Authenticated Group Diffie Hellman Key Exchange for Secure Group Communication

Lavanya R[a] , S V Sathyanarayana[b]
[a] *M.Tech Student, DECS, JNNCE, Shimoga, Karnataka, India*
email:lavanyarn.nayak@gmail.com
[b] *Professor, Department of ECE, JNNCE, Shimoga, Karnataka, India*
email:svs@jnnce.ac.in

Abstract. Security is a major industry concern that could significantly slow IoT market growth. IoT security is a multi-layered problem with the added complexity of practical implementation challenges arising from supplier diversity and legacy systems. Many IoT based application devices have limited resources in terms of computational processing capability, power, bandwidth, and memory. These resource-constrained application devices have a limited capability to support security-related functions. Rather than device-to-device communications, group communications in the form of broadcasting and multicasting incur efficient message deliveries among resource-constrained IoT devices. The major security concern in group communication is authentication, confidentiality and integrity of messages, forward and backward secrecy. In recent years several group key management schemes have been suggested for group based application to provide security of information in the group. In this paper, for a distributed group system, an efficient Group Diffie-Hellmann Key Exchange (GDH) algorithm, that is an extension of two party Diffie-Hellmann Key Exchange is proposed to exchange the common group key among all the members of group. The group key generated can then be used to encrypt or decrypt the messages and to prevent the intruders from gaining access to the group information. Since all the members contribute their own shares in group key generation, only the authorized group members will be able to generate the secret key, which prevents unauthorized group members from gaining access to group key. Hence, in order to maintain the backward and forward secrecy, the group keys are updated whenever a new member joins or leaves the communication group. The proposed algorithm is more efficient in terms of group key generation. Re-keying operation is performed immediately after membership changes and the updated group key is entirely different from the previous group key. Moreover, the key generated is also very strong and secure since, it uses cryptographic techniques.

Keywords: IOT, Group Key management, group key, Diffie-Hellman Key Exchange, authentication, confidentiality, integrity, forward secrecy, backward secrecy

1. Introduction

Numerous applications such as video and audio conferencing, data communication, information service, etc., utilize group communication model. In secure group communication, members of the group can communicate with each other by sending messages and the messages are secured to such an extent that exclusive group members can get to those messages. So, the principal objectives of a secure group communication model are to provide confidentiality, integrity and authentication of the messages sent to the group.

Cryptography is the only means by which the security can be enhanced in the group communication. In most applications, in view of the group communication, the security challenge lies in giving a viable strategy to controlling access to the group and its data. communication inside the group is often prone to a few attacks, for example, Replay attack, Impersonation attack, Eavesdropping, Denial of Service (DoS) attacks etc.

A primary method of limiting access to information is through encryption and selective distribution of the keys to encrypt group information. An encryption algorithm takes input information (e.g., a Group message) and plays out a few transformations on it utilizing a cryptographic key. This procedure creates a ciphered text. There is no simple approach to recover the first message from the ciphered text other than by knowing the correct key. Applying such a method, one can run secure group sessions. The messages are protected by encryption, utilizing the chosen key, which in the context of group communication is known as the group key. Only the individuals with the group key will be able to recover the first message. So the general security of the group relies upon the secrecy and strength of the group key.

In a secure group communication model, a symmetric key is shared between all the members of group and this symmetric key is utilized to scramble and decode the messages. In a group the members are not fixed and they may leave or join the group at any time. The process of refreshing and circulating the group key to all the members of the group as soon as any member joins or leaves the group is called re-keying. Every secure group communication model must maintain certain requirements that must be followed strictly. The requirements are as follows:

- If a new member joins the group, then re-keying operation should be performed in order to refresh the group key. This group key refreshing avoids the new member from accessing the previous messages and maintains backward secrecy.
- If an existing member leaves the group, then re-keying operation should be performed in order to change the group key. This avoids old member from accessing the future messages and maintains forward secrecy.

The most important issue in secure group communication is how to distribute the group key to the members, both initially or after a member joins/leaves, the total communication cost involved and the total number of encryptions performed should be minimized and should work for a large dynamic group, without any scalability problem. Hence, each independent secure group should have its own group key.

Based on the nature of communication a key management scheme can be classified into three categories:

Centralized Group Key Management: There is a trusted central key distribution server also known as Key Distribution Center (KDC) which is responsible for key generation and distribution of keys among the members of the group. Hence a group key management protocol seeks to minimize storage requirements, computational power on both client and server side, and bandwidth utilization.

