 Duo CPU T9300 2.50-GHz and 4-Gbytes memory. Only one processor was used for computation. The code is built with 32-bit gcc version 4.4.3, optimization flag -O2. For the ACV-BGKM scheme, we use V. Shoup's NTL library [26] version 5.4.2 for finite field arithmetic, and SHA-1 implementation of OpenSSL [27] version 0.9.8 for cryptographic hashing.

We implemented the ACV-GKM scheme with both the bucketization and the subset cover optimizations. We utilized the complete subset algorithm introduced by Naor et al. [22] for the subset cover. We assumed that 5 percent of the users satisfying a given PC are revoked. With the bucketization optimization, we assumed the average case for the **KeyDer** algorithm where Usrs require to derive half of the intermediate keys before deriving the group key. For the experiments involving a fixed number of buckets, 10 buckets are utilized. All finite field arithmetic operations in our scheme are performed in a 512-bit prime field.

Fig. 2 reports the average time spent to execute the **KeyGen** algorithm of the ACV-BGKM scheme without any optimization, with bucketization, and with subset cover optimization for different group sizes. The bucketization outperforms the base scheme as it divides the nonlinear **KeyGen** algorithm into smaller and more efficient computations. Subset-cover optimization provides even better performance as it reduces the effective group size considerably by sharing secrets among multiple Usrs. As shown in Fig. 3, the **KeyDer** algorithm has similar results.

Fig. 4 shows the average time to execute the **KeyGen** algorithm for 2,500 and 5,000 user groups with an increasing number of buckets. When more buckets are utilized, the size of the problem the **KeyGen** has to solve reduces and, hence, the bucketization provides a better performance. However, as mentioned in Section 3.1, the performance starts to degrade as the number of buckets is greater than the optimal number of buckets. For $n = 2{,}500$ and 5,000, the optimal number of buckets are around 100 and 150, respectively. These values are consistent with the

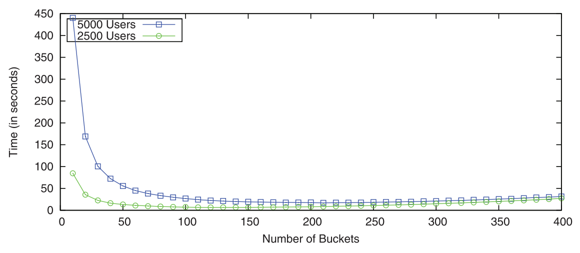


Fig. 4. Average time to generate keys with different bucket sizes.

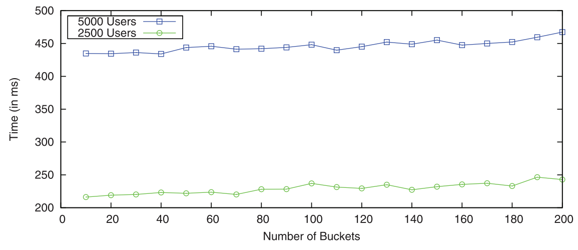


Fig. 5. Average time to derive keys with different bucket sizes.

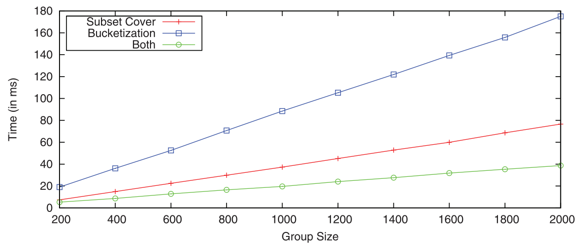


Fig. 7. Average time to derive keys with the two optimizations.

theoretical minimum overhead. Under similar settings, Fig. 5 shows the time to execute the **KeyDer** algorithm. The key derivation time slowly increases as the number of buckets increases because the complexity of the second level **KeyDer** function increases.

We closely analyzed the two optimizations. Fig. 6 shows the average time to execute the **KeyGen** algorithm with the bucketization, the subset cover, and both where the bucketization is applied after the subset cover technique. Both techniques together provides a huge performance improvement. Under the similar setting, as shown in Fig. 7, the **KeyGen** also performs much better compared to the individual optimizations.

