Defence Science Journal, Vol. 72, No. 5, September 2022, pp. 703-711, DOI : 10.14429/dsj.72.17461

Simple and Efficient Group Key Distribution Protocol using Matrices

Atul Pandey,[#,*] Indivar Gupta[$] and Dhiraj Kumar Singh[%]

[#]*Department of Mathematics, University of Delhi, Delhi - 110 007, India*
[$]*DRDO - Scientific Analysis Group (SAG), Metcalfe House, Delhi - 110 054, India*
[%]*Zakir Husain Delhi College (University of Delhi), Jawaharlal Nehru Marg, Delhi - 110 002, India*
[*]*E-mail: pandeyatul_ap@yahoo.com*

ABSTRACT

Group Key Distribution (GKD) protocols are designed to distribute a group key to several users for establishing a secure communication over a public network. The central trusted authority, called the key distribution center (KDC) is in charge of distributing the group keys. For securing the communication, all the users share a common secret key in advance with KDC. In this paper, we propose a secure and efficient Group Authenticated Key Distribution (GAKD) protocol based on the simple idea of encryption in matrix rings. In this protocol, each user registers in private with the KDC, while all the other information can be transferred publicly. The scheme also supports authentication of group keys without assuming computational hard problems such as Integer Factorization Problem (IFP).

The analysis of our GAKD protocol shows that the proposed protocol is resistant to reply, passive and impersonation attacks. Our construction leads to a secure, cost and computation- effective GAKD protocol.

Keywords: Group key distribution protocols; Matrices; Group communications

1. INTRODUCTION

The basic condition for secure group communications over public channels is that all group users should agree on a common secret key. Group Key Exchange (GKE) protocol is the most basic component of group communications where the fundamental goal is to establish a common secret key (group key) in a way that no one other than the group members can obtain the group key. The objective of group key exchange protocol with authentication is to establish a secret group key between the legitimate group members who can verify the authenticity of the shared key. This secret group key (session key) is used to facilitate secure communication services such as confidentiality, authentication, data integrity, etc.

Most of the popular group key protocols are divided into two categories: (1) Group Key Exchange (GKE) protocols: there is no explicit KDC and all communicating parties interactively determine the session keys and (2) Centralized Group Key Distribution (GKD) protocols, where a Key Distribution Center (KDC) is in charge of managing the entire group from selecting session keys to transporting these secretly to all communicating entities. The most famous key exchange protocol is Diffie–Hellman key agreement protocol[12] which can provide session keys for only two entities. Various attempts have been made for extending the 2-party Diffie-Hellman key agreement protocol to its multi-party variant.[13,1,7]

Centralized group key distribution protocols are widely used due to their efficiency in implementation. Guo,[3] *et al.*also proposed a GAKD protocol based on the generalized Chinese remainder theorem. Zheng,[16] *et al.*proposed two variations for centralized key distribution protocols named Fast Chinese Remaindering Group Key and Chinese Remaindering Group Key. Shamir's secret sharing has also been used to design group key distribution protocols.[5,9,15] For example, Harn-Lin[5] and Liu,[8] *et al.* proposed authenticated group key transfer protocols where they use the IFP to resist insider attacks. Meng, *et al*.[9] in have also proposed a GKD protocol which is based on a secret sharing scheme by Shamir but the security of their protocol does not rely on any computational hard problem. There are several research articles where the construction and analysis of group key protocols are discussed.[6,11-12]

In the protocols proposed in,[5,8-9] one-way hash functions are computed by users to authenticate the session key. The KDC publishes the hash value of the session key in advance, which is used to verify the authenticity of the group key. Recently we have also worked on cryptographic protocols which are based on matrices over rings[4,10].

On the other hand, there are some limitations of these protocols: some cryptographic algorithms assume the hardness of mathematical problems, many need a vast number of operations and there are some which cannot prevent reply attacks. Several protocols have been proposed in past years but most of these are deficient in terms of the communication overhead, computational complexity, storage complexity, and a large number of users. Thus, it is essential to design a Group Authenticated Key Distribution (GAKD) protocol, which has the ability to overpower the above weaknesses.

Received : 31 August 2021, Revised : 8 April 2022
Accepted : 7 July 2022, Online published : 1 November 2022

Our contribution: In this paper, we design a secure and efficient GAKD protocol that is based on the simple idea of symmetric encryption in matrix rings. In the proposed protocol, each user needs to register with KDC in private while all the other messages can be transferred publicly. The protocol supports authentication of group keys without assuming any hard mathematical problem. We have also proved the scheme to be secure against passive, impersonation, and reply attacks. The scheme is feasible due to its efficiency in communication and computation cost.

The rest of the article is organized in the following way: in section 2, we provide primary definitions and results for a better understanding of the protocol. Section 3 presents the structure of group authenticated key distribution (GAKD) protocol, entities, and threat models for GAKD protocols. In section 4, we construct a group authenticated key distribution protocol using the results of section 4. In section 5, we discuss its security against passive, impersonation, and reply attacks. Section 6 discusses various complexities of the proposed scheme. In section 7, we provide experimental results with the implementation of the proposed protocol. Conclusions are finally drawn in section 8.