However, with only one managing entity, the central server is a single point of failure. The entire group will be affected if there is a problem with the controller. The group privacy is dependent on the successful functioning of the single group controller; when the controller is not working, the group becomes vulnerable to attacks, because the keys, which are the base for the group privacy, are not being generated/regenerated and distributed [1].

Decentralized Group Key Management: The large group is split into small sub-groups and different controllers are used to manage each sub-group. This scheme minimizes the problem of concentrating the work on a single place. Group keys can be easily managed because it avoids the single point of failure and its hierarchy makes it easy to manage group key [1].

Distributed Group Key management: There is no explicit Key Distribution Centre (KDC), and the members themselves do the key generation. All the members can perform access control and the generation of the key can be either contributory, meaning that all members contribute some information to generate the group key, or done by one of the members. Distributed group key management scheme is robust against transmission delay, network failure or compromise of node keys [1].

In view of securing group communication, this paper presents an efficient algorithm based on distributed group key management scheme. In reality the membership of the group is not statistic; users may join or leave the group at any time. Further, group may split into disjoint components due to network failure or even attacks. Since, each and every user contributes its own share to exchange common secret key within the group; no users other than group can exchange the secret key. To provide secure group communication the group key is updated immediately after membership changes. The updated secret key after membership change is entirely different from previous or future updated secret keys. Various cryptographic techniques implemented at each stage of group key management ensure strong and safe exchange of group key.

2. Literature survey

A secure group communication model relies strongly on the group key management protocols. The group key management protocol aims at providing the secret group key which is shared between all the members within the group. This shared group key is used to encrypt and decrypt the messages that need to be communicated securely over the network. The shared group Key also provides authentication of each group member and thus provides the control access to the group and its data.

The group key must be updated and re-distributed to all the authorized users in a Secure and reliable fashion whenever user joins or leaves the group. Thus group communication model strongly relies on the cryptographic strength of the group key and the way in which group key management schemes are implemented to generate the group key.

Even though group communication can be benefited from IP multicast to achieve scalable exchange of messages, there is a challenge of effectively controlling the access to the transmitted data [1]. IP multicast doesn’t itself provide any mechanism for preventing non-group members to have an access to the group communication. Although an encryption can be used to protect the message exchanged among the group members, distributing the cryptographic key becomes an issue. Researchers have proposed different approaches to the group key management. These approaches can be divided into three main classes: Centralized group key management protocols, Decentralized group key management and Distributed group key management protocols. The three classes are described and an insight given to their features and goals. The area of group key management is then surveyed and proposed solutions are classified according to those characteristics.

A new research direction of group key management and its classification are well described in paper [2]. Each of these methods has its own merits subject to the network size, membership dynamics and loss characteristics.

In paper [3] E.Bresson et al. proposed the provably Secure Authenticated Group Diffie Hellman Key Exchange Protocol which provides secure communication within the group. The proposed model is sufficiently generic to be adapted to many cryptographic scenarios and is well suited for the key generation and reliable sharing among the users in the group. The proposed protocol is efficient in distributed system and is able to provide secure authentication of the group entities.

In paper [4] Manulis and Emmanuel Bresson proposed Tree Diffie Hellman protocol. This is a distributed group key management protocol that satisfies strong security goals such as authenticated key exchange and mutual authentication for a group in the presence of powerful adversaries. Group key generation using Tree Diffie Hellman is immune against all types of possible attack and it proceeds in three rounds. Thus it tries to reduce the processing and communication requirement.

Hierarchies arise in the context of access control whenever the user population can be modeled as a set of partially ordered classes (represented as a directed graph). A user with access privileges for a class obtains access to objects stored at that class and all descendant classes in the hierarchy. The problem of key management for such hierarchies then consists of assigning a key to each class in the hierarchy, so that keys for descendant classes can be obtained via efficient key derivation. In this the security of the scheme is based on mainly pseudorandom functions, without reliance on the Random Oracle model [5].

Giuseppe Ateniese, Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci [6] proposed a time-bound hierarchical key assignment scheme which is a method to assign time-dependent encryption keys to a set of classes in a partially ordered hierarchy, in such a way that each class can compute the keys of all lower classes in the hierarchy, according to temporal constraints. The two different constructions for time-bound key assignment schemes will be considered. The first one is based on symmetric encryption schemes, whereas the second one makes use of bilinear maps. Both constructions support updates to the access hierarchy with local changes to the public information and without requiring any private information to be re-distributed. The scheme is provable secure and efficient.

Jason Crampton considered interval-based access control policies, [7] of which temporal and geo-spatial access control policies are special cases. Such policies can be enforced using cryptographic methods, often called key assignment schemes.

