IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 15, 2019, accepted December 23, 2019, date of publication January 3, 2020, date of current version July 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2963782

Group Key Management Protocol for File Sharing on Cloud Storage

SHOUYI ZHANG[1], SI HAN[2], BAOKUN ZHENG[2], KE HAN[3], AND ENTONG PANG[4],

[1]School of Mechanical, Electronic and Control Engineering, Beijing Jiaotong University, Beijing 100044, China
[2]Department of Science and Technology, China University of Political Science and Law, Beijing 102249, China
[3]School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100083, China
[4]Aerospace Hongka Intelligent Technology (Beijing) Company, Ltd, Beijing 100012, China

Corresponding author: Si Han (hansi@cupl.edu.cn)

This work was supported in part by the School Youth Fund Project of the China University of Political Science and Law under Grant 10818435, and in part by the Fundamental Research Funds for the Central Universities under Grant 2019PTB-016.

ABSTRACT The large-scale sharing needs of many enterprises promote the development of cloud storage. While the cloud computing stores the shared files outside the trust domain of the owner, the demands and concerns for file security is arising. In this paper, a Group Key Management Protocol for file sharing on cloud storage (GKMP) is proposed. Faced with network attacks from public channel, a group key generation scheme based on mixed encryption technology is proposed. And a verification scheme is used to prevent shared files from being attacked by the collusion attack of cloud providers' and group members'. Security and performance analyses indicate that the proposed protocol is both secure and efficient for data sharing in cloud computing.

INDEX TERMS Cloud storage, group key, file sharing, key distribution.

I. INTRODUCTION

Faced with today's innovative blow-up of cloud technologies, rebuilding services in terms of cloud have become more popular. In a shared-tenancy cloud computing environment, data from different clients which can be hosted on separate virtual machines may reside on a single physical machine [1]. Under this paradigm, the data storage and management is under full control of the cloud provider, so data owners are left vulnerable and have to solely rely on the cloud provider to protect their data. Recent news shows that Google provided the FBI all the documents of one of its users after receiving a search warrant, but the users have not been aware of the search until they are arrested.Because cloud provider has the full access to the data, the privacy of data could be violated if user's data is intercepted or modified by the cloud provider.

A common way to guarantee privacy is encrypting and authenticating the shared files [2]. There is a series of cryptographic schemes [3] under such circumstance that a third party auditor is able to check the availability of files while nothing about the file leaks. Likewise, cloud users probably will not hold the strong belief that the cloud server is doing a good job in terms of confidentiality. The cloud users are motivated to encrypt their files with their own keys before uploading them to the cloud server. The remaining challenge is how to share and manage the cryptographic keys among valid users without the participant of the cloud provider.

Theoretically, access control [4] and group key management [5], [6] can be used for key management on file sharing. However, some unique features of cloud storage introduce new problems that have not been fully considered [7], [8]. Firstly, shared files are transmitted via the network and the files may be intercepted by various network monitoring. Just using access control on the cloud storage cannot fully address this problem. Secondly, group key management depends on the cloud provider to manage the encryption key. That can prevent the shared files from intercepting by the network, while the shared files can be intercepted by the cloud provider.

In this paper, we proposed a secure group key management protocol on cloud storage over unreliable channels, aiming at protecting the shared files on the cloud storage. Mixed encrypiton techonology is used to generate and distribute group keys, which resistance attacks from network monitor. In addition, we propose a verified protocol that against the attacks from the file sharers or the cloud provider.

The rest of the paper is organized as follow. Section 2 discusses the related work. In section 3, we present our protocol

The associate editor coordinating the review of this manuscript and approving it for publication was Shagufta Henna.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

as well as notations to be used in this paper. Section 4 gives the details of our protocol. Section5, we address some security issues of our method and Section6 describes our prototype implementation on commodity hardware and software. Finally, the paper draws the conclusion in Section 7.

II. RELATED WORKS

The security of storage systems has always been an area of active research. There are many actual systems, such as CFS [9] and NASD [10]. CFS is tailored towards single-user workstations and relied on user-supplied passwords for data encryption. NASD proposes a distributed system comprising intelligent disks and users supplied keys as proofs of authorization. Approaches such as NASD and SNAD [11] focus mainly on securing network traffic and preventing out-side attacks.

Rao [12] proposed a secure sharing schemes of personal health records in cloud computing based on ciphertext-policy attributed-based(CP-ABE) signcryption [13]. It focus on restricting unauthorized users on access to the confidential data. Liu et al. [14] proposed an access control policy based on CP-ABE for personal records in cloud computing as well. In [12] and [14],only one fully trusted central authority in the system is responsible for key management and key generation.

Huang *et al.* [15] introduced a novel public key encryption with authorized equality warrants on all of its ciphertext or a specified ciphertext. To strengthen the securing requirement, Wu *et al.* [16] proposed an efficient and secure identity-based encryption scheme with equality test in cloud computing. Xu *et al.* [17] proposed a CP-ABE using bilinear pairing to provide users with searching capability on ciphertext and fine-grained access control. He *et al.* [18] proposed a scheme named ACPC aimed at providing secure, efficient and fine-grained data access control in P2P storage cloud. Recently, Xue *et al.* [19] proposed a new framework, named RAAC, to eliminate the single-point performance bottleneck of the exiting CP-ABE based access control schemes for public cloud storage. While these schemes use identity privacy by using attribute-based techniques which fail to protect user attribute privacy.

