CERIAS Tech Report 2010-03

BROADCAST GROUP KEY MANAGEMENT WITH ACCESS CONTROL VECTORS

by Ning Shang, Mohamed Nabeel, Elisa Bertino, Xukai Zou

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Broadcast Group Key Management with Access Control Vectors

Ning Shang [#], Mohamed Nabeel [#], Elisa Bertino [#], Xukai Zou [*]

[#]*Purdue University,*
West Lafayette, Indiana, USA
{nshang, nabeel, bertino}@cs.purdue.edu

[*]*Indiana University Purdue University Indianapolis*
Indianapolis, IN 46202, USA
xkzou@cs.iupui.edu

Abstract

Secure collaborative applications currently enabled by the Internet need flexible and efficient mechanisms for managing and distributing group keys. The secure transmission of information among collaborating users should be efficient as well as flexible in order to support access control models with different granularity levels for different kinds of applications such as secure group communication, secure dynamic conferencing, and selective/hierarchical access control disseminated information. In this paper, we propose the first provably secure broadcast Group Key Management (BGKM) scheme where each user in a group shares a secret with the trusted key server and the subsequent rekeying for join or departure of users requires only one broadcast message. Our scheme satisfies all the requirements laid down for an effective GKM scheme and requires no change to secret shares existing users possess. We analyze the security of our BGKM scheme and compare it with the existing BGKM schemes which are mostly ad-hoc.

Index Terms

Access Control, Broadcast Group Key Management, Security Model

I. INTRODUCTION

The rapid development of the Internet and the Web in past decades has significantly changed the way people live, work, learn, think, shop, and communicate all over the globe. The open nature of the Internet makes it a double-edged sword: On the one hand, telecommunication and exchange of information have never been faster, easier, and more effective; on the other hand, new forms of threats like worms, viruses, cyber crimes have emerged that compromise data/information security and user privacy, and have posed many open challenges to the world. Cryptographic techniques such as data encryption provide solutions for protecting information transmitted over the Internet. Efficient symmetric-key encryption algorithms like AES [1], Twofish [2], Blowfish [3], RC4 [4] are widely used in many Internet-based protocols and applications for securing data. In general, the strength of data encryption with a symmetric-key algorithm depends on the strength of the secret key, which must be known by all participating parties in communication. The process of selecting, distributing, storing and updating secret symmetric keys is called *key management*. Reliable, efficient and secure key management is usually a challenging problem in many real-world applications.

Group key management (GKM), as a specific case of key management, is related to the following scenario: Consider a server that sends data to a group of users in a multicast/broadcast[1] session through an open communication channel. To ensure data confidentiality, the server shares a secret group key K with all group members and encrypts the broadcast data using a symmetric encryption algorithm with K as the encryption key. Knowing the symmetric key K, any valid group member can decrypt the encrypted broadcast message. When the group dynamics changes, i.e., when a new user joins or an existing user leaves the group, a new group key must be generated and redistributed in a secure way to all current group members, so that a new group member cannot recover earlier transmitted data (backward secrecy), and a user who has left the group cannot learn anything from future communications in the group (forward secrecy). This process is called *update* or *rekeying*. The technique to maintain, distribute and update the group keys is called *group key management*. Group key management is a crucial component of secure group-oriented communications (SGC) and applications such as secure data broadcasting/dissemination [5], audio/video conferences, interactive online group games, file sharing, online publication, pay TV, and various emerging scenarios in cloud computing.

A number of approaches have been proposed for group key management. The *centralized approaches* [6], [7], [8], [9], [10], [11], [12], [13], [14] use a single trusted party to generate, distribute and update shared group keys. The *decentralized approaches* [15], [16], [17], [18], [19], [20] assume an infrastructure of group members and make use of multiple collaborating trusted entities to manage the group keys. In the *distributed approaches* [21], [22], [23], [24], [25], [26], [27], the data server can be treated as a group member, and all group members cooperate to compute the shared group keys. The aforementioned GKM schemes are good at supporting specific GKM scenarios such as SGC, as described above, secure dynamic conferencing (SDC) where any subset of a group of users can form a SGC and selective/hierarchical access control. However, they do not have the flexibility and efficiency to support combinations of these scenarios which we increasingly encounter in secure collaborative computing (SSC). A broadcast GKM (BGKM) scheme addresses these weaknesses. In such a scheme each user in a group shares a secret with a trusted key server and the subsequent rekeying for join or

[1]The key management discussed in this paper will be similar in multicast and broadcast cases. Therefore without loss of generality, we will only mention broadcast in the following discussions.

departure of users requires only one broadcast message and no change to secret shares existing users possess. However, the security of existing such schemes [28], [29], [30] have neither been analyzed fully nor proven formally.

A major difference between GKM protocols and secret sharing schemes, such as Shamir's (n, k)-threshold scheme [31], is that the former are designed to allow any individual group member to obtain a shared secret by itself, and no persistent secure communication channel is assumed between valid group members, whereas the latter are to prevent a single group member from gaining the secret alone, and require a secure communication channel, when group members combines the "secret shares", to protect the shared secret from being learned by parties outside the group. Another major difference between these two approaches is that the GKM protocols are intended for applications such as content broadcasting and collaborative applications, in which parties need to quickly gain accesses to the protected information. In some of these applications, parties may also need to keep private from other parties the fact that they are getting access to the protected information. Secret sharing schemes are more suitable for applications requiring separation of duty, in which no single party can be entrusted with the secret key.

Several requirements are identified and discussed by Challel and Seba [32] 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.
- **Forward secrecy** means that a member who has left the group cannot access any future group keys.
- **Backward 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.

Note that the trivial group key management in which the key server delivers new keys to each user though a secure private communication channel whenever rekeying occurs suffers from a high volume of communications and the fact that secure communication channels are in general costly to set up and maintain. Thus such a trivial approach is not practical for large scale systems. The problem becomes worse for SDC applications where there can be possibly $O(2^n)$ sub-groups, where n is the number of users, each sharing a unique key.