Based on the above survey, this work mainly concentrates on the group key management for achieving the secured group communication. The study and implementation is based on the authenticated group Diffie Hellman Key Exchange protocol, which belongs to the class of distributed group key management. This method is used to generate the common secret key among all the members of the group. This scheme does not relay on the group controller during the setup time has an advantage over those schemes with the group controller, because all the members are considered with equal priority. Failure of one or more members in the group setup will not affect the performance of group operation.

3. Contribution

This paper presents secure group communication model based on group Diffie Hellman Key Exchange protocol. In this model, the two party Diffie Hellman Key Exchange, that generates the common secret key between two users, is extended to the group. The most significant approach of the proposed Group Diffie Hellman Key Exchange (GDH) protocol is that no users other than the group can have access to the group communication. Each user in the group is strongly authenticated such that group key is updated whenever the member joins or leaves the group and thus provides backward and forward secrecy. The updated group key is completely independent from any previously used and future secret group keys. The unauthorized users without group key cannot gain access to the group communication.

Diffie-Hellman Key Exchange is not limited to negotiating a key shared by only two participants. The same algorithm can be extended to a group with larger size. Diffie-Hellman Key Exchange is a method of securely exchanging the cryptographic keys over a public channel and it is one among the public key protocols. The key is shared between the participants in such a way that, the secret cannot be revealed by observing the communication. The protocol is considered secure against Eavesdropper, if all the basic parameters which are required to establish the secret key, are chosen properly. The Eavesdropper has to solve the Diffie-Hellman Problem (DHP) to obtain the secret key. The process of solving the DHP is considered difficult for the group, whose order is quite large. The security of Diffie-Hellman Key Exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is difficult to calculate discrete logarithm. For the larger primes, the later task is considered infeasible.

4. Diffie-Hellman Key Exchange

Diffie–Hellman Key Exchange (DHK) is a specific method of exchanging cryptographic keys. DHK prepares a context for safe communication over an unsecure channel between two parties without having any prior knowledge from each other by sharing an agreed secret key. This shared secret key is used for symmetric encryption and decryption of transmitted message within an unsecure channel.

Suppose user A and user B wish to communicate with each other and use a secret key to encrypt the message on that connection, the DHK algorithm consists of following steps to generate a common secret key:

- User A and B agrees on a prime number *p* and an integer *α* which is a primitive root of *p*.
- User A selects a random integer $x_A < p$ as a private key and computes the public key Y_A.

$$Y_A = \alpha^{x_A} \bmod p \tag{1}$$

- User B selects a random integer $x_B <$ p as a private key and computes the public key Y_B.

$$Y_B = \alpha^{x_B} \bmod p \tag{2}$$

- Each user public key will be made available to the other user.
- Using B's public key user A computes the secret key

$$K = {Y_B}^{x_A} \bmod p = \alpha^{x_A x_B} \bmod p \tag{3}$$

- Using A's public key user B computes the secret key *K*.

$$K = {Y_A}^{x_B} \bmod p = \alpha^{x_A x_B} \bmod p \tag{4}$$

In this way the two users have exchanged a common secret key.

5. Proposed method

The proposed method for generating the group key is based on the Group Diffie Hellman (GDH) key exchange protocol. Group Diffie Hellman scheme for authenticated key exchange is designed to provide a group of participants communicating over an insecure channel and each holding a pair of matching public or private keys with a common secret key. This secret key may be used to achieve some cryptographic goals such as data confidentiality and integrity. Group re-keying is an important task that must be performed when the user joins or leaves the group, thus providing backward secrecy and forward secrecy respectively.

The Group is a finite cyclic group with '*n*' number of user $u_1, u_2 \ldots\ldots\ldots\ldots$ is considered with prime order . Each user is assigned with a private key (where $i = 1,2,3 \ldots$ which is chosen randomly over an interval $[1, p -$.The user participants are arranged in the ring and user with highest index in the group is considered as the group controller for some time interval. All the operation uses multiplicative group of integers modulo p. Figure 1 illustrates the process of operation involved in group key generation.

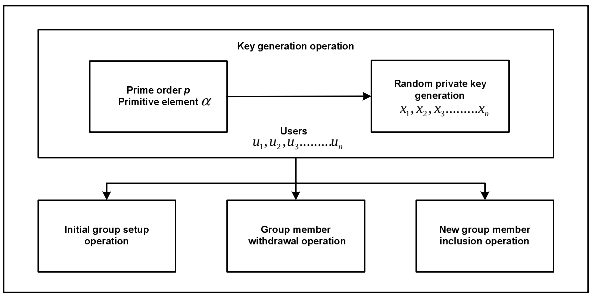

Key generation operation
Prime order *p*
Primitive element α
Random private key generation
$x_1, x_2, x_3 \ldots\ldots\ldots x_n$
Users
$u_1, u_2, u_3 \ldots\ldots\ldots u_n$
Initial group setup operation
Group member withdrawal operation
New group member inclusion operation

Fig. 1. Group key generation process.