8 Related Work

Approaches closely related to our work have been investigated in different areas: GKM, selective publication and broadcast of documents, and secure data outsourcing.

GKM. Section 2.2 discusses previous research efforts on the traditional GKM and the BGKM. Additionally, a recent research effort introduces a related BGKM approach based on access control polynomials [14]. This approach encodes secrets given to users at registration phase in a special polynomial of order at least n in such a way that users can derive the secret key from this polynomial. The special polynomials used in this approach represent only a small subset of domain of all the polynomials of order n, and the security of the approach is neither fully analyzed nor proven. Further, it appears that the security of the scheme weakens as n increases.

Selective dissemination. The database and security communities have carried out extensive research concerning techniques for the selective dissemination of documents based on acps [28], [29], [30]. These approaches fall in the following two categories.

1. Encryption of different subdocuments with different keys, which are provided to users at the registration phase, and broadcasting the encrypted subdocuments to all users [28], [29].
2. Selective multicast of different subdocuments to different user groups [30], where all subdocuments are encrypted with one symmetric encryption key.

The latter approaches assume that the users are honest and do not try to access the subdocuments to which they do not have access authorization. Therefore, these approaches provide neither backward nor forward key secrecy. In the former approaches, users are able to decrypt the subdocuments for which they have the keys. However, such approaches require all [28] or some [29] 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 process is not transparent, thus shifting the burden of acquiring new keys on existing users when others leave or join. Having identified these problems, our preliminary work [17], proposes an approach to make rekey transparent to users by not distributing actual keys during the registration phase. However, the security of the approach is not analyzed and it cannot handle large user groups.

Secure data outsourcing. With the increasing utilization of cloud computing services, there has been a real need to control access to encrypted documents stored in an untrusted third party. Our work falls into this category. There has been some recent research efforts [31], [32] to construct privacy preserving access control systems by combining oblivious transfer and anonymous credentials. The goal of such work is similar to ours but we identify the following limitations. Each transfer protocol allows one to access only one record from the database, whereas our approach does not have any limitation on the number of records that can be accessed at once since we separate the access control from the authorization. Another drawback is that the size of the encrypted database is not constant with respect to the original database size. Redundant encryption of the same record is required to support acps involving disjunctions. However, our approach encrypts each data item only once as we have made the encryption independent of acps. While ABE-based approaches [33] support expressive policies, they cannot handle revocations efficiently. Yu et al. [34] proposed an approach based on ABE utilizing PRE to handle the revocation problem of ABE. While it solves the revocation problem to some extent, it does not preserve the privacy of the identity attributes as in our approach.

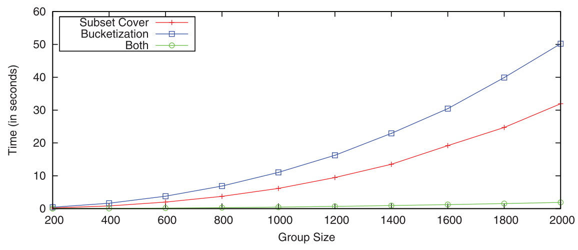


Fig. 6. Average time to generate keys with the two optimizations.

9 Conclusions

We have formalized the notion of BGKM and proved the security of our BGKM scheme, that is, the ACV-BGKM scheme. Further, we have proposed optimizations to significantly improve the performance of the ACV-BGKM scheme. Based on our BGKM scheme, we have proposed an approach to support attribute-based access control while preserving privacy of users' identity attributes for sharing documents in an untrusted cloud storage service. Our approach is supported by a new GKM scheme which is secure and allows qualified users to efficiently extract decryption keys for the portions of documents they are allowed to access, based on the subscription information they have received from the data owner. The scheme efficiently handles joining and leaving of guaranteed, with guaranteed security. Experimental results show that users efficiently derive decryption keys, and the data owner can efficiently large number of users.

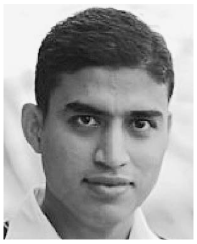
As future work, we plan to add traitor tracing and privacy preserving querying capabilities to our approach.