In this paper, we propose a new BGKM scheme which, to the best of our knowledge, is the first provably secure BGKM scheme. Our new scheme is flexible, efficient and secure. It keeps the use of secure private communication channels minimal by not requiring any private communications when rekeying takes place either among the group members or between the key server and a persisting group member. The size of the broadcast rekeying messages is linear with the total number of group members. In order to obtain a shared group key, a group member need only perform efficient hashing operations and an inner product of vectors over a finite field.

The following are the key contributions of our work:

- Formalization of the problem of BGKM.
- Introduction to a new BGKM scheme using the formalization.
- Thorough analysis of the security of the new scheme.
- Comparison of security and complexity of our scheme with existing BGKM schemes.
- Introduction to a faster variant of the proposed scheme.
- Empirical evaluation of the performance of the new scheme.

The rest of the paper is organized as follows. Section II discusses related work in BGKM. Section III formally defines BGKM, and proposes ACV-BGKM, our new solution to BGKM. Section IV analyzes ACV-BGKM with respect to security and efficiency, compares ACV-BGKM with existing BGKM schemes. Section V presents experimental results. Section VI discusses a possible scheme to improve the performance of ACV-BGKM. Section VII concludes the paper.

II. Related Work

Roughly speaking, we call a centralized group key management protocol a *broadcast group key management (BGKM)* scheme if it only uses a broadcast communication channel for rekeying. A formal definition of BGKM can be found in Section III. An important advantage of BGKM is that it is **easy to maintain**, in that an existing group member does not need to privately communicate with any other party when rekeying happens.

In this section we review existing BGKM protocols which are comparable to the new scheme we propose. We will compare them with our BGKM scheme later in Section IV-C.

CRT-BGKM. The first known BGKM scheme is proposed by Chiou and Chen [28] and is based on the concept of a *secure lock* implemented using the Chinese Remainder Theorem (CRT) [33]. The CRT-BGKM can be described as follows. There are a key server and a group of N members in the system considered by the scheme. The key sever first shares a secret value k_i with each of the group member, through a secure private communication channel.[2] The key server also publishes N (large) integers m_i that are pairwise relatively prime. The key server chooses a secret value K as the shared group key, encrypts K using a symmetric-key encryption algorithm with k_i as the encryption key to obtain a ciphertext K_i, and uses the CRT to compute an integer M such that $M = K_i \pmod{m_i}, 1 \leq i \leq N$. The key server then broadcast M to the group. For a group member to derive the symmetric key K, it computes $K_i = M \pmod{m_i}$ and decrypts K_i with its secret value k_i to get the desired group key K. When rekeying happens, a similar process is executed for all updated group members by only using the broadcast channel. The security of CRT-BGKM has not been formally analyzed so far.

SS-BGKM. A BGKM scheme proposed by Berkovits [29] is based on "k out of n" secret sharing. They present two examples using polynomial interpolation [31] and a related vector formulation [34]. In both examples, each of the N members are given a secret share and another $N + r$ (where $r > 0$) shares are given to all the users in the system. In other words, it creates a $N + r + 1$ out of $2N + r + 1$ secret sharing scheme. A valid user who has $N + r + 1$ shares can recover the secret, but others cannot. In the first example, each member reconstructs the secret using $N + r + 1$ shares whereas in the second example, the common $N + r$ shares

[2]Chiou and Chen proposed a similar BGKM using public-key cryptosystem [28]. However, such a scheme is less efficient than the private-key version summarized here, and thus we will not cover it.

are used in a pre-evaluation and only the needed result is given, to reduce the load on members. Like CRT-BKGM, when rekeying happens, the secret sharing system needs to be constructed again. Both variants are theoretically correct. However, it is not clear what security implications the proposed variants have due to certain assumptions made about the properties of secret shares and repeated use of the same shares for rekeying.

ACP-BGKM. Another BGKM scheme proposed by Zou, Dai and Bertino [30] is based on a construction called an *access control polynomial (ACP)*. The ACP-BGKM can be summarized as follows. Like in the case of CRT-BGKM and SS-BGKM, there are a key server and a group of N members. A finite field $\mathbb{F}_q$ and a cryptographic hash function $H(\cdot) : \{0,1\}^* \rightarrow \mathbb{F}_q$ are pre-chosen as public system parameters. The key server first chooses and sends a secret bit string sid_i to each group member through a secure private communication channel. The key server then picks a random bit string z, and creates an *access control polynomial*

$$A(x) = \prod_{i=1}^{N} \left(x - H(\mathsf{sid}_i||z)\right).$$

The key server randomly chooses $K \in \mathbb{F}_q$ as the shared group key, and sets $P(x) = A(x) + K$. The key server broadcasts $P(x)$ and z to the group. A group member with sid_i can derive the shared group key as $K = P\left(H(\mathsf{sid}_i||z)\right)$. When rekeying takes place, a new K is chosen by the key server, and new z and $P(x)$ are generated according to the updated group membership. Again, only the broadcast channel is used for rekeying. Unlike CRT-BGKM, no encryption algorithm is needed in ACP-BGKM. This makes ACP-BGKM potentially more efficient. However, like CRT-BGKM and SS-BGKM, no formal security analysis is available yet for ACP-BGKM. It is not clear so far what security assumption ACP-BGKM is based on. More specifically, as will be shown in Section IV-C, failing to satisfy the *key hiding* property, ACP-BGKM easily allows anyone to determine whether any given value is a valid group key using only part of the public information.

In Section III, we abstract out common features from the above three BGKM implementations, and formally define a general BGKM scheme as well as its relevant security notions. We propose a new and secure BGKM implementation, ACV-BGKM, which uses linear algebra and is based on a structure called *access control vector (ACV)*. ACV-BGKM is analyzed and compared with CRT-BGKM, SS-BGKM and ACP-BGKM in Section IV.

III. BGKM WITH ACCESS CONTROL VECTORS

In this section, we formally define a broadcast group key management scheme and its security, and propose a new group key management scheme which enables any valid member in the group which holds an individual subscription token (IST) to derive a common group key.

Definition 1 (BGKM): A broadcast group key management scheme (BGKM) is composed of two entities: 1) a key server (Svr), and 2) group members (Usrs), a persistent broadcast channel from Svr to all Usrs, an ephemeral private channel[3] between Svr and each individual Usr, and the following phases:

ParamGen Svr takes as input a security parameter k and outputs a set of public parameters Param, which includes the domain $\mathcal{KS}$ of possible key values.