The secret group key generation for ensuring secure group communication consists of the following four steps of operation:

- Key generation operation
- Initial group setup operation
- Group member withdrawal operation
- New group member inclusion operation

5.1. Key generation algorithm

Each user in a group G agrees on the prime number *p*. Using *p* as a prime number, the primitive element α is generated, which is the primitive root of *p*. Algorithm 1 describes how the primitive root is generated using Lucas Primality test.

Then each user agrees to choose the private key $(for\ i = 1,2,3 \ldots$ over the random interval [1, *p*-1].

Then each user agrees to choose the private key x_i (for $i = 1,2,3,\ldots\ldots n$) over the random interval [1, *p*-1].

5.2. Initial group setup algorithm

The algorithm consists of two stages: Up-flow and Down-flow. The group *G* (For example consided the initial group with *G*= $\{u_1, u_2, u_3, u$) is set to *I*. As illustrated in Figure 2 in the up-flow (FL denotes flow) the user receives a set *X*= (*Y*, *Z*) of intermediate values, with

Algorithm 1: Primitive root generation

Input: Prime number

Output: Primitive element *α*

- Consider the prime number *p*
- Choose α randomly in the range [2,*p*-1]
- α is the primitive root of prime number p, if the following conditions are satisfied

  $$\alpha^{p-1} \equiv 1(\mathrm{mod}\ p)$$

  and for every prime factor *q* of (*p*-1)

  $$\alpha^{(p-1)/q} \neq 1(\mathrm{mod}\ p)$$

$$Y = \bigcup_{1 \le m < i} \{Z^{1/x_m}\} \qquad (5)$$

$$Z = \alpha^{\prod_{1 \le t < i} x_t} \qquad (6)$$

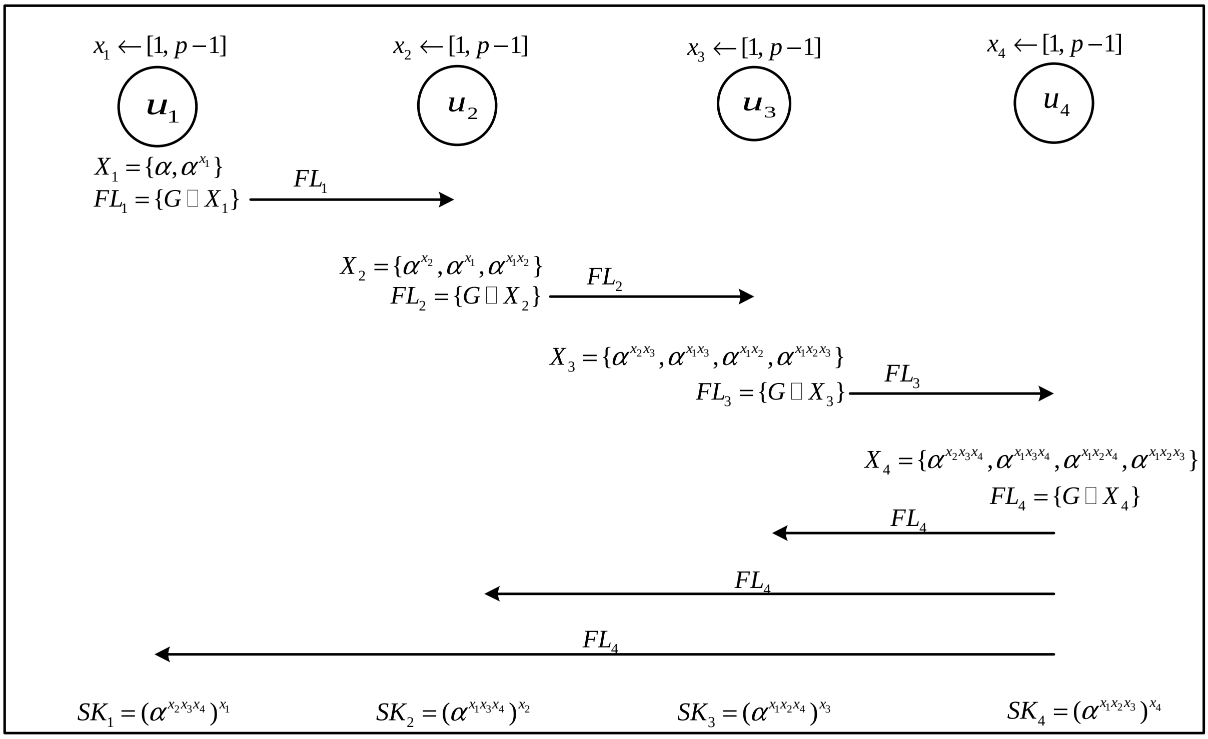
Each user randomly chooses a private value , and then raises the values in *Y* to the power of and then concatenates with *Z* to obtain its intermediate values