Acknowledgments

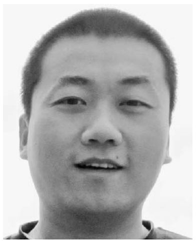
The work reported in this paper has been partially supported by the MURI award FA9550-08-1-0265 from the Air Force Office of Scientific Research. A preliminary version of this paper appeared in the Proceedings of the IEEE International Conference on Data Engineering (ICDE '10) [1].

References

[1] N. Shang, M. Nabeel, F. Paci, and E. Bertino, "A Privacy-Preserving Approach to Policy-Based Content Dissemination," *Proc. IEEE 26th Int'l Conf. Data Eng. (ICDE '10)*, 2010.
[2] "Liberty Alliance," http://www.projectliberty.org/, 2013.
[3] "OpenID," http://openid.net/, 2013.
[4] "Microsoft Windows CardSpace," http://msdn.microsoft.com/en-us/library/aa480189.aspx, 2013.
[5] "Higgins Open Source Identity Framework," http://www.eclipse.org/higgins/, 2013.
[6] R. Richardson, "CSI Computer Crime and Security Survey," http://www.ppclub.org/CSIsurvey2008.pdf, technical report, Computer Security Inst., 2008.
[7] Y. Challal and H. Seba, "Group key Management Protocols: A Novel Taxonomy," *Int'l J. Information Technology*, vol. 2, no. 2, pp. 105-118, 2006.
[8] H. Harney and C. Muckenhirn, "Group key Management Protocol (GKMP) Specification," technical report, Network Working Group, United States, 1997.
[9] H. Chu, L. Qiao, K. Nahrstedt, H. Wang, and R. Jain, "A Secure Multicast Protocol with Copyright Protection," *SIGCOMM Computer Comm. Rev.*, vol. 32, no. 2, pp. 42-60, 2002.
[10] C. Wong and S. Lam, "Keystone: A Group Key Management Service," *Proc. Int'l Conf. Telecomm. (ICT)*, 2000.
[11] A. Sherman and D. McGrew, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees," *IEEE Trans. Software Eng.*, vol. 29, no. 5, pp. 444-458, May 2003.
[12] G. Chiou and W. Chen, "Secure Broadcasting Using the Secure Lock," *IEEE Trans. Software Eng.*, vol. 15, no. 8, pp. 929-934, Aug. 1989.
[13] S. Berkovits, "How to Broadcast a Secret," *Proc. 10th Ann. Int'l Conf. Advances in Cryptology (EUROCRYPT '91)*, pp. 535-541, 1991.
[14] X. Zou, Y. Dai, and E. Bertino, "A Practical and Flexible Key Management Mechanism for Trusted Collaborative Computing," *Proc. IEEE INFOCOM*, pp. 538-546, Apr. 2008.
[15] A. Shamir, "How to Share a Secret," *Comm. ACM*, vol. 22, no. 11, pp. 612-613, 1979.
[16] E.F. Brickell, "Some Ideal Secret Sharing Schemes," *Proc. Workshop the Theory and Application of Cryptographic Techniques on Advances in Cryptology (EUROCRYPT '89)*, pp. 468-475, 1990.
[17] N. Shang, M. Nabeel, F. Paci, and E. Bertino, "A Privacy-Preserving Approach to Policy-Based Content Dissemination," *Proc. IEEE 26th Int'l Conf. Data Eng. (ICDE '10)*, 2010.
[18] O. Goldreich, *Foundations of Cryptography: Basic Tools.* Cambridge Univ. Press, 2000.
[19] M. Bellare and P. Rogaway, "Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols," *Proc. First ACM Conf. Computer and Comm. Security (CCS '93)*, pp. 62-73, 1993.
[20] S. Goldwasser, S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proof-Systems," *Proc. 17th Ann. ACM Symp. Theory of Computing (STOC '85)*, pp. 291-304, 1985.
[21] D. Dummit and R. Foote, "Gaussian-Jordan Elimination," *Abstract Algebra*, second ed., p. 404, Wiley, 1999.
[22] D. Naor, M. Naor, and J.B. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers," *Proc. 21st Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '01)*, pp. 41-62, 2001.
[23] D. Halevy and A. Shamir, "The LSD Broadcast Encryption Scheme," *Proc. 22nd Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '02)*, pp. 47-60, 2002.
[24] J. Li and N. Li, "OACerts: Oblivious Attribute Certificates," *IEEE Trans. Dependable and Secure Computing*, vol. 3, no. 4, pp. 340-352, Oct.-Dec. 2006.
[25] T. Pedersen, "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing," *Proc. 11th Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '91)*, pp. 129-140, 1992.
[26] V. Shoup, "NTL Library for doing Number Theory," http://www.shoup.net/ntl/, 2013.
[27] "OpenSSL the Open Source Toolkit for SSL/TLS," http://www.openssl.org/, 2013.
[28] E. Bertino and E. Ferrari, "Secure and Selective Dissemination of XML Documents," *ACM Trans. Information System Security*, vol. 5, no. 3, pp. 290-331, 2002.
[29] G. Miklau and D. Suciu, "Controlling Access to Published Data using Cryptography," *Proc. 29th Int'l Conf. Very Large Data Bases (VLDB '03)*, pp. 898-909, 2003.
[30] A. Kundu and E. Bertino, "Structural Signatures for Tree Data Structures," *Proc. VLDB Endowment*, vol. 1, no. 1, pp. 138-150, 2008.
[31] S. Coull, M. Green, and S. Hohenberger, "Controlling Access to an Oblivious Database Using Stateful Anonymous Credentials," *Proc. 12th Int'l Conf. Practice and Theory in Public Key Cryptography*, pp. 501-520, 2009.
[32] J. Camenisch, M. Dubovitskaya, and G. Neven, "Oblivious Transfer with Access Control," *Proc. 16th ACM Conf. Computer and Comm. Security (CCS '09)*, pp. 131-140, 2009.
[33] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," *Proc. IEEE Symp. Security and Privacy (SP '07)*, pp. 321-334, 2007.
[34] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute Based Data Sharing with Attribute Revocation," *Proc. Fifth ACM Symp. Information, Computer and Comm. Security (ASIACCS '10)*, pp. 261-270, 2010.