TkDeliv Svr sends each Usr an *individual subscription token (IST)* through a private channel.

KeyGen Svr chooses a shared group key $K \in \mathcal{KS}$. Based on the ISTs of Usrs, Svr computes a set of values PubInfo. Svr keeps K secret, and broadcasts through the broadcast channel PubInfo to all group members Usr.

KeyDer Usr uses its IST and PubInfo to compute the shared group key K.

Update When the shared group K can no longer be used (e.g., when there is a change of group dynamics such as join and departure of group users), Svr generates new group key K' and PubInfo$'$, then broadcasts the new PubInfo to the group. Each Usr uses its IST and the new PubInfo$'$ to compute the new shared group key K'. We call the system after the Update phase a new "session". The Update phase is also called a rekeying phase.

A. Basic notions

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 $f(n) < 1/p(n)$ [35].

Random oracle model

The random oracle model is a paradigm introduced by Bellare and Rogaway [36] for design and analysis of certain cryptographic protocols. Intuitively, a random oracle is a mathematical

[3]In our BGKM definition, an ephemeral private channel is a secure communication channel which is used only once in the life time of a group member. One can think of an instantiation of such a channel as an onsite/face-to-face registration process.

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.

B. Security definitions

As motivated in Section I, with Svr being trusted, 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.

1) Correct: Let Usr be a current group member with an IST. Let K and PubInfo be Svr's output of the KeyGen phase. Let K' be Usr's output of the KeyDer phase. 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.

2) Sound: Let Usr be an individual without a valid IST. A BGKM is *sound* if the probability that Usr can obtain the correct group key K by substituting the IST with a value val that is not one of the valid ISTs and then following the key derivation phase KeyDer is negligible.

3) Key hiding: A BGKM is *key hiding* if given PubInfo, any party which does not have a valid IST cannot distinguish the real group key from a randomly chosen value in the keyspace $\mathcal{KS}$ with nonnegligible probability. More specifically, a BGKM is *key hiding* if for any adversary $\mathcal{A}$ as a probabilistic interactive Turing machine [37], $\Pr[\mathcal{A}$ wins the game in Fig. 1$] \leq 1/2 + f(k)$, where f is a negligible function in k.

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

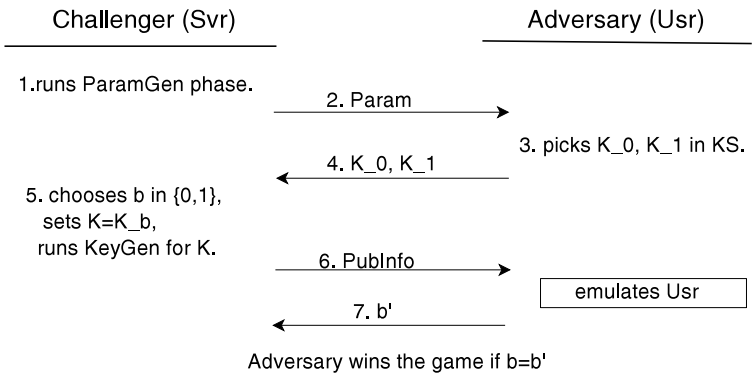

Fig. 1. The adversary game for BKGM's key hiding property. With the knowledge of PubInfo, the adversary is not able to distinguish one of its chosen keys from the other.

C. ACV-BGKM

In this section we describe a new BGKM scheme, ACV-BGKM, which is designed to be secure and efficient, and satisfies all the aforementioned requirements. ACV-BGKM uses an *access control vector (ACV)* which enables every valid group member to derive the shared group key.

Protocol 1 (ACV-BGKM): ACV-BGKM involves a trusted key server Svr and a group of members $\mathsf{Usr}_i, i = 1, 2, \ldots, n$. There is a long-term broadcast channel from Svr to all Usrs, and an ephemeral private channel between Svr and each individual Usr which is required only once.
ParamGen Svr takes a security parameter $k = \ell$. Svr chooses an ℓ-bit prime number q, a positive integer $N \geq n$ which represents the maximum allowed number of group members, and a cryptographic hash function $H(\cdot) : \{0,1\}^* \rightarrow \mathbb{F}_q$, where $\mathbb{F}_q$ is a finite field with q elements, which can be represented by $\{0, 1, \ldots, q-1\}$ with modular arithmetic. Svr sets the keyspace $\mathcal{KS} = \mathbb{F}_q$. Svr outputs via the broadcast channel $\mathsf{Param} = \langle \mathcal{KS}, N, H(\cdot) \rangle$.
TkDeliv For each $1 \leq i \leq n$, Svr chooses a random bit string $\mathsf{ist}_i \in \{0,1\}^*$ as an IST for each Usr_i, and sends ist_i to Usr_i using a private channel. Note that this is the only case when a private channel is used in the protocol. Svr saves these ist_i together with the group's membership information locally. Without loss of generality, we also assume that $\mathsf{ist}_i \neq \mathsf{ist}_j$ for $i \neq j$. In practice, an ist_i is chosen long enough (e.g., ≥ 80 bits) so that guessing becomes infeasible.

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

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

where

$$a_{i,j} = H(\mathsf{ist}_i || z_j), 1 \leq i \leq n, 1 \leq j \leq N. \tag{1}$$

Svr then solves for a nonzero $(N+1)$-dimensional column $\mathbb{F}_q$-vector Y such that $AY = 0$. Note that such a nonzero Y always exists as the nullspace of matrix A is nontrivial by construction. Here we require that Svr chooses Y from the nullspace of A uniformly randomly.[4] We call such a vector Y an *access control vector (ACV)*. 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 pre-chosen shared group key. Svr lets $\mathsf{PubInfo} = \langle X, (z_1, z_2, \ldots, z_N) \rangle$, and broadcasts PubInfo via the broadcast channel.