The most recent work addressing the privacy issues in a cloud-based storage is carried out by Pervez *et al.* [20], who proposed a privacy aware data sharing scheme SAPDS. It combines the attribute based encryption along with proxy re-encryption and secret key updating capability without relying on any trusted third party. But the storage and communication overhead of SAPDS is decided by attribute encryption scheme.

The above systems give an identical data access permission to groups of users, and any user who can access the shared files based on the access permission. These group permissions are typically used to secure the keys of data encryption. We can observe that how to securely share data files in a multiple-owner manner for groups while preserving identity privacy from a distrust cloud remains to be a challenging issue.

III. PROTOCOL MODEL AND DEFINITIONS

A. PROTOCOL MODEL

1) GOALS

Our general goal is to develop an efficient group key management protocol for file sharing on cloud storage, the resulting techniques should be able to confront two main problems. One is ensuring that the content of the shared files cannot be learned by the unauthorized peoples. The other is protecting the files against misoperation by the cloud provider and interception by the network.

2) SHARE MODEL

Users who want to share files constitute a sharing group, each sharing group is managed by the cloud provider. Every sharer in the sharing group owns a pair of key used to process the communication message. The public key is managed by the cloud provider, while the private key is only known by the sharers. Whenever a sharer wants to share his file within the group, it should generate a group key and encrypt the file with the group key before transmitting the file to the cloud. Then he uses a key distribution scheme to distribute the group key to the other group sharers without the participation of the cloud provider. Recovering the group key needs the collaboration of all the group members. Our share model is shown as Fig.1.

3) COMMUNICATION MODEL

To focus on the group key management, we adopt a simplified group communication model. Assuming that all file sharers use common network to broadcast message, the file sharers may broadcast a message to the other group sharers directly.

4) THREAT MODEL

Three kinds of adversary may threaten our protocol. The first is the cloud provider or passive adversary who only gathers information but does not affect the behavior of the group members in the communication. The second is the positive adversary who could alter the output information as a file sharer. The last is adaptive adversary who could compromise one or more group sharers and with the ability of gathering and alter the compromised ones' output information. Our goal is that once passive adversary or positive adversary is detected, our protocol will be terminated while the adaptive adversary has to compromise n group members to defeat our protocol, where n is the quantity of the group members.

5) ALGORITHM MODEL

Consider a sharing group G and every group mem P_i with a broadcast message B.

D is a personal key share protocol if:

(a) for any group member U_i, is determined by K and B.

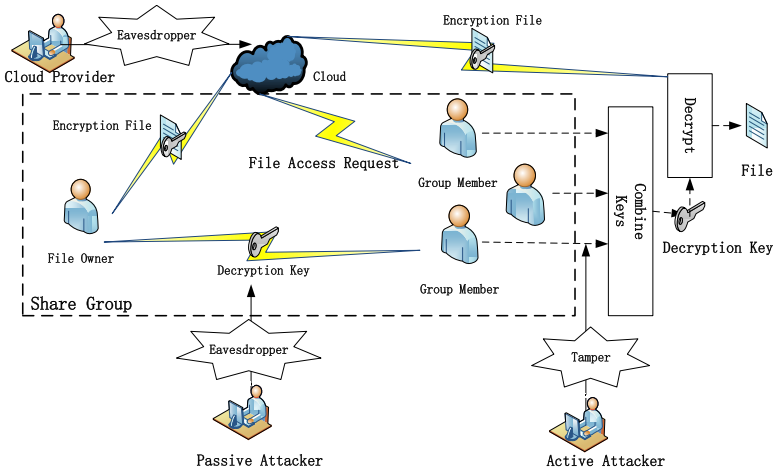


FIGURE 1. Sharing model of GKMP.

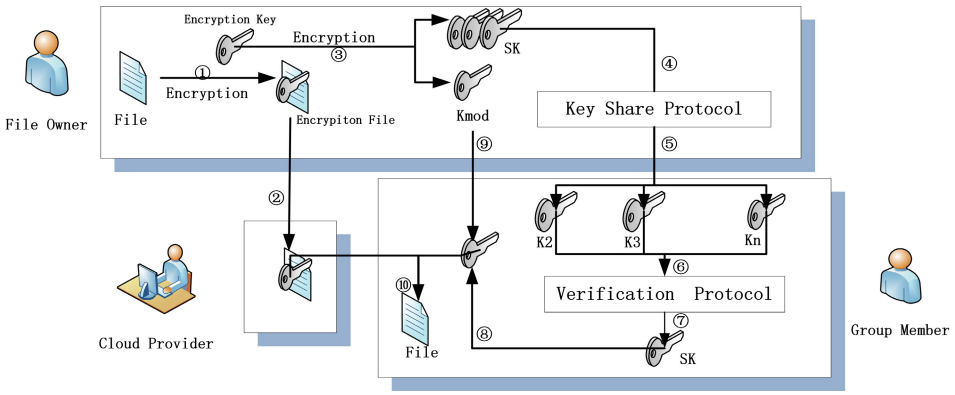


FIGURE 2. The process of GKMP.

(b) all members in the sharing group are not able to learn anything about K.

(c) no information of m_i is learned from either the broadcast message or the secret key K alone.