Mohamed Nabeel is currently working toward the PhD degree at the Department of Computer Science, Purdue University, West Lafayette, Indiana. His research interests include data privacy, distributed system security, and applied cryptography. His PhD thesis topic is Privacy Preserving Access Control in Third-Party Data Management Systems. His research adviser is Professor Elisa Bertino. He has published in the areas of privacy preserving content dissemination and group key management. He received the Fulbright fellowship in 2006, Purdue Cyper Center research grant in 2010, and Purdue research foundation grant in 2011. He is a member of the IEEE, the Center for Education and Research in Information Assurance and Security, and the ACM.



Ning Shang received the PhD degree in mathematics from Purdue University, West Lafayette, Indiana. He is a product security engineer at Qualcomm Inc. Before coming to Qualcomm, he was a postdoctoral researcher in the Department of Computer Science at Purdue University and a software development engineer at Microsoft. His research interests include algorithmic number theory, curve-based cryptography, and design and implementation of security systems.

Elisa Bertino is a professor of computer science at Purdue University, West Lafayette, Indiana, and serving as a research director of the Center for Education and Research in Information Assurance and Security and interim director of Cyber Center (Discovery Park). Previously, she was a faculty member and department head in the Department of Computer Science and Communication at the University of Milan. Her main research interests include security, privacy, digital identity management systems, database systems, distributed systems, and multimedia systems. She is currently serving as a chair of the ACM SIGSAC and as a member of the editorial board of the following international journals: *IEEE Security and Privacy*, *IEEE Transactions on Service Computing*, *ACM Transactions on Web*, and *ACM Transactions on Information and System Security*. She is the incoming editor in chief of the *IEEE Transactions on Dependable and Secure Computing* starting in 2014 and has also served as editor in chief of the *VLDB Journal*. She coauthored the book *Identity Management—Concepts, Technologies, and Systems*. She received the 2002 IEEE Computer Society Technical Achievement Award for outstanding contributions to database systems and database security and advanced data management systems and the 2005 IEEE Computer Society Tsutomu Kanai Award for pioneering and innovative research contributions to secure distributed systems. She is a fellow of the IEEE and the ACM.