2. PRELIMINARIES

In this section, we propose the following symmetric encryption scheme and discuss its security for any passive adversary:

2.1 Proposed Symmetric Encryption Scheme

Let C be a finite field with p (prime) elements and m, n are positive integers. Suppose Alice and Bob are two entities that share a common secret vector $r \in Z_p^m$ and A be a public matrix in $Mat_{m\times n}(Z_p)$ with $m \geq n$ and $rank(A)=n.$ For encrypting a message Z_p^n, Alice computes

$$Ax + r = b$$

and sends $b \in Z_p^m$ to Bob. Bob removes the secret part r from b and solves the system

$$Ax = b - r$$

Since $rank(A)=n$, we have by Rank-Nullity theorem that $nullity(A)=n-rank(A)=0$. Clearly, x is a solution of the above system and hence Bob solves the system to obtain the message x uniquely.

2.1.1Definition 1(Problem-A)

For a given public matrix $A \in Mat_{mxn}(Z_p)$ with m≥n, rank(A)=*n,* and a vector $b \in Z_p^m$, find a vector pair $(x,r) \in Z_p^n \times Z_p^m$ such that the following equation holds:

$$Ax + r = b \tag{1}$$

provided such x and r exists.

2.2 Brute Force Attack on Problem-A

Since the message vector x is chosen from Z_p^n and the secret vector r is selected from $Z_p^m((x,r) \in Z_p^n \times Z_p^m)$, the number of choices for x is p^n whereas the number of choices for the vector r is p^m. Hence, the exhaustive search attack results in $p^n \times p^m = p^{n+m}$ possible solutions.

2.3 A Linear Algebra Attack on Problem-A

Consider the known matrices

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

and the column vectors

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \text{and} \quad r = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix}$$

then Eqn. (1) can be rewritten as the following system of linear equations:

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n + 1.r_1 + 0.r_2 + \cdots + 0.r_m = b_1 \\ &a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n + 0.r_1 + 1.r_2 + \cdots + 0.r_m = b_2 \\ &\cdot \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \cdot \\ &\cdot \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \cdot \\ &\cdot \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \cdot \\ &a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n + 0.r_1 + 0.r_2 + \cdots + 1.r_m = b_m \end{aligned}$$

which can further be written as the following new system of m equation in $m+n$ variables,

$$\bar{A}\bar{x} = b \tag{2}$$

where,

$$\bar{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} & 1 & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & a_{2n} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} & 0 & 0 & \dots & 1 \end{bmatrix}_{m\times(m+n)} \quad \text{and} \quad \bar{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix} \tag{3}$$

Due to the linear independence of the rightmost *m* columns of the matrix $\bar{A}$, we have that $rank(\bar{A}) \geq m$ and since $\bar{A}$ is $m\times(m+n)$ matrix, we have that $rank(\bar{A}) \leq m$. Thus, we conclude that $rank(A)=m.$ Hence by Rank-Nullity theorem,

$$nullity(\bar{A}) = m + n - rank(\bar{A}) = n$$

Using an algorithm such as Gaussian elimination, the system in Eqn. (2) can be solved in polynomial time but the number of solutions to the system is p^n (since $nullity(\bar{A})=n$). From these p^n solutions, say from $\bar{x}$ the corresponding solutions (x,r) can be retrieved.

For an initially fixed value of *r*, there will be a unique value of *x* such that *(x,r)* is a unique solution of the system in (1), whereas the total number of possible solution *(x,r)* for the system is p^n. This means that users having *r* can solve Problem-A uniquely whereas any adversary will have p^n possible number of solutions for Problem-A with only one solution being the correct one. We summarize these details in Theorem 5.

2.4 Another Attack on Problem-A

Suppose we want to solve Eqn. (1) for x and r, we begin by fixing a value of r and then solve the system

$$Ax = b - r$$

which has either no solution or unique if exists (since $nullity(A) = 0$). Since $r \in Z_p^m$, we would have to try this method p^m times for every fixed value of r and we know from subsection 2.3 that there will be exactly p^n values of r for which we will be finding the unique corresponding value of x.

Here, instead of choosing the value of r we can also choose a value of *x* and obtain corresponding values of *r*. Since $x \in Z_p^n$, we obtain p^n number of solutions for Eqn. (1) exactly as the method of subsection 2.3.

Based on the above discussion and different types of attacks, we conclude that for sufficient values, say $p \cong 2^{20}, n = 15$ and $m = 20$, it is not feasible for any adversary to find the 'correct' solution to Problem-A as there are 2^{300} possible solutions for it.

2.5 Impersonating Property of Proposed Symmetric Encryption Scheme

Theorem 1. Suppose an adversary wants to send some arbitrary message *z* to Bob by impersonating as Alice. It computes $Az + r = b$ by choosing some arbitrary value of $\bar{r} \in Z_p^m$. Let $C = (b, auth)$ be the ciphertext with the authentication information of the message z . In order for Bob to correctly obtain the message z, it should satisfy

$$r = \bar{r}$$

Proof: Since *auth* information is associated with message *z*, in order for Bob to decrypt the message correctly, *z* should be a solution of the following system

$$Az = b - r$$

as *r* is the secret key shared between Alice and Bob. From the generation of ciphertext C, we also have $Az + \bar{r} = b$. Thus $b - r + \bar{r} = b$ and hence

$$r = \bar{r}\ .$$

3. GAKD PROTOCOL, ENTITIES AND ATTACK MODELS

In this section, we define the structure of GAKD protocol, types of entities, and attack models for these protocols.