Definition 2: Group key management protocol guarantees equity and availability if any set $P \subset U_1, U_2, \ldots, U_n$ where the size of $P < n$, the members in P together cannot get any information about K. And after interactive operation of all the group members, K would be reconstruction.

Definition 3: Group key management protocol resists passive attack if any people $P_i \notin U_1, U_2, \ldots, U_n$ cannot get any information about K, even with the knowledge of all the interactive message.

Definition 4: Group key management protocol resists active attack if any people with the ability to tamper the output information cannot get any information about K.

IV. GKMP

In this section, we present our techniques for group key distribution. The motivation of the protocol is distribute group key without the cloud provider's participation. A key share protocol is proposed for the file owner to distribute the group keys. To detect whether there are adversaries among the key share protocol, a verification protocol is proposed as well.

The processing of GKMP is shown as Fig2.

A. KEY SHARE PROTOCOL

The purpose of key share protocol is to distribute a group key to group members, and the other members cannot get any information of the key. In our approach, the file owner broadcasts a message, and all the group members can derive the key from the message. We propose an approach with the combination of AES and RSA [22], AES is used to encrypt the shared file and RSA is used to encrypt the broadcast message. Suppose that U_1 wishes to share a file F to $U_2, U_3, \ldots U_n$. Our key distribution protocol can be shown as Fig3 and summarized as follows:

1) INITILIZATION

The cloud provider creates a sharing group G containing $U_1, U_2, \ldots, U_n$. Each U_i generates a pair of key $(P_i, S_i 0)$ and U_i sends P_i to the cloud provider via the public channel. The cloud provider transmits the public keys of the members to the file owner U_1. U_1 produces a group key K secretly and it encrypts F using Equation1. Specifically, $cipher(F)$ is sent

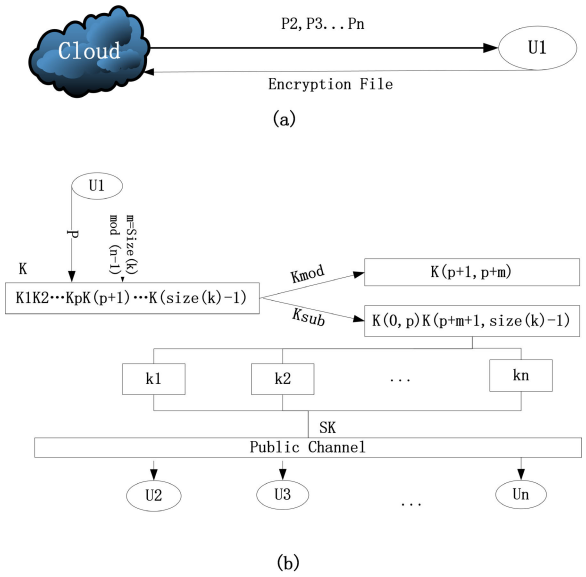


FIGURE 3. Initialization and key generation.

and stored on the cloud storage.

$$cipher(F) = ENC(AES, F, K). \tag{1}$$

2) ENCRYPITON KEY GENERATION

U_1 uses the public keys $P_2, P_3, \ldots, P_n$ of $U_2, U_3, \ldots, U_n$ which have been received from the cloud provider to generate the broadcast message SK. Firstly, U_1 calculates $m = fracsize(K)(n-1)$ and generates a random value p, taking m bits of K (record as K_{mod}) from the $(p+1)$ bit to $(p+m)$ bit secretly and splits the rest bites of K(record as K_{sub}) into $(n-1)$ piece $k_2, k_3, \ldots, k_n$ equally. Then U_i encrypts each k_i with P_i using RSA. Finally, U_i encapsulates all the encrypted k_i to SK and broadcast SK on the public channel.

$$K_{sub} = K(0, p)K(p+m+1, size(k)-1). \tag{2}$$

$$k_i = K_{sub}(\frac{size(K_{sub} \times (i-2))}{n-1}, \frac{size(K_{sub} \times (i-1)}{n-1}). \tag{3}$$

$$cipher(k_i) = ENC(RSA, k_i, P_i). \tag{4}$$

$$SK = cipher(k_2)cipher(k_3)...cipher(k_n). \tag{5}$$

step 2 to n-1) At step $j, j = 2, 3, \ldots, n$-1 U_i gets $K_{i,j-1}$ which has been received from U_{i-1} (U_2 gets it from U_n) and does the calculation steps shown as follow.

$$cipher(k_i) = ENC(RSA, k_i, P_i). \tag{6}$$

3) USER SUBSCRIPTION AND DECRYPTION

All the members of the sharing group G may get the broadcast message from the public channel which contains SK.The next task for the group members is reconstructing the group key from SK. The steps of reconstruct algorithm are shown as Fig4. .