KeyDer Having ist_i and PubInfo, Usr_i computes $a_{i,j}, 1 \leq j \leq N$, as in formula (1) and sets an $(N+1)$-dimensional row $\mathbb{F}_q$-vector $v_i = (1, a_{i,1}, a_{i,2}, \ldots, a_{i,N})$. Usr_i derives the group key as $K' = v_i \cdot X$.

Update Svr runs the KeyGen phase again with respect to the current group users, creates a new group key $\hat{K}$ and random $\hat{z}_i, 1 \leq i \leq N$, and broadcasts $\widehat{\mathsf{PubInfo}} = \langle \hat{X}, (\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_N) \rangle$ via the broadcast channel. A current Usr derives the shared group key by following the same procedure specified in the KeyDer phase.

IV. Analysis

In this section, we prove the security of the ACV-BGKM and further analyze the scheme with respect to the requirements discussed in Section I.

[4]Note that the nullspace of A has dimension at least $N+1-n$. Let $\{v_1, \ldots, v_s\}$ be a basis of this nullspace. Then Y can be chosen as $Y = \sum_{i=1}^{s} \beta_i v_i$, where β_i are uniformly randomly chosen from $\mathbb{F}_q$. We want Y to be chosen in this way to prevent an adversary from successfully guessing by taking advantage of the fact that certain linear solvers output basis vectors of the nullspace in a patterned format (e.g., in Echelon form).

A. ACV-BGKM security analysis

In the security analysis of ACV-BGKM, we will model the cryptographic hash function H as a random oracle. We further assume $q = O(2^k)$ is a sufficiently large prime power and N is relatively small.

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 pre-selected 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 $m \leq n$. Let v be an F-vector uniformly randomly chosen from V. Then the probability that $v \in W$ is $1/q^{n-m}$.

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

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

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

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

■

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

Proof: Let $w_i = (v_i^{(2)}, \ldots, v_i^{(n)}), 1 \leq i \leq m$, 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,

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

It follows that

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

■

Theorem 1: ACV-BGKM is correct.

Proof: The correctness of ACV-BGKM can be easily seen: Knowing its IST ist_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 formula (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 $v^{(i+1)} = H(\mathsf{val}||z_i), 1 \le i \le N$. Usr can derive the group key using v by following the KeyDer phase if and only if v is linearly dependent of $v_i, 1 \le i \le N$. When val is not a valid IST 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 of $\{v_i, 1 \le i \le N\}$ 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 $\mathsf{PubInfo} = \langle X, (z_1, \ldots, z_N)\rangle$ be the public information broadcast from 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 of 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 phase. 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 protecting" 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 forward key protecting property of ACV-BGKM. Suppose that after the Update phase, an adversary has one IST ist from the previous session S_0 which do not propagate to the new session S_1. As the choices of ist and the nullspace of the ACV in session S_0 can be viewed as (statistically) jointly independent of 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 phase, Usr cannot learn the group key for session S_1 with non-negligible probability due to the key hiding property of ACV-BGKM.

Similarly, ACV-BGKM is backward key protecting. ■

Other related GKM security aspects mentioned in Section I are briefly discussed as follows.

Minimal trust. In order 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 TkDeliv phase. 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 ACV-BGKM construction. Furthermore, the ISTs are also independent of 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 IST or the derived group key to

others is not addressed by BGKM. Similar to the analysis for ACV-BGKM's forward/backward key protecting property, ACV-BGKM is resistant to polynomially computationally bounded adversaries. In particular, colluding group members are not able to get other members ISTs to derive group keys of earlier or later sessions.

B. ACV-BGKM efficiency analysis

We discuss the efficiency of ACV-BGKM with respect to computational costs and required bandwidth for rekeying.

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

For every rekeying phase, Svr needs to form a matrix A by performing N^2 hashing operations, and then solve a linear system of size $(N+1) \times N$. Solving the linear system is the most costly operation as N gets large for computation on Svr: It requires $O(N^3)$ field operations in $\mathbb{F}_q$ when the method of Gauss-Jordan elimination [38] is applied. Experimental results in Section V shows that this can be performed in a short time when N is up to 1000. Moreover, in practice, this process can be parallelized on Svr to further reduce the real execution time.

When a rekeying process takes place, the new information to be broadcast is $\mathsf{PubInfo} = \langle X, (z_1, \ldots, z_N) \rangle$, where X is a vector consisting of $(N+1)$ elements in $\mathbb{F}_q$, and without loss of generality we can pick z_i to be strings with a fixed length. This gives an overall communication complexity $O(N)$. An advantage of ACV-BGKM is that the peer-to-peer private channel is not needed for any persisting group members when rekeying happens.

Nowadays we generally care less about storage costs on both Svr and Usrs. Nevertheless, for a group of maximum N users, ACV-BGKM only requires each Usr to store its own IST ($O(1)$), and the Svr to keep track of all $O(N)$ ISTs of the group members.

C. Comparison with existing BGKM schemes

In this section, we compare ACV-BGKM with the CRT [28], SS [29] and ACP [30] approaches to BGKM from the view points of efficiency and security.

TABLE I

EFFICIENCY COMPARISON OF BGKM SCHEMES. N IS THE NUMBER OF GROUP MEMBERS. ℓ IS THE BIT LENGTH OF THE MODULI IN CRT-BGKM, AND THAT OF THE FINITE FIELDS IN ACP-BGKM, ACV-BGKM, AND SS-BGKM (WE ONLY DISCUSS THE POLYNOMIAL-BASED APPROACH IN THIS DISCUSSION). NOTE THAT AMONG ALL BGKM SCHEMES, ONLY ACV-BGKM IS PROVABLY SECURE.