3.1 Group Authenticated Key Distribution (GAKD) Protocol

A GAKD protocol can be described as a 5-tuple

$$\Pi = \left(KDC, U, A_{KDC}, A_{U_i}, \mathbb{A}\right)$$

where KDC is an entity trusted by all the users U_i of the set U. Algorithm generates the group key distribution message, whereas the algorithm A_{U_i} recovers the group key for user U_i from the group key distribution message. Algorithm A authenticates that the retrieved key was indeed generated by KDC.

3.2 Entities

In a GKD protocol, entities are divided into the following categories:

KDC: KDC is the central trusted party and it is in charge of issuing session keys to the participating users. During registration, it certifies users' identities and shares with each user a 'secret' of some kind. On receiving a group key initialization message from users, KDC sends group key distribution messages to users.

User: A user U_i who shares a 'secret' with KDC while registering. Users are required to maintain the secrecy of this 'secret' because it is used for future communications. When a set of users need a group key, they submit a request to KDC. After receiving the corresponding distribution message, the corresponding user retrieves a session key and verifies its authenticity by using algorithm $\mathbb{A}$.

Adversary: An adversary is defined as an entity that wants to attack the protocol in some way. The adversaries are further classified as

Insider: A legal user who attempts to derive 'secret' shared by other users of *U* with KDC.

Outsider: Any adversary not in *U,* who wants to attack the protocol. The goal of outsiders is to either obtain a session key or prevent users in *U* from obtaining a valid session key.

3.3 Models of Attacks

We consider the following three attack models for adversaries:

3.3.1 Passive Attack

In passive attacks, the goal of an adversary is to break its confidentiality by observing the transcript of GAKD protocol.

3.3.2 Impersonation Attack

Impersonation attacks are those where any entity tries to impersonate to be a legal user/KDC to attack a protocol. In our protocol, these are specified as

Imp I: Outsider sends group key requests to KDC by impersonating as a legal user.

Imp II: Adversary impersonates to be KDC to distribute group keys.

3.3.3 Reply attack

In this case, an entity resends outdated messages to others to attack the system in the following ways:

Rep I: Outsider resends an outdated group key request to KDC.

Rep II: Any adversary redistributes an outdated group key to users.

3.4 Meng, *et al.'s* GAKD Protocol

The group key distribution protocol proposed by Meng, *et al.*[7] consists of the following phases:

3.4.1 Preparatory Phase

1. KDC initialization: KDC selects a random prime p and a hash function $h(\cdot)$. Both $h(\cdot)$ and p are publicly known parameters.
2. User's registration: Each user registers with KDC for joining the group. In this process, each user U_i shares a private coordinate (x_i,y_i) with KDC, where x_i and y_i are in the finite field Z_p. KDC should make sure that each $x_i \neq 0$ and $x_i \neq x_j$ for $i \neq j$. Every user makes its identity U_i public while keeping it's coordinate (x_i, y_i) secret.

3.4.2 Distribution Phase

1. Let m be the total number of legal users who have registered with KDC. They constitute the group $U=\{U_1,U_2,\ldots,U_m\}$ and their private coordinates form a set $\Omega=\{(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)\}$.
2. KDC randomly generates a polynomial of degree m, $f(x)=a_0+a_1x+\cdots+a_mx^m$ and chooses a_0 as the group key k, that is, $k=f(0)=a_0$.
3. KDC picks $2m$ different coordinates on $f(x)$ to form two more sets, namely $\Omega_1=\{(x_1',y_1'),(x_2',y_2'),\ldots,(x_m',y_m')\}$ and $\Omega_2=\{(\overline{x}_1,\overline{y}_1),(\overline{x}_2,\overline{y}_2),\ldots(\overline{x}_m,\overline{y}_m)\}$ such that $\Omega\cap\Omega_1=\Omega_1\cap\Omega_2=\phi$.
4. KDC uses $x_i(i=1,2,\ldots,m)$ in Ω_1 and all the coordinates in Ω_2 to compute group key information given by

$$d_i'=\sum_{i=1}^{m}\overline{y}_t\frac{-x_i'}{x_t-x_i'}\prod_{j=1,j\neq t}^{m}\frac{-\overline{x}_j}{\overline{x}_t-\overline{x}_j}\bmod p$$