Step 0) $U_i, i = 2...n$ gets the information from U_1 via the public channel and the encryption part $cipher(k_i)$ from SK and decrypts it using his private key S_i.

$$cipher(k_i) = SK(\frac{size(SK) \times (i-2)}{n-1}, \frac{size(SK) \times (i-1)}{n-1} - 1). \tag{7}$$

Step 1) U_i decrypts $cipher(K_i)$ using his private key S_i. Then encryption K_i with the public key of U_{i+1}. Generate $K_{i,1}$ by replacing $cipher(k_{i+1})$ with the encrypted $ENC(RSA, k_i, P_i)$ in SK and send $K_{i,1}$ to U_{i+1}(U_n transmit it to U_2).

$$k_i = DEC(RSA, cipher(K_i), S_i). \tag{8}$$

$$K_{i,1} = cipher(K_2)...cipher(k_i)ENC(RSA, k_i, P_{i+1}) \times cipher(K_{i+2})...cipher(K_n)). \tag{9}$$

step 2to n-1) At step $j, j = 2, 3, \ldots, n$-1 U_i gets $K_{i,j-1}$ which has been received from U_{i-1}(U_2 gets it from U_n) and does the calculation steps shown as follow.

$$k = \begin{cases} i-j+1 & i-j > 0 \\ n+i-j & i-j <= 0 \end{cases} \tag{10}$$

$$cipher(k_k) = K_{i,j-1}(\frac{size(SK) \times (i-2)}{n-1}, \frac{size(SK) \times (i-1)}{n-1} - 1). \tag{11}$$

$$k_k = DEC(RSA, cipher(k_k), S_k). \tag{12}$$

$$K_{i,j} = cipher(K_2)...cipher(k_i)ENC(RSA, k_k, P_{i+1}) \times cipher(K_{i+2})...cipher(K_n)). \tag{13}$$

Then U_i sends $K_{i,j}$ to U_{i+1} (U_n send it to U_2). Finally, after $n-1$ steps, every group members computes $k_i, i = 1, 2, \ldots, n$ and gets a copy of K_{mod},$K_i = k_2K_3...k_n$. The intermediate information U_i has sent and received show as table 1 and table 2.

B. VERIFICATION PROTOCOL

Key share protocol is an efficient protocol to distribute group key to group members. Here we further extend it to enable the group members to verify their own intermediate information. And the process is shown as Figure.5. During the key distribution, every group member U_i receives the information from the public channel and computes a copy of K_{sub}. Verification protocols consists of four steps in order to check K_i.

1) INITIALIZATION

Our approach chooses a one way hash function $HASH()$ to calculate the hash value of K_{sub}. U_1 broadcasts $HASH(K_{sub})$ through the public channel.

2) CALCULATE VERIFY VALUE OF K_I

Each U_i computes the hash value K_i and broadcasts their verification value V_i on the public channel.

$$V_i = ENC(RSA, HASH(K_i)) = HASH(K_{sub}.\ U_1 : 0, P_1)). \tag{14}$$

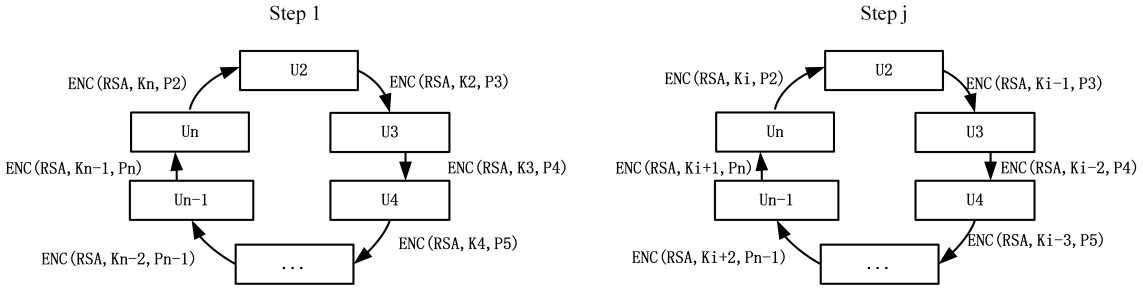


FIGURE 4. Reconstruct algorithm of GKMP.

TABLE 1. Received data by U_i.

Step	Received from U_{i-1}(U_2 from U_n)
1	$cipher(k_2)cipher(k_3)...cipher(k_n)$
2	$cipher(k_2)....cipher(k_{i-1})ENC(RSA, k_{i-1}, P_i)cipher(k_{i+1})...cipher(k_n)$
3	$cipher(k_2)....cipher(k_{i-1})ENC(RSA, k_{i-2}, P_i)cipher(k_{i+1})...cipher(k_n)$
...	...
i-1	$cipher(k_2)....cipher(k_{i-1})ENC(RSA, k_2, P_i)cipher(k_{i+1})...cipher(k_n)$
i	$cipher(k_2)....cipher(k_{i-1})ENC(RSA, k_n, P_i)cipher(k_{i+1})...cipher(k_n)$
...	...
n-1	$cipher(k_2)....cipher(k_{i-1})ENC(RSA, k_{i+1}, P_i)cipher(k_{i+1})...cipher(k_n)$

TABLE 2. Send data by U_i.

Step	Send Data By U_i
1	$cipher(k_2)....cipher(k_i)ENC(RSA, k_i, P_{i+1})cipher(k_{i+2})...cipher(k_n)$
2	$cipher(k_2)....cipher(k_i)ENC(RSA, k_{i-1}, P_{i+1})cipher(k_{i+2})...cipher(k_n)$
3	$cipher(k_2)....cipher(k_i)ENC(RSA, k_{i-2}, P_{i+1})cipher(k_{i+2})...cipher(k_n)$
...	...
i-1	$cipher(k_2)....cipher(k_i)ENC(RSA, k_2, P_{i+1})cipher(k_{i+2})...cipher(k_n)$
i	$cipher(k_2)....cipher(k_i)ENC(RSA, k_n, P_{i+1})cipher(k_{i+1})...cipher(k_n)$
...	...
n-1	Null

3) VERIFICATION

U_1 computes the summarize of V_i, $s = \sum_{i=2}^{n} V_i$ and broadcasts the result according to following steps:

1) If($S \neq n-1$), U_1 announces key distribution fails and the protocol terminates.