	Comm.	**Usr**		**Svr**		**Key indish.**	**Need enc. alg.**
		Space	**Time**	**Space**	**Time**		
SS-BGKM	$O(N\ell)$	$O(\ell)$	$O(N^2\ell^2)$	$O(N\ell)$	$O(N\ell^2)$	Yes	No
CRT-BGKM	$O(N\ell)$	$O(\ell)$	$O(N\ell^2)$	$O(N\ell)$	$O(N\ell^2)$	Yes	Yes
ACP-BGKM	$O(N\ell)$	$O(\ell)$	$O(N\ell^2)$	$O(N\ell)$	$O(N^2\ell^2)$	No	No
ACV-BGKM	$O(N\ell)$	$O(\ell)$	$O(N\ell^2)$	$O(N\ell)$	$O(N^3\ell^2)$	Yes	No

In order to make a relatively fair comparison of the schemes, we assume that the secret values k_i, public moduli m_i in CRT-BGKM and the finite fields in ACP-BGKM and ACV-BGKM have about the same size so that the three schemes provide roughly the same level of security against brute-force attacks. Let this size be ℓ bits, and let N be the number of group members. It is clear that in all the schemes, $O(\ell)$ (disk) space is needed for a group member Usr to store the secret information obtained from the key server Svr through the private channel[5]. Assuming that a modular multiplication in $\mathbb{F}_q$ takes time $O(\ell^2)$ and a hashing operation takes constant time, the key derivation cost for a Usr is $O(N\ell^2)$ in both ACP-BGKM and ACV-BGKM. In CRT-BGKM, Usr needs to perform a long division between an $(N \cdot \ell)$-bit integer M and an ℓ-bit modulus m_i, followed by a symmetric-key decryption. When the schoolbook long division is used and assuming that the decryption operation takes $O(1)$ time, the overall complexity of key derivation in CRT-BGKM is also $O(N\ell^2)$. The key derivation cost of SS-BGKM is somewhat different: it requires a Usr to reconstruct the polynomial (at least its constant term) through polynomial interpolation; when the Lagrange interpolation formula is being used, it requires $O(N^2\ell^2)$ time. In all the algorithms, Svr takes $O(N)$ space to store group members' ISTs. In order to generate

[5]Depending on implementation, a group member may require more space (in memory) when performing computation for shared group key derivation.

the broadcast PubInfo for rekeying, in CRT-BGKM Svr uses CRT to construct the message M, which takes time $O(N\ell^2)$ when Garner's algorithm (Algorithm 2.1.7 in [39]) is used and schoolbook multiplication is considered; in ACP-BGKM Svr takes time $O(N^2\ell^2)$ to generate the access control polynomial, when polynomial multiplication is done in a straightforward way; in SS-BGKM Svr takes time $O(N)$ to create the polynomial, and time $O(N\ell^2)$ to generate the shares via polynomial evaluation, which produce an overall $O(N\ell^2)$; and in ACV-BGKM it takes time $O(N^3\ell^2)$ for Svr to compute an access control vector by solving a linear system with the method of Gauss-Jordan elimination. The communication complexity for rekeying is $O(N\ell)$ for all the BGKM schemes since only one broadcast of $O(N)$ elements is performed.

The efficiency of ACV-BGKM is comparable to the existing BGKM schemes, except that in ACV-BGKM the asymptotic order for Svr to generate the rekeying information is higher than that in the other three. However, in practice it can still be done very efficiently, as shown in Section V, even when there are thousands of members in the group. We also present an overview of a faster ACV-BGKM in Section VI. Most importantly, ACV-BGKM has the following advantages when compared with its counterparts.

First note that an encryption algorithm is needed for CRT-BGKM, but not for ACP-BGKM and ACV-BGKM. Given that, a one-to-one correspondence between the secret values k_i and the public moduli m_i must be kept by the server in CRT-BGKM, and also the size of the moduli m_i is required to be at least as large as the block size of the encryption algorithm.[6] SS-BGKM, ACP-BGKM and ACV-BGKM do not have these restrictions. Therefore SS-BGKM, ACP-BGKM and ACV-BGKM are easier to maintain than CRT-BGKM.

Another important fact to mention is that ACP-BGKM does not satisfy the *key hiding* (indistinguishability) property as defined in Section III-B. This can be easily seen in the following toy example.

Example 1: Let $N = 2$ and the finite field be $\mathbb{F}_2$ in ACP-BGKM. Suppose Svr broadcasts a polynomial $P(x) = x^2 + x + 1 \in \mathbb{F}_2[x]$ through the broadcast channel. Note that in this case the keyspace $\mathcal{KS} = \mathbb{F}_2$, and thus ideally the probability that 0 or 1 is the designated group key should be 0.5 for anyone without a valid IST sid. However, note that $P(x)$ is irreducible over

[6]If AES is used, each m_i must be at least 128-bit long.

$\mathbb{F}_2$; hence the only way to write $P(x)$ in the form $\prod_{i=1}^{N}(x-a_i)+K$ is

$$P(x) = (x-0)(x-1)+1.$$

Therefore, knowing only the polynomial $P(x)$, anyone can conclude that the shared group key K must be 1. This is against the indistinguishability of the key (IND-key) property of BGKM. In general, to win the game in Fig. 1 the adversary can compute two polynomials $A_0(x) = P(x) - K_0$ and $A_1(x) = P(x) - K_1$, and check if one of $A_0(x)$ and $A_1(x)$ fails to split completely into a product of linear polynomials in $\mathbb{F}_q[x]$: If it happens, then the other polynomial must correspond to the real group key. Numerical results have shown that when n gets large, the number of values $K \in \mathbb{F}_q$ such that $P(x) - K$ splits completed over $\mathbb{F}_q$ gets rare. Hence the adversary can win the game with nonnegligible probability.

Most importantly, compared to existing schemes, the newly proposed ACV-BGKM is more flexible and provably secure. The security of the existing BGKM schemes are not formally proven. A concerning fact about the existing BGKM schemes is that as the number of users increases, the security of the scheme appears to weaken. We summarize the above discussion in Table I.

V. Experimental results

In this section we evaluates the computational efficiency of ACV-BGKM. We simulate the KeyGen phase at Svr and the KeyDer phase at Usrs. In the experiment, we vary both the size of the underlying prime field $\mathbb{F}_q$ and the size of the group of Usrs, and measure the Svr-side and Usr-side computation time. To emphasize on the arithmetic operations, we do not count the time for hashing operations in the experiment. The code is written in the Magma scripting language [40], and uses Magma's internal library for finite field arithmetic and solving linear systems. The experiment was performed on a machine running GNU/Linux kernel version 2.6.9 with a Dual Core AMD Opteron(TM) Processor 2200 MHz and 16 Gbytes memory. Only one processor was used for computation. The experimental results are reported in Fig. 2 and Fig. 3.