5. KDC computes the hash values $h(x_i',y_i')$ to generate values, $d_i=d_i'+h(x_i',y_i')mod\ p$ which is protected group key distribution information.
6. The initiator sends a group key initialization message I to KDC.
7. On receiving the initialization message, KDC broadcasts a response message $R_\sigma=\{\sigma,\overline{x}_1,\overline{x}_2,\ldots,\overline{x}_m\}$, where σ is a group communication identifier selected by KDC.
8. Each user U_i randomly selects a coordinate (x_i^*,y_i^*), where $x_i^*\neq x_i$. It sends to KDC the following request message $M_{\acute{o},i}$ given by

$$M_{\sigma,i}=\{\sigma,U_i,(x_i^*,y_i^*)\}$$

9. KDC already shares a private coordinate (x_i,y_i) with every user U_i and receives (x_i^*,y_i^*) in message $M_{\sigma,i}$. It uses these coordinates to form the linear function

$$g_i(x)=y_i\frac{x-x_i^*}{x_i-x_i^*}+y_i^*\frac{x-x_i}{x_i^*-x_i}mod\ p$$

10. KDC uses (x_i',y_i') to compute the values $g_i(x_i')$ and $g_i^{-1}(y_i')$ where $g_i^{-1}(y)$ is the inverse of $g_i(x)$.
11. KDC generates key distribution message $K_i=\{U_i,g_i(x_i'),g_i^{-1}(y_i'),d_i,h(k,\sigma)\}$ and sends it to the corresponding user U_i, where $h(k,\sigma)$ is authentication information about the group key k.

3.5 Key Recovery-Authentication Phase

1. Every user U_i forms $g_i(x)$ and $g_i^{-1}(y_i')$ using coordinates (x_i,y_i) and (x_i^*,y_i^*) After receiving key distribution message Ki from KDC, user Ui recovers $x_i'=g_i^{-1}(g_i(x_i'))$ and $y_i'=g_i^{-1}(g_i(y_i'))$.
2. Every user U_i uses the coordinate (x_i',y_i'), m public values $\overline{x}_1,\overline{x}_2,\ldots,\overline{x}_m$ in message R_σ, to calculate the Lagrange components given by

$$\Delta_i=y_i'\prod_{j=1}^{m}\frac{-\overline{x}_j}{x_i'-\overline{x}_j}\bmod p$$

3. The group key can be obtained as follows

$$k_i=d_i+\Delta_i-h(x_i',y_i')\bmod p$$

4. U_i uses the hash function h(·) to compute the hash values

$$h_i=h(k_i,\sigma)$$

if $h_i=h(k_i,\sigma)$ holds, the group key is correctly sent by KDC, that is, $k_i=k$. Otherwise, users should make a new group key request to KDC.

4. GROUP AUTHENTICATED KEY DISTRIBUTION PROTOCOL

In this section, we present our GAKD protocol with a detailed explanation. It is described as the 5-tuple

$$\Pi=(KDC,U,A_{KDC},A_{U_i},\mathbb{A})$$

which consists of the following phases: (1) KDC initialization phase, (2) Distribution phase, (3) Key recovery phase and (4) Authentication of the group key.

4.1 Initialization of KDC

The Key Distribution Center (KDC) selects two one-way hash functions $h_1:Z^m\mapsto Z_p^m, h_2(\cdot)$, a random prime p, and a random matrix $A\in Mat_{m\times n}(Z_p)$ with $m\geq n$ and $rank(A)=n$.

4.1.1User's Registration

Each user registers with KDC for joining the group. During the registration, KDC selects distinct random vectors

$r_i \in Z^m$ for the respective users $U_i, 1 \le i \le k$. Each user makes its identity U_i public while keeping its vector r_i private.

4.1.2 Definition 2 (No of group key request i_j by user U_i)

For each i, i_j represents j^{th} request $(1 \le j \le m)$ for group key by user U_i and $r_i \oplus i_j$ is defined as

$$r_i \oplus i_j = \begin{bmatrix} r_{i1} \\ r_{i2} \\ \vdots \\ r_{ij} \oplus j \\ \vdots \\ r_{im} \end{bmatrix}$$

During the key distribution process, KDC keeps a counter C_i corresponding to every user U_i, which keeps the record of the number of requests for group keys by user U_i .

Table 1. List of notations

Symbols	Description
Z	Set of integers
Z^m	m-tuples over Z
Z_p	Finite field with p elements
$Mat_{m\times n}(Z_p)$	Set of $m \times n$ matrices over Z_p
$h_1(\cdot), h_2(\cdot)$	Hash functions
U_i	i^{th} user, $1 \le i \le k$
$U = \{U_1, U_2, \cdots, U_k\}$	Set of legal users
r_i	'Secret' shared between user U_i and KDC
M	Group key initialization message
Res	Response message by KDC
M_i	Group key request message by user U_i
$t_i \in (Z_p^*)^n$	Random vectors with non-zero entries from Z_p
i_j	j^{th} group key request by user U_i
$x \in Z_p^n$	Group key selected by KDC
$auth$	Authentication information of group key x
D_i	Group key distribution message of user U_i

4.2 Distribution Phase (A_{KDC})

1. The initiator transmits key initiation message M to KDC.
2. KDC broadcasts the response $Res = \{A\}$ to all the users.
3. User U_i randomly picks vector $t_i \in (Z_p^*)^n$. U_i sends KDC its group key request message M_i as
$$M_i = \{t_i, U_i\}$$
4. KDC randomly selects a group key $x \in Z_p^n$ and computes the hadmard product $t_i x$ of vectors t_i and x, that is, the entry-wise product of vectors.
5. KDC then computes
$$A(t_i x) + h_1(r_i \oplus i_j) = c_i + h_1(r_i \oplus i_j) = b_i (1 \le j \le m),$$
the authentication information $auth = h_2(x, r_1, r_2, \ldots, r_k, U_1, U_2, \ldots, U_k)$ and sends the distribution message $D_i = \{auth, b_i, U_i\}$ to the corresponding U_i , where $h_2(\cdot)$ is a one way hash function.