2) If($S = n-1$), U_1 announces that key share succeeds and it publishes K_{mod}, m.

In verification protocol, if a group member sends wrong intermediate information to our group members, it may detected by U_1.

Key share protocol is used to distribute group key to members of the sharing group without the participation of the cloud provider. Verification Protocol is used to judge whether there is any cheating exists in key share protocol and provide the security of key sharing. By executing these protocols stepwise, the group key is distributed to the group memberships secretly though public channels.

V. SECURITY ANALYSIS

In this section we address some security issues of GKMP. We start by analyzing security issue of GKMP and then giving a simple comparison between GKMP, Local Key Hierarchy (LKH) protocols presented by Wong et al and Wallner [22] and SAPDS [20] a self-healing attribute-based privacy aware date sharing in cloud.

A. SECURITY ANALYSIS OF GKMP

In the next section, we prove the security of GKMP in terms of equity, availability and resistance that are defined in Section3.2.

Theorem 1: Key share protocol is an equity secure personal key share protocol.

Proof 1: In order to prove that our protocol is an equity secure personal key share protocol, according to definition2, we need to prove that the W available participants cannot get any information about when $W < N$. Where noted the size of the sharing group is W and the available online members are N.

Attack Game1: The available members $U_k \in G_m$, $|G_m| = M$ and each U_k with the knowledge of his owner id k. They conclude to reconstruct the decryption key.

Firstly they resort their turn according their ids. Then they does the sharing protocol to reconstruct the K. As the key share protocol describes, at each turns j, U_k decrypts $K_{k-1,j-1}$ and calculates $K_{k,j}$. It sends $K_{k,j}$ to U_{k+1}, the protocol works well with the condition that $U_{k-1}, U_k \in G_m$. Conducting

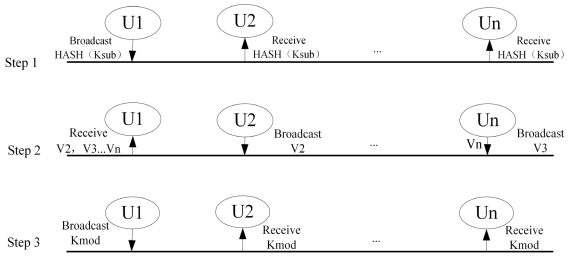


FIGURE 5. The propess of verification protocol.

M steps, every online participants gets M parts of the sub keys $k_{min}, k_{min+1,...k_m}$, where m is the minimum ID of G_m.

As there are m parts of keys is concluded, there are still $\frac{size(K)-Size(K_{mod})}{size(K)} \times \frac{n-m}{n}$ percent of the group key are secret. The probability of U_{min} guesses the key is $P1 = (2^{\frac{(size(K)-size(K_{mod}))\times(n-m)}{m}})^{-1})$.

Otherwise, the game is based on the assumption that the ids of the online users is continuous. The propobility of the assumption is $P1 = \frac{n}{C_n^m}$

The M Group Members can guess key with the probability $P = P1 \times P2 = \frac{n}{C_n^k(2^{\frac{(size(K)-size(K_{mod}))\times(n-m)}{m}})^{-1})}$.which can be ignored.

Theorem 2: GKMP is an equity secure personal key share protocol with the ability to resist passive and active attack.

Proof 2: In order to prove that our protocol has the ability to resist passive adversary. We need to show that an adversary A with the ability of gather information on the public channel can't get any information of the group key. Even if it cloud corrupt some additional users and publishes the wrong value of intermediate information can't get any information of the group key as well.

Attack Game2: Suppose that there is a passive attacker A on the public channel and A receives all the intermediate information on the public channel. The first step A takes is to decode the intermediate information with K included. As all the intermediate information is encrypted by the public keys of the group members in order, A should get the private keys of all the members. The probability of A gets all the private keys is $P1 = (\frac{1}{2^{size(SK)}})^{n-1}$).And the intermediate information is encrypted by the public keys in order, the probability of get the right group key is $(\frac{1}{2^{size(SK)}})^{n-1}) \times \frac{1}{(n-1)!}$. Thus there is more advantage than disadvantage for the attacker to crack the key.

U_i decides to corrupt some additional users U' and publishes the wrong value of intermediate information. At the end of the protocol, every uncorrupted user in $U'' = U - U'$ outputs K_j which is not equal with $HASH(K_{sub})$. The file owner would not publish K_{mod} and the protocol would be abandoned as well.

By the analysis above, we conclude that the proposed protocol achieve the security goals including equity, availability as well as resistance.

Theorem 3: GKMP is an equity secure personal key share protocol with the ability to resist cloud provider attack.

TABLE 3. Security comparison.