Fig. 2 reports the ACV-BGKM running time at Svr and Usr for group sizes 600, 800, 1000 and 1200, and with the size of the prime field ranging from 64 to 128 bits. The running time is averaged over 20 iterations. As shown in the figure, the average computation time increases in

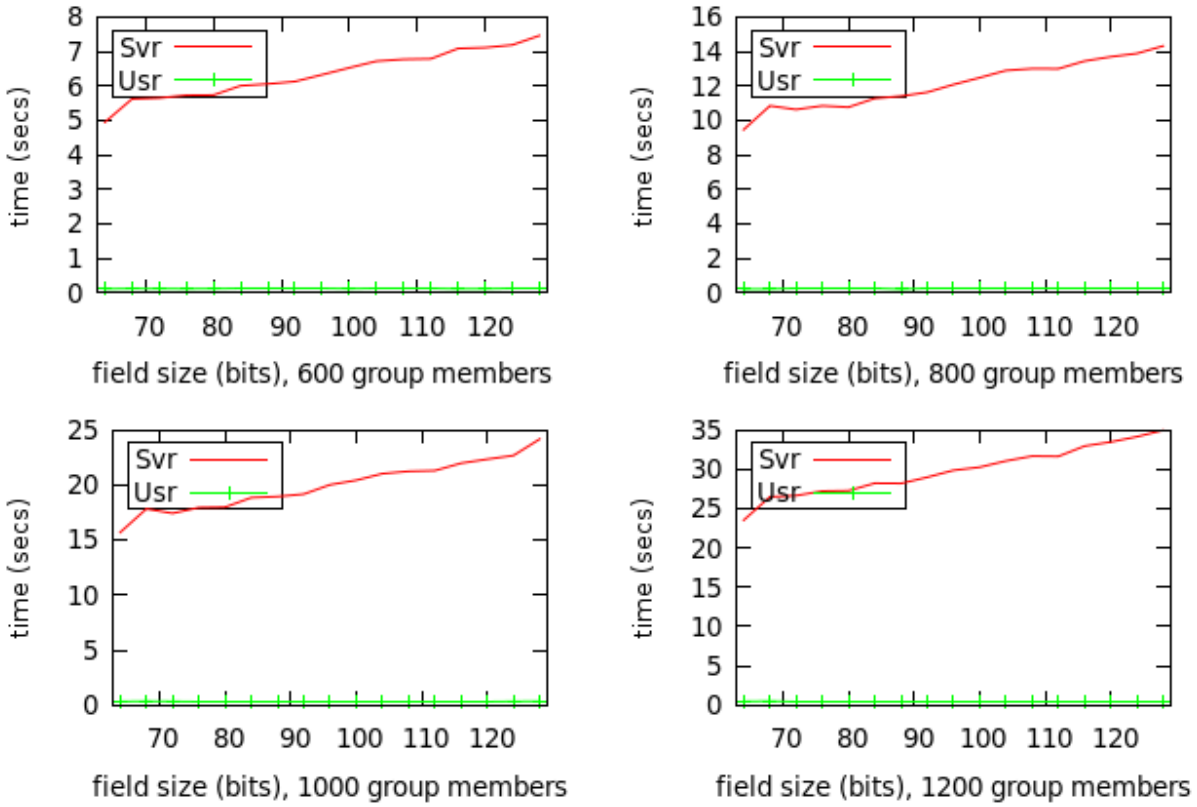

Fig. 2. Computation time at Svr and Usr for different $\mathbb{F}_q$

general as the size of the prime field increases. The actual running time depends on the prime field that is chosen and the way field arithmetic is performed in Magma.

Fig. 3 reports the ACV-BGKM running time at Svr and Usr for fixed field lengths (in bits) 64, 80, 96 and 112, with the size of the group ranging from 100 to 2000 members. The running time is averaged over 20 iterations. It shows that the ACV-BGKM rekeying process runs fast on Svr when there are hundreds of Usrs in the group. It takes less than two minutes for Svr to generate new PubInfo when there are up to 2000 Usrs and when the prime field is large enough.

Both figures show that it takes very little time for a Usr to derive the shared group key, and a practically short time for the Svr to generate the key and the broadcast rekeying information, when the underlying finite field and the group size are both considerably large. Further performance gains can be achieved when the prime number q is chosen to be in a special form, e.g., a generalized Mersenne prime (Solinas prime) [41], for which fast field arithmetic in $\mathbb{F}_q$ is