4.3 Key Recovery Phase A_{U_i}

1. Each user uses its private vector r_j with the hash function h_i and its number of requests to obtain their corresponding value of $A(t_i x) = c_i$ which is now a system of equations with known matrix A and the vectors c_i.
2. User U_i solves the system of linear equations
$$A(t_i x) = c_i$$
and obtains the solution $t_i x$ uniquely (since $nullity(A) = 0$). Each user computes the common group key x using their random vectors t_i with invertible entries.

4.4 Authentication of the group key ($\mathbb{A}$)

Each user verifies the authenticity of group key x using auth, their respective random vectors t_i, and the public hash function h_2.

4.5 Correctness analysis

Theorem 2. All the legal user $U_i (1 \le i \le k)$, compute the common group key x. Each user is certain that the group key x is indeed sent by the KDC if $auth = h_2(x, t_i, U_i)$ holds.

Proof: On receiving the group key distribution message $D_i = \{auth, b_i, U_i\}$ from KDC user U_i, solves the following system of equations for variable z

$$Az = b_i - h_1(r_i \oplus i_j) = b_i' \quad (4)$$

Clearly, $t_i x$ is a solution of the above equation and since $nullity(A) = 0$, user U_i solves the system to get the unique value $z_i = t_i x$. Since $t_i \in (Z_p^*)^n$, user U_i retrieves the group key x as

$$t_i^{-1} z_i = \begin{bmatrix} t_{i1}^{-1} z_{i1} \\ t_{i2}^{-1} z_{i2} \\ \vdots \\ t_{in}^{-1} z_{in} \end{bmatrix} = \begin{bmatrix} t_{i1}^{-1} t_{i1} x_1 \\ t_{i2}^{-1} t_{i2} x_2 \\ \vdots \\ t_{in}^{-1} t_{in} x_n \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x$$

Since each user recovers the same group key x, they authenticate the group key x by checking if $auth = h_2(x, t_i, U_i)$ holds.

5. SECURITY ANALYSIS OF THE GAKD PROTOCOL

In this section, we examine the solution set of Problem-A. We then discuss the security of our proposed protocol against various types of attacks.

5.1 Security of Problem-A

Theorem 3. Let $A \in Mat_{m\times n}\left(Z_p\right)$ be a public matrix with $m \geq n, rank\left(A\right)=n$ and a public vector $b \in Z_p^m$. Suppose someone wants to solve the following problem for $\left(x,r\right)$

$$Ax + r = b \tag{5}$$

provided such *x* and *r* exists. Then, the probability of finding the correct solution of Eqn. (5), for an initial fixed value of *r* is $\frac{1}{p^n}$.

Proof: The following equation $Ax + r = b$ has variables $\left(x,r\right) \in Z_p^n \times Z_p^m$ and according to the method of section 2.3, Eqn. (4) has exactly p^n number of solutions. But for an initial fixed value of *r*, we get exactly one x such that $\left(x,r\right)$ is a unique solution of Eqn. (5). Thus, the probability of finding the correct solution to Eqn. (5) is $\frac{1}{p^n}$.

5.2 Resistance to Passive Attack

Theorem 4. On seeing the transcript of GAKD protocol, no outsiders can retrieve the group key and no adversary can obtain the 'secret' shared by other users with KDC.

Proof: Let *trans(Π)* denote the transcript of the protocol Π, that is,

$$trans\left(\Pi\right)=\left\{A,t_1,t_2,\ldots,t_k,b_1,b_2,\ldots,b_k,i_j\left(1\leq i\leq k,1\leq j\leq m\right),h_1,h_2\right\}$$

where we have,

$$A\left(t_i x\right)+h_1\left(r_i \oplus i_j\right)=b_i$$

Suppose any outsider wants to obtain the group key x by accessing *trans(Π)* only, it will have to solve the following problem for y

$$Ay+h_1\left(r_i \oplus i_j\right)=b_i$$

Although the outsider knows i_j, it does not have the secret vector r_i and since h_1 is a hash function, it has no information about $h_1\left(r_i \oplus i_j\right)$. Hence $h_1\left(r_i \oplus i_j\right)$ is also a variable for any outsider and it is required to solve the following problem

$$Ay+\overline{r_i}=b_i \tag{6}$$

for two-variable vectors y and $\overline{r_i}$. By virtue of Theorem 3, the probability for any outsider to obtain the group key x is $\frac{1}{p^n}$, which is negligible for sufficient values of p and n.

For deriving the 'secret' shared by other users, any outsider will have to solve Problem-A which will provide p^n values of $h_1\left(r_i \oplus i_j\right)$ but preimage resistance does not allow to obtain the value $r_i \oplus i_j$. For deriving the 'secret' of some user U_l, any insider will be able to compute the value of $h_1\left(r_l \oplus l_j\right)=b_l - A\left(t_l x\right)$.