	LKH	SAPDS	GKMP
Access Control	own	own	own
Key Distribution Channel	Security	Security	Public
Encryption File	no	yse	yes
Group member Management	yes	yes	no
Defend Passive Attacker	yes	yes	yes
Defend Cloud Provider	no	yes	yes
Defend Active Attacker	no	no	yes

Proof 3: In order to prove that our protocol resist cloud provider, we must make sure that the shared data cloud not be decrypted by cloud provider. As proved in proof2, cloud provider couldn't get the decryption key by gathering information or correpting a group members. The shared data stored on the cloud is encrypted using AES algorithm. As the security performance of AES is excellent and unknown attack methods can attack non-linear components, we conclude that shared data could not be decrypted by cloud provider.

B. SECURITY COMPARISON

Table 3 summarized the comparison between LKH, SAPDS and GKMP. One of the major differences between GKMP, SAPDS and LKH is the roles of key manager. In the group key approach, the key manager is the cloud provider, whereas in GKMP and SAPDS, the group key is determined by the group members and shared without the participant of the cloud provider. Even more, in group key approach and SAPDS, a safety transmission channel must be exist to protect the master key from being stolen by attackers. While in GKMP, the group key is encrypted using each group members' public key and only public transmission is needed to distribute the group key.

Another important distinction among these three approaches is the security level of shared file. As the files are stored in an open environment, the security of files became more important. In LKH, master key is just used to control the access and the files are stored on the cloud without encryption. While in GKMP and SAPDS the group key is used to encrypt shared files as well. As the cloud provider just manages the encrypted shared files and public keys of group members, the shared files have become more safety than before. Obviously, GKMP and SAPDS are more suitable to the cloud storage. But SAPDS assumed that group members behave honestly, by which they mean that only passive attackers are defend.

VI. RESULTS AND EVALUATION

In this section, we provide the performance assessment of the proposed scheme. Particularly, our assessment focuses on the storage and computational overhead of GKMP.

A. PERFORMANCE

A series of experiments are designed to analysis the efficiency of GKMP. A server with Intel core 8 Duo 2.93GH processer and 8GB RAM is used to store the shared files as cloud

TABLE 4. Computing complexity.

Group Members	10	20	30	40	50	60	70	80	90	100
GKMP($\times Lbits$)	10	20	30	40	50	60	70	80	90	100
SAPDS($\times Lbits$)	17	20	40	50	61	72	82	93	102	113
Percent(%)	41.1	31.1	25	20	18	16.6	14.6	13.9	11.7	11.5

storage does. And varying numbers of threads running on a personal computer with Intel core 2 Duo 2.93GH processer and 2GB RAM as participants.

B. STORAGE OVERHEAD

In this section, the storage overhead of SAPDS and GKMP are tested. As CP-ABE [12], [14] used by the SAPDS to distribute the key, ciphertext size, public and private key size between the latest CP-ABE scheme [14] and GKMP were counted. In our paper, we assume that the quantity group numbers was n and the size of key is. The efficiency of GKMP is measured in the following item. **Ciphertext Size** Implied the communication cost that the file owner needed to send to the cloud(SAPDS) or the data owner needed to send to group members(GKMP). In SAPDS and GKMP, the shared files and encrypted key were sent by the file owner to the cloud.

$$\begin{aligned} NUM_c(SAPDS) &= NUM_c(GKMP) \\ &= size(EncryptedFile) \\ &\quad + size(EncryptedKey). \end{aligned} \tag{15}$$

Private Key Size Represented the storage cost of the group members' private keys in the scheme.

In SAPDS, every group member needs to store a pair of private-public keys and a number of access key tree's attribute, We consider the number of attributes is r, $r < log_2 n$ and the size of attributes is $L'bits$. The total size of keys stored by a group member is $2 \times L + r \times L')bits$. ($2 \times L$ bits are the size of a pair of asymmetric key and $(r \times L')bits$ stands for the size of attribute keys.

$$NUM_p(SAPDS) = (2 \times L + r \times L')bits. \tag{16}$$

In GKMP, the group members only need a pair of private-public keys.

$$NUM_p(GKMP) = (2 \times L)bits. \tag{17}$$

$r \times L'bits$ of private key size was need by SAPDS than GKMP.
Public Key Size Represented the storage cost for cloud to store all members' public keys.

In SAPDS, CP-ABE was used to distribute encryption keys, at least $2 \times Log_2 n$ access key tree attributes are stored by the cloud provider.

$$NUM_p(SAPDS) = (n \times L + 2 \times Log_2 n \times L')bits. \tag{18}$$

In GKMP, only members' public keys were stored by the cloud provider.

$$NUM_p(GKMP) = (n \times L)bits. \tag{19}$$

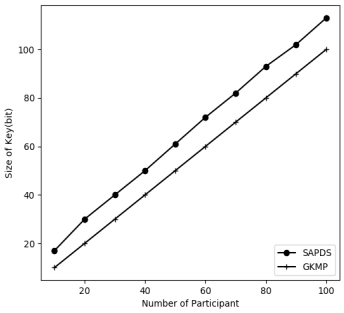


FIGURE 6. Public key size of SAPDS and GKMP.

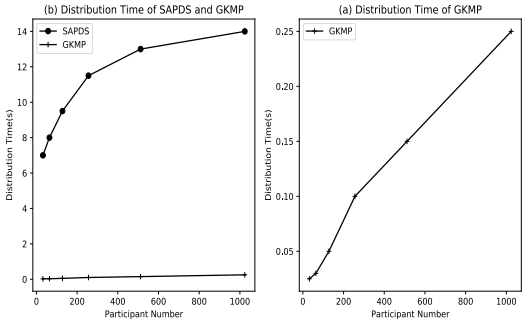


FIGURE 7. Computation overhead with different participant number of SAPDS and GKMP.