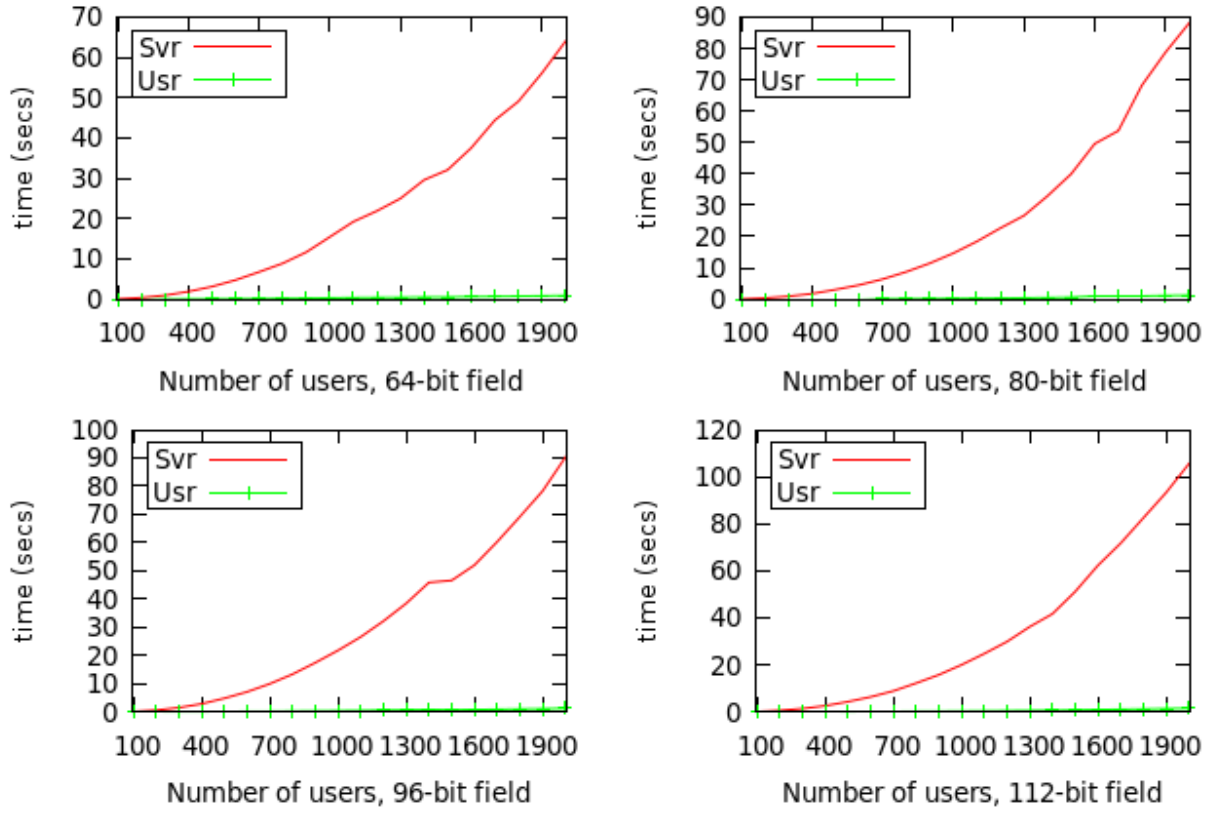

Fig. 3. Computation time at Svr and Usr for different group sizes

available.

VI. Towards a faster ACV-BGKM scheme

In this section we lay the foundation towards a faster protocol of ACV-BGKM, called FACV-BGKM, at the expense of additional space, precomputation and an index. It follows a baby-step-giant-step (BSGS) rekey mechanism where infrequent giant steps are performed analogous to the ACV-BGKM scheme. However, the amortized computational and communication cost is reduced by the introduction of frequent baby steps. Due to space constraints, we only describe the changes to the ACV-BGKM protocol below.

Protocol 2 (FACV-BGKM): FACV-BGKM works under similar conditions as ACV-BGKM. **ParamGen** Svr selects $N' = N + M$ where $M \leq N$. For the maximum security and minimum amortized cost, it is recommended to set $M = N$.

TkDeliv Svr assigns an index i ($1 \leq i \leq N$), selected uniformly at random, to each of the n current users . Svr chooses N ISTs and sends an IST and corresponding index to each user. The remaining $N - n$ precomputed ISTs are used for rekeying when new users join the group.
KeyGen Svr creates an $N \times (N + M)$ $\mathbb{F}_q$-matrix A where for a given $1 \leq i \leq N$

$$a_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } 1 \leq j \leq N \text{ and } i \neq j \\ H(\mathsf{ist}_i || z_j) & \text{if } N < j \leq N + M \end{cases}$$

Like in ACV-BGKM protocol, Svr computes the null space of A with a set of its M basis vectors, and selects an access control vector Y as one of the basic vectors. Svr caches these basic vectors and marks Y as "used." Svr constructs an $(N + M)$-dimensional $\mathbb{F}_q$-vector $X = (\sum_{i=1}^{n} K \cdot e_i^T) + Y$, where e_i is the i^{th} standard basis vector of $\mathbb{F}_q^{N+M}$. Notice that, unlike ACV-BGKM, the key is embedded to all the locations corresponding to valid indices. Like, ACV-BGKM, Svr sets PubInfo $= \langle X, (z_1, z_2, \ldots, z_M) \rangle$, and broadcasts PubInfo via the broadcast channel.
KeyDer Usr_i, knowing the index i and ist_i, derives the $(N + M)$-dimensional row $\mathbb{F}_q$-vector v_i which corresponds to a row in A. Usr_i derives the group key as $K = v_i \cdot X$.
Update Unlike ACV-BGKM, Svr does not run the complete KeyGen phase again. If a new Usr_t joins the group, Svr selects an unused index t and ist_t from the precomputed ISTs and computes the new $\hat{X}$ with a new key $\hat{K}$. If an existing Usr_r leaves the group, Svr selects a new key $\hat{K}$ and computes a new

$$\hat{X} = (\sum_{\substack{j=1 \\ j \neq i}}^{n} \hat{K} \cdot e_i^T) + \hat{Y},$$

where $\hat{Y}$ is an "unused" basis vector which is among the precomputed set in KeyGen phase. Svr marks $\hat{Y}$ as "used", and broadcasts only $\hat{X}$ while keeping the other public information unchanged. We call these operations a "baby-step rekey" since it only takes time $O(N)$ compared to $O(N^3)$ in ACV-BGKM. A complete KeyGen (i.e. "giant-step rekey") with time $O(N^3)$ needs to be performed every M Updates since otherwise a group member who has been valid for the last M sessions can recover the null space of A, thus the matrix A itself. A giant-step rekey also needs to be performed with a resized matrix A before M updates, if the number of joins exceeds $N - n$ after the current giant-step rekey to accommodate new users.

As described above, the KeyGen cost is amortized to obtain a scheme faster than the ACV-BGKM scheme. Due to space constraints, we omit the security/performance analysis of the FACV-BGKM scheme from this paper. We note that it is an interesting open research problem to decide the optimal M and N values depending on the application scenario.

VII. Conclusions

We have proposed a new BGKM scheme ACV-BGKM which is controlled by a trusted key server, and allows any valid user in the group to derive a shared group key on its own from broadcast public information. The scheme minimizes the usage of private peer-to-peer communication channels, and only uses a broadcast channel to deliver new rekeying messages when the group key needs to be changed. The communication overhead is linear with the number of users in the group. The scheme uses only efficient hash operations and linear algebra over finite fields in computation, and does not require any encryption scheme. It is secure in that even a computationally unbounded adversary cannot obtain the shared group key without a valid token from the key server. The key derivation is efficient for any group member. The experimental results show that the generation of the rekeying information takes a short time on a personal computer for a group of thousands of members. Such a scheme is useful to important practical applications like selective data dissemination, secure VoIP conferences, group-based online computer games, and so on. We have used the Magma computer algebra system to obtain the experimental results. We expect to see a better performance with more efficient software/hardware implementations of ACV-BGKM. As future work, we plan to empirically evaluate the performance of the FACV-BGKM scheme under different parameters.