But again, the pre-image resistance of hash function implies that no insider can derive the 'secret' of other users by obtaining the transcript of the protocol.

Hence, no outsider can retrieve the group key and no adversary can obtain the 'secret' shared by other users with KDC by observing trans (Π).

5.3 Resistance to Impersonation Attack

Theorem 5. During the execution of GAKD protocol, no outsiders can retrieve the group key or prevent other users from deriving group key by Imp I.

Proof: Suppose, some outsider $\mathbb{E}$ pretends to be a legal user U_1 and sends a request message to KDC. We have already proved in Theorem 4 that no outsider can obtain the group key with non-negligible probability.

Meanwhile, all other legal users U_i can still recover the group key by using its 'secret', the number of request, hash function and the key distribution message $M_i=\left\{A,auth,d_i,U_i\right\}$ as proved in Theorem 2. Thus, legal users are not influenced by $\mathbb{E}'s$ bogus request message and outsiders cannot stop the users from computing the session key by impersonation attack Imp I.

Theorem 6. Any adversary, whether insider or outsider, cannot circulate a group key to users by the impersonation attack Imp II.

Proof: Suppose some outsider wants to distribute group keys to all the users by impersonating as KDC .On receiving the challenges t_i from users U_i, an outsider can select a random group key z and compute *A(tiz)*. In this case, the outsider has no information about the hash values of any user, that is, $h_1\left(r_i \oplus i_j\right)$. So, outsider can choose arbitrary value $\overline{r_i}$ compute $A\left(t_i z\right)+\overline{r_i}=d_i$ and sends the key distribution message $\overline{M}_i=\left\{A,auth,d_i,U_i\right\}$ to corresponding user U_i. By Theorem 1, in order for the users to correctly retrieve the group key, it should satisfy $\overline{r_i}=h_1\left(r_i \oplus i_j\right)$ for all i because auth is authentication information associated with key z only. But since h_1 is a hash function and r_i is unknown, choosing such $\overline{r_i}$ is not possible. Similarly, if an insider wants to distribute a group key to user U_i, it should know their hash values $h_1\left(r_i \oplus i_j\right)$, which is not possible.

Thus, any adversary, whether insider or outsider, cannot circulate a group key to users by the impersonation attack Imp II.

5.4 Resistance to Reply Attack

Theorem 7. No outsider can retrieve an old/outdated group key by reply attack Rep I.

Proof: During the distribution phase, the group key x is selected randomly and has nothing to do with the random vectors t_i. This means that even if some outsider sends an outdated group key request to KDC, it will choose random

group key z and follow the steps of the distribution phase. And again, by Theorem 1, no outsider can obtain the group key z by observing the transcript of GAKD protocol.

Hence, no outsider can retrieve an outdated group key by reply attack Rep I.

Theorem 8: No adversary can circulate an old group key to legal users by reply attack Rep II.

Proof: Suppose an adversary obtains an outdated group key x. It has all the previous information associated with x such as group key request message $M_i = \{t_i, U_i\}$, group key distribution message $D_i = \{auth, b_i, U_i\}$ .When users request for new group key using fresh values of vector $t_i^{'}$, adversary sends outdated group key distribution message $D_i = \{auth, b_i, U_i\}$ where, $b_i = A(t_i x) + h_1(r_i \oplus i_j)$. Since the number of group key requests changes, while recovering the group key x, users will use hash values $h_1(r_i \oplus i_{j+1})$ which would be completely different from $h_1(r_i \oplus i_j)$. As a result, users will solve the following system

$$Az = b_i - h_1(r_i \oplus i_{j+1})$$

which may or may not have a solution. Even if it has a solution, say z, it is not necessary that this z would be satisfying $z = t_i^{'}x$ ($t_i^{'}$ constitute a new group key request message). Hence users do not retrieve the old group key x. Thus, no adversary can disburse an old group key to the users by Rep II.

6. EXPERIMENT

In this section, we implement our GAKD protocol and provide experimental results for the different numbers of users. To support the claim of efficiency of our proposed GAKD protocol, we implemented the protocol with SageMath. We executed the protocol for the massive number of users to collect data about the time taken in responding to group key requests and the time taken in recovering group keys.

Table 2 shows the specification of the computer used for executing the protocol using SageMath.

Table 2. System specifications

Processor	**Intel(R) Core (TM) i3-5005U CPU @ 2.00 GHz**
Operating system	Windows 10 pro, bit
RAM	4 GB
Programming language	SageMath

Here we have computed response time of KDC and key recovery time of a user for a single group key request. For the simulation of hash function, we have used random tuples from Z^m. The computation time of hashing and key authentication is neglected.

It is worth noting that in Table 3, the key recovery time is almost the same even if there are a different number of users in the group. This is due to the fact that key recovery is independent of the number of users present in the group and it is required to solve 20 equations in 15 variables over Z_p by all the users for all the cases.