To simplify the computation, we assume the attribute have the same size of the key. With the growth of the quantity of participant members and key size, the Key size is shown as Table4. As is shown in the Fig7, about 20 percent of key size was saved by GKMP. Even with same quantity of ciphertext size, $(r \times L')bits$ of private key and 20% of public keys are saved by GKMP.

C. COMPUTATION OVERHEAD

The first step in SAPDS and GKMP is generating a secret key K to encrypt the shared files and then encryption algorithm is used to process the secret Key. The process time is tested in our experiment with the 512bit secret key and the statistics is shown in Fig7. The static shows that, the process of secret key with five different group members would take maximum 14.2s by SAPDS. However, GKMP just takes at most 190ms to process the secret key.

In SAPDS user executes CP-ABE decryption process to the secret key K. And in GKMP share is primarily required when user gets K for the very first time. Fig8 shows the computation overhead of CP-ABE and GKMP decryption process over 56-bit, 128-bit and 256-bit decryption keys.

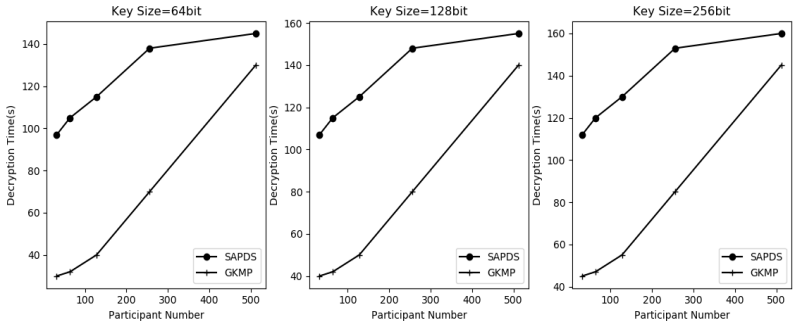


FIGURE 8. Computation overhead with different encryption key of SAPDS and GKMP.

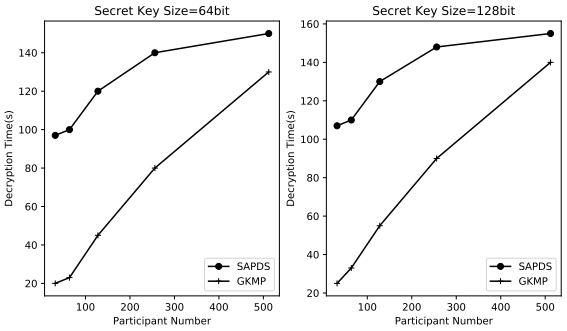


FIGURE 9. Computation overhead with different secret key of SAPDS and GKMP.

The static shows that, the process of encryption key with five different group members would take maximum 14.3s by SAPDS. However GKMP just take at most 191ms and the size of decryption keys has little influence in GKMP's computational overhead.

SAPDS and GKMP exhibits different decryption time for different sizes of secret key K, encrypted under same size of encryption key. Shown in Fig9, SAPDS tends to consume slightly more time as compared to GKMP. Furthermore, GKMP shows linear decryption overhead with the increase in number of group member.

VII. CONCLUSION

In this paper, we propose a novel group key management protocol for file sharing on cloud storage. Public key are used by GKMP to guarantee the group key distribute fairly and resist attack from compromised vehicles or the cloud provider. We give detailed analysis of possible security attacks and corresponding defense,which demonstrates that GKMP is secure under weaker assumptions. Moreover we demonstrate the ptotocol exhibits less storage and computing complexity.

REFERENCES

[1] P.-W. Chi and C.-L. Lei, "Audit–free cloud storage via deniable attribute–based encryption," *IEEE Trans. Cloud Comput.*, vol. 6, no. 2, pp. 414–427, Apr. 2018.

[2] J. Zhou, H. Duan, K. Liang, Q. Yan, F. Chen, F. R. Yu, J. Wu, and J. Chen, "Securing outsourced data in the multi–authority cloud with fine–grained access control and efficient attribute revocation," *Comput. J.*, vol. 60, no. 8, pp. 1210–1222, Aug. 2017.

[3] J. Wu, Y. Li, T. Wang, and Y. Ding, "CPDA: A confidentiality-preserving deduplication cloud storage with public cloud auditing," *IEEE Access*, vol. 7, pp. 160482–160497, 2019.

[4] H. Xiong and J. Sun, "Comments on verifiable and exculpable outsourced attribute–based encryption for access control in cloud computing," *IEEE Trans. Depend. Sec. Comput.*, vol. 14, no. 4, pp. 461–462, Jul. 2017.

[5] J. Shao, R. Lu, and X. Lin, "Fine-grained data sharing in cloud computing for mobile devices," in *Proc. IEEE Conf. Comput. Commun.(INFOCOM)*, Apr. 2015, pp. 2677–2685.

[6] R. Ahuja, S. K. Mohanty, and K. Sakurai, "A scalable attribute-set-based access control with both sharing and full-fledged delegation of access privileges in cloud computing," *Comput. Elect. Eng.*, vol. 57, pp. 241–256, Jan. 2017.