References

[1] "Advanced Encryption Standard (AES)," http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[2] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, "Twofish," http://www.schneier.com/twofish.html.

[3] B. Schneier, "Blowfish," http://www.schneier.com/blowfish.html.

[4] "RC4," http://www.rsa.com/rsalabs/node.asp?id=2250.

[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.

[6] H. Harney and C. Muckenhirn, "Group key management protocol (GKMP) specification," http://tools.ietf.org/html/rfc2093, United States, 1997.

[7] ——, "Group key management protocol (GKMP) architecture," http://tools.ietf.org/html/rfc2094, United States, 1997.

[8] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The VersaKey framework: versatile group key management," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 1614–1631, 1999.

[9] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: Issues and architectures," ftp://ftp.isi.edu/in-notes/rfc2627.txt, United States, 1999.

[10] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions," in *Proc. IEEE INFOCOM'99*, vol. 2. New York, NY: IEEE, Mar. 1999, pp. 708–716.

[11] C. Wong, M. Gouda, and S. Lam, "Secure group communications using key graphs," *IEEE/ACM Trans. Netw.*, vol. 8, no. 1, pp. 16–30, 2000.

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

[13] A. Nemaney Pour, K. Kumekawa, T. Kato, and S. Itoh, "A hierarchical group key management scheme for secure multicast increasing efficiency of key distribution in leave operation," *Comput. Netw.*, vol. 51, no. 17, pp. 4727–4743, 2007.

[14] W. Ng, M. Howarth, Z. Sun, and H. Cruickshank, "Dynamic balanced key tree management for secure multicast communications," *IEEE Trans. Comput.*, vol. 56, no. 5, pp. 590–605, 2007.

[15] A. Ballardie, "Scalable multicast key distribution," http://tools.ietf.org/html/rfc1949, United States, 1996.

[16] R. Oppliger and A. Albanese, "Distributed registration and key distribution (DiRK)," *Information systems security: facing the information society of the 21st century*, pp. 199–208, 1996.

[17] L. Dondeti, S. Mukherjee, and A. Samal, "Scalable secure one-to-many group communication using dual encryption," *Computer Communications*, vol. 23, pp. 1681–1701, 2000.

[18] B. DeCleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan, and C. Zhang, "Secure group communications for wireless networks," in *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, vol. 1, 2001, pp. 113–117 vol.1.

[19] A. Schaff, "Dynamic group communication security," in *ISCC '01: Proceedings of the Sixth IEEE Symposium on Computers and Communications*. Washington, DC, USA: IEEE Computer Society, 2001, p. 49.

[20] S. Rafaeli and D. Hutchison, "Hydra: A decentralised group key management," in *WETICE '02: Proceedings of the 11th IEEE International Workshops on Enabling Technologies*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 62–67.

[21] I. Ingemarsson, D. Tang, and C. Wong, "A conference key distribution system," *Information Theory, IEEE Transactions on*, vol. 28, no. 5, pp. 714–720, Sep 1982.

[22] A. Fiat and M. Naor, "Broadcast encryption," in *CRYPTO '93: Proceedings of the 13th annual international cryptology conference on Advances in cryptology*. New York, NY, USA: Springer-Verlag New York, Inc., 1994, pp. 480–491.

[23] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman key distribution extended to group communication," in *CCS '96: Proceedings of the 3rd ACM conference on Computer and communications security*. New York, NY, USA: ACM, 1996, pp. 31–37.

[24] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," *Lecture Notes in Computer Science*, vol. 950, pp. 275–286, 1995.

[25] C. Boyd, "On key agreement and conference key agreement," in *ACISP '97: Proceedings of the Second Australasian Conference on Information Security and Privacy*. London, UK: Springer-Verlag, 1997, pp. 294–302.

[26] K. Becker and U. Wille, "Communication complexity of group key distribution," in *CCS '98: Proceedings of the 5th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 1998, pp. 1–6.

[27] Y. Kim, A. Perrig, and G. Tsudik, "Communication-efficient group key agreement," in *Sec '01: Proceedings of the 16th international conference on Information security: Trusted information*, 2001, pp. 229–244.

[28] G. Chiou and W. Chen, "Secure broadcasting using the secure lock," *Software Engineering, IEEE Transactions on*, vol. 15, no. 8, pp. 929–934, Aug 1989.

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

[30] X. Zou, Y. Dai, and E. Bertino, "A practical and flexible key management mechanism for trusted collaborative computing," in *INFOCOM*. IEEE, 2008, pp. 538–546. [Online]. Available: http://dblp.uni-trier.de/db/conf/infocom/infocom2008.html#ZouDB08

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

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

[33] T. Hungerford, "Chinese remainder theorem," in *Algebra (Graduate Texts in Mathematics)*. Springer, February 2003, pp. 131–132.

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

[35] O. Goldreich, *Foundations of Cryptography: Basic Tools*. New York, NY, USA: Cambridge University Press, 2000.

[36] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," in *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security*. New York, NY, USA: ACM, 1993, pp. 62–73.

[37] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1985, pp. 291–304.

[38] D. Dummit and R. Foote, "Gauss-Jordan elimination," in *Abstract Algebra*, 2nd ed. Wiley, 1999, p. 404.

[39] R. Crandall and C. Pomerance, *Prime numbers: A computational perspective*. New York: Springer-Verlag, 2001.

[40] W. Bosma, J. Cannon, and C. Playoust, "The MAGMA algebra system I: the user language," *J. Symb. Comput.*, vol. 24, no. 3-4, pp. 235–265, 1997.

[41] J. Solinas, "Generalized mersenne numbers," CACR, Tech. Rep. CORR 99-39, 1999.