Table 3. Computation time for entities in seconds

Number of users	**Response time of KDC**	**Key recovery time of single user**
50	0.018233	0.0016902
100	0.033114	0.0017551
200	0.071087	0.0016338
300	0.097746	0.0017149
400	0.130984	0.0017013
500	0.163447	0.0016374
600	0.200384	0.0016661
700	0.232993	0.0018009
800	0.266245	0.0016548
900	0.306462	0.0016618
1000	0.338680	0.0016951

7. PERFORMANCE ANALYSIS AND COMPARISON

We compare our protocol with a few existing protocols on various aspects such as computational complexity of KDC, computational complexity of single user, communication overhead, storage complexity, etc.

7.1 The Computation Complexity of KDC

Let k be the number of legal users. In our protocol, KDC performs k hadmard product multiplications, k vector additions, k matrix multiplication and $k+1$ hash operations. Thus, the total number of field operations required for KDC to distribute the keys are $kmn + kn$ multiplicative operations, $km + km(n-1)$ additions, and $k+1$ hash operations. For fixed values of $m = 20$ and $n = 15$, the complexity of KDC is $O((\log_2 p)^2)$ since multiplication has quadratic complexity $O((\log_2 p)^2)$ in Z_p.

7.2 The Computation Complexity of Users

After receiving the distribution message from KDC, a user starts to derive and verify the authenticity of the group key. Each user needs 2 hash operations, *m* matrix subtractions, m^3 field operations for Gaussian elimination, and n field inversions. Thus, the complexity for recovering the group key is $O(1)$ field multiplications (for a fixed size matrix A) and is independent of the number of users in the group.

7.3 Communication Overhead

Users send about kn numbers to KDC and KDC transfers $mn + km$ numbers to the users. Thus the total communication overhead is $kn + km + mn$ numbers of Z_p. For our parameters, that is, $p \approx 2^{20}, m = 20$ and $n = 15$, the total communication overhead is $(300 + 35k)\log_2(2^{20}) = 700k + 6000$ -bits.

7.4 Storage Complexity

In our protocol, KDC must store secret vectors r_i of users and matrix A which means that it must store $km + mn$ numbers whereas each user needs m numbers to store its secret vector r_i and mn numbers to store the matrix A where each of these numbers is in Z_p. Thus, the storage complexity is $O(k\log_2 p)$ since each number in needs $\log_2 p$ bits to store its value.

7.5 Comparative Analysis

Let k denotes the number of legal users and Z_n be the platform for all the protocols. For Harn-Lin's protocol, n is a 1024 -bit RSA modulus, for Meng *et al's* protocol, n is a 260 -bit prime, for our protocol n is a 20 -bit prime and the size of the matrix A is 20×15.

Table 4 compares our protocol with the existing protocols on many aspects such as hard problem assumption, resistance to passive, impersonation and reply attacks, etc.

Analysis of Table 5 shows that for our proposed protocol, the computational complexity for KDC is linear in the number of users whereas the existing protocols have quadratic complexity. The computational complexity for a single user is also independent of the number of users present in the system whereas it is linear/quadratic for existing protocols.

The computation time of hashing is $O(1)$, which is constant and hence it is neglected.

Analysis of Table 6 shows that our protocol performs better than Meng et al's protocol for the time taken to respond by KDC and for recovering the key by a user. Meng et al's protocol is implemented on a more powerful system than ours. If we use a more advanced computer for implementing our protocol, we definitely will get much better results than Meng et al's protocol.

8. CONCLUSION

We have proposed a GAKD protocol using the simple idea of encryption in matrix rings. We have proved that the scheme is secure against passive, impersonation, and reply attacks. We have obtained the computational complexity for the proposed protocol and experimental results for a different number of users to validate our claim of the efficiency of the protocol.

ACKNOWLEDGMENT

The research of the first author is supported by University Grants Commission (UGC), reference number-1100(DEC-2016). The third author is grateful for the support from the SERB-MATRICS scheme (MTR/2020/000508) of the Department of Science and Technology, Government of India.

Table 4. Comparison of various GAKD protocols

Properties	Harn-Lin's protocol	Meng et al's protocol	Liu et al's protocol	Our protocol
Assumption of hard problem	Yes	No	Yes	No
Number of hash functions	1	1	2	2
Resistant to passive attacks	Yes	Yes	Yes	Yes
Resistant to impersonation attacks	Yes	Yes	Yes	Yes
Resistant to reply attacks	No	Yes	Yes	Yes

Table 5. Complexity comparison of GAKD protocols

Complexity	Harn-Lin's protocol	Meng et al's protocol	Our protocol
Computational Complexity of KDC	$O\left(k^2\left(\log_2 n\right)^2\right)$	$O\left(k^2\left(\log_2 n\right)^2\right)$	$O\left(k\left(\log_2 n\right)^2\right)$
Computational Complexity of single user	$O\left(k^2\left(\log_2 n\right)^2\right)$	$O\left(k\left(\log_2 n\right)^2\right)$	$O\left(\left(\log_2 n\right)^2\right)$
Communication Overhead bits	$5120k$ **-bits**	$2340k$ **-bits**	$700k+6000$ **- bits**
Storage space of KDC	$2klog_2n=2048k$ **-bits**	$6klog_2n=1560k$ **-bits**	$(20k+300)log_2n=40k+6000$ **-bits**
Storage space of single user	$2log_2n=2048$ **-bits**	$2log_2n=520$ **-bits**	$320log_2n=6400$ **bits**