[7] S. Roy, A. K. Das, S. Chatterjee, N. Kumar, S. Chattopadhyay, and J. J. P. C. Rodrigues, "Provably secure fine–grained data access control over multiple cloud servers in mobile cloud computing based healthcare applications," *IEEE Trans. Ind. Informat.*, vol. 15, no. 1, pp. 457–468, Jan. 2019.

[8] Z. Fu, X. Sun, S. Ji, and G. Xie, "Towards efficient content-aware search over encrypted outsourced data in cloud," in *Proc. IEEE 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[9] M. Blaze, "A cryptographic file system for UNIX," in *Proc. 1st ACM Conf. Comput. Commun. Secur. (CCS)*, 1993, pp. 9–15.

[10] H. Gobioff, "Security for a high performance commodity storage subsystem," Ph.D. dissertation, School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, 1999.

[11] E. Miller, "Strong security for network-attached storage," in *Proc. Conf. File Storage Tech.*, vol. 6, 2002, pp. 1–13.

[12] Y. S. Rao, "A secure and efficient ciphertext–policy attribute–based signcryption for personal health records sharing in cloud computing," *Future Gener. Comput. Syst.*, vol. 67, pp. 133–151, Feb. 2017.

[13] J.-S. Su, D. Cao, X.-F. Wang, Y.-P. Sun, and Q.-L. Hu, "Attribute–based encryption schemes," *J. Softw.*, vol. 22, no. 6, pp. 1299–1315, Jun. 2011.

[14] J. Liu, X. Huang, and J. K. Liu, "Secure sharing of personal health records in cloud computing: Ciphertext–policy attribute–based signcryption," *Future Gener. Comput. Syst.*, vol. 52, pp. 67–76, Nov. 2015.

[15] K. Huang, R. Tso, Y.-C. Chen, S. M. M. Rahman, A. Almogren, and A. Alamri, "PKE–AET: Public key encryption with authorized equality test," *Comput. J.*, vol. 58, no. 10, pp. 2686–2697, Oct. 2015.

[16] L. Wu, Y. Zhang, K.-K.-R. Choo, and D. He, "Efficient and secure identity-based encryption scheme with equality test in cloud computing," *Future Gener. Comput. Syst.*, vol. 73, pp. 22–31, Aug. 2017.

[17] Q. Xu, C. Tan, Z. Fan, W. Zhu, Y. Xiao, and F. Cheng, "Secure multi–authority data access control scheme in cloud storage system based on attribute–based signcryption," *IEEE Access*, vol. 6, pp. 34051–34074, 2018.

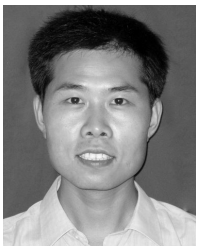
[18] H. He, R. Li, X. Dong, and Z. Zhang, "Secure, efficient and fine–grained data access control mechanism for P2P storage cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 4, pp. 471–484, Oct. 2014.

[19] K. Xue, Y. Xue, J. Hong, W. Li, H. Yue, D. S. L. Wei, and P. Hong, "RAAC: Robust and auditable access control with multiple attribute authorities for public cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 953–967, Apr. 2017.

[20] Z. Pervez, A. M. Khattak, S. Lee, and Y.-K. Lee, "SAPDS: Self-healing attribute-based privacy aware data sharing in cloud," *J. Supercomput.*, vol. 62, no. 1, pp. 431–460, Oct. 2012.
[21] N. T. Courtois and G. V. Bard, "Algebraic cryptanalysis of the data encryption standard," in *Cryptography and Coding*, vol. 4887. Berlin, Germany: Springer, 2007, pp. 152–169.
[22] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," *J. Cryptol.*, vol. 26, no. 1, pp. 80–101, Jan. 2013.

SHOUYI ZHANG was born in Linfen, Shanxi, China, in 1988. He received the M.S. degree from Beijing Jiaotong University (BJTU), China, in 2014, where he is currently pursuing the Ph.D. degree in mechanical engineering. His research interest includes manufacturing system optimization.

SI HAN was born in Suzhou, Anhui, China, in 1988. She received the Ph.D. degree from the Beijing University of Posts and Telecommunications (BUPT), China, in 2015. She is currently a Lecturer with the China University of Political Science and Law. Her research interests mainly include information security, cloud computing, and the Internet of Thing.



BAOKUN ZHENG born in Baoding, Anhui, China, in 1978. He received the M.S. degree from Renmin University of China. He is currently pursuing the Ph.D. degree with the Beijing Institute of Technology. He is also working as an Associate Professor with the Beijing Institute of Technology, China University of Political Science and Law. His research interests include security networks and blockchain.

KE HAN was born in Heze, Shangdong, China, in 1980. He received the Ph.D. degree from the Beijing University of Posts and Telecommunications (BUPT), China, in 2008. He is currently an Associate Professor with BUPT. His research interest mainly includes electronic and telecommunications.

ENTONG PANG was born in Chahaeryouyiqianqi, Inner Mongolia, China, in 1988. She received the M.E.M. degree from Peking University, China, in 2019. She is currently an Engineer with Aerospace Hongka Intelligent Technology (Beijing) Company, Ltd. Her research interests mainly include information security and industrial control security.