Table 6. Comparison of KDC response time and key recovery time

Number of users	KDC response time		Key recovery time	
	Meng's protocol	Our protocol	Meng's protocol	Our protocol
50	0.104552	0.018233	0.019082	0.0016902
100	0.134091	0.033114	0.037025	0.0017551
200	0.328321	0.071087	0.091200	0.0016338
300	0.633199	0.097746	0.160992	0.0017149
400	1.047877	0.130984	0.245661	0.0017013
500	1.295323	0.163447	0.315709	0.0016374

REFERENCES

1. Bresson, E.; Chevassut, O.; Pointcheval, D.: Provably secure authenticated group Diffie–Hellman key exchange. *ACM Trans. Inf. Syst. Secur. (TISSEC),* **10**(3), 10 (2007).
2. Diffie, W. & Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theory*, **22**(6), 644-654 (1976).
3. Guo, C. & Chang, C.C. An authenticated group key distribution protocol based on the generalized Chinese remainder theorem. I*nt. J. Commun. Syst.*, 2014, **27**(1), 126–134.
4. Gupta, I.; Pandey, A. & Dubey, M.K. A key exchange protocol using matrices over group ring, Asian-European *J. Math.*, 2019.
5. Harn L. & Lin C. Authenticated group key transfer protocol based on secret sharing. *IEEE Trans. Comput.* **59**(6), 842–846 (2010).
6. Jaiswal, P.; Tripathi, S. Cryptanalysis of olimid's group key transfer protocol based on secret sharing. *J. Inf. Optim. Sci.,* 2018, **39**(5), 1129-1137. doi: 10.1080/02522667.2017.1292655
7. Kim, Y.; Perrig, A. & Tsudik, G. Tree-based group key agreement. *ACM Trans. Inf. Syst. Secur. (TISSEC)*, 2004, **7**(1), 60–96.
8. Liu, Y.; Cheng, C. & Cao, J. An improved authenticated group key transfer protocol based on secret sharing. *IEEE Trans. Comput.,* 2013, **62**(11), 2335–2336.
9. Meng, K.; Miao, F. & Yu, Y. A secure and efficient on-line/off-line group key distribution protocol. *Des. Codes Cryptogr.*, 2019, **87**(7), 1601–1620.
10. Pandey, A.; Gupta, I.; Singh, D.K. On the security of DLCSP over $GL_n(F_q[Sr])$, Applicable algebra in engineering, communication and computing, 2021.
11. Santhanalakshmi, S.; Sangeeta, K. & Patra, G.K. Design of group key agreement protocol using neural key synchronization. *J. Interdiscip. Math.*, 2020, **23**(2), 435- doi: 10.1080/09720502.2020.1731956
12. Srivastava, G.; Singh, J.N. & Manjul, M. Group key management: Issues and opportunities. *J. Discrete Math. Sci. & Cryptography*, 2021, **24**(3), 787-795. doi: 10.1080/09720529.2020.1794518
13. Steiner, M.; Tsudik, G. & Waidner, M. Diffie–Hellman key distribution extended to group communication. In Proceedings of the 3rd ACM Conference on Computer and Communications Security, pp. 31–37. ACM, New York (1996).
14. Storjohann, A. & Mulders, T. Fast algorithms for linear algebra modulo N. Proceedings of Algorithms—ESA'98. Springer Berlin Heidelberg, 1461, 139-150 (1998).
15. Sun, Y.; Wen, Q. & Sun, H. An authenticated group key transfer protocol based on secret sharing. *Procedia Eng.* 2012, **29**, 403–408.
16. Zheng, X.; Huang, C.T. & Matthews, M. Chinese remainder theorem-based group key management. *In* Proceedings of the 45th annual southeast regional conference (ACM-SE 45). ACM, New York, NY, USA, 266-271 (2007).

CONTRIBUTORS

Mr Atul Pandey has completed his BSc(H), Mathematics, MSc(Mathematics) from University of Delhi, India. He has recently submitted his PhD thesis in Mathematics from University of Delhi. He has qualified NET, JRF and GATE in Mathematics. He is currently working as an Assistant Professor at Galgotias University, Greater Noida.
For this work, he has defined and discussed the security of Problem-A and used it for the construction of the proposed GAKD protocol.

Dr Indivar Gupta completed his PhD from IIT Delhi, India. He has been working as a scientist in Scientific Analysis Group, DRDO since 2000, and has research contributions in various areas related to cryptology and information security. His area of research interests includes computational algebra, number theory and high performance computing.
For this study, he has guided with the analysis and experimental results using the SageMath software.

Dr Dhiraj Kumar Singh is Associate Professor in the Department of Mathematics, Zakir Husain College, Delhi (University of Delhi). He has an experience of 10 years in teaching and research. His area of research is Information Theory and Cryptology. He has published more than 30 research papers in journals of international repute. He is MATRIX Fellow of Department of Science and Technology, Govt. of India.
In the present work, he provided the research inputs for analysis, modification and revision of the manuscript.