$$Y' = \bigcup_{1 \le m \le i} \{Z'^{1/x_m}\} \qquad (7)$$

$$Z' = Z^{x_i} = \alpha^{\prod_{1 \le t \le i} x_t} \qquad (8)$$

Then each user then forwards the values (Y') to the next user in the ring. Down-flow takes place when the last user with highest index denoted by u receives the last up-flow. At that point the last user performs the same steps the user in the up-flow but broadcasts the set of intermediate values to the rest of the users in the group.

In effect, the value computed by the last user will lead to the secret key *SK*, since $Z' = \alpha^{\prod_{1 \le t \le n} x_t}$. User in group *G* computes *SK* and is accepted as secret key.



$x_1 \leftarrow [1, p-1]$
$x_2 \leftarrow [1, p-1]$
$x_3 \leftarrow [1, p-1]$
$x_4 \leftarrow [1, p-1]$
u_1
u_2
u_3
u_4
$X_1 = \{\alpha, \alpha^{x_1}\}$
$FL_1 = \{G \square X_1\}$
FL_1
$X_2 = \{\alpha^{x_2}, \alpha^{x_1}, \alpha^{x_1x_2}\}$
$FL_2 = \{G \square X_2\}$
FL_2
$X_3 = \{\alpha^{x_2x_3}, \alpha^{x_1x_3}, \alpha^{x_1x_2}, \alpha^{x_1x_2x_3}\}$
$FL_3 = \{G \square X_3\}$
FL_3
$X_4 = \{\alpha^{x_2x_3x_4}, \alpha^{x_1x_3x_4}, \alpha^{x_1x_2x_4}, \alpha^{x_1x_2x_3}\}$
$FL_4 = \{G \square X_4\}$
FL_4
FL_4
FL_4
$SK_1 = (\alpha^{x_2x_3x_4})^{x_1}$
$SK_2 = (\alpha^{x_1x_3x_4})^{x_2}$
$SK_3 = (\alpha^{x_1x_2x_4})^{x_3}$
$SK_4 = (\alpha^{x_1x_2x_3})^{x_4}$

Fig. 2. Illustration of initial group setup algorithm

5.3. Group member withdrawal algorithm

This algorithm consists of Down-flow (FL denotes flow) only. The group *G* is set to *G\I*. As illustrated in Figure 3 (For example consider the initial group with *G*= $\{u_1, u_2, u_3, u$. Let user to be withdrawn is denoted by *I*= $\{u_2, u$. The new Group is then set to (*G\I*)= $\{u_1, u$ and $u_{GC} =$) the group controller u (i.e. user with the highest-index in *G\I*) generates a new random private value x and removes old private key x (x is u*'s* previous secret value) from the saved previous broadcast values designated to the users in *I*. u then raises all the remaining values in which x appeared, with the power of ($x_{GC}^{-1} . x$) and broadcasts the result. Users in Group (*G\I*) compute *SK* and accept the common secret key. Users in *G* erase any previous internal data and thus u erases x by internally saving x.

5.4. New group member inclusion algorithm

This algorithm consists of two stages: Up-flow and Down-flow (FL). The group *G* is then set to $(G \cup$. As illustrated in Figure 4 (For example consider the group with *G*= $\{u_1, u$. Let the user to be included is denoted by *I*= $\{u$ and $u_{GC} =$. The new Group is $(G \cup =$ $\{u_1, u_2, u$) the group controller u (i.e. user with the highest-index in *G*) generates a new private random value x, raises the values from the saved previous broadcast in which x appears with the power of ($x_{GC}^{-1} . x$) and obtains a set of values . (x is u's previous secret exponent) . u also computes the value by raising the last value in to x. u then forwards the values (,) to the first joining user in *I*. From that point inclusion algorithm will work as the initial group setup algorithm. Upon receiving the broadcast flow, users in $(G \cup$ erase previous session key, then compute *SK* and accept the common secret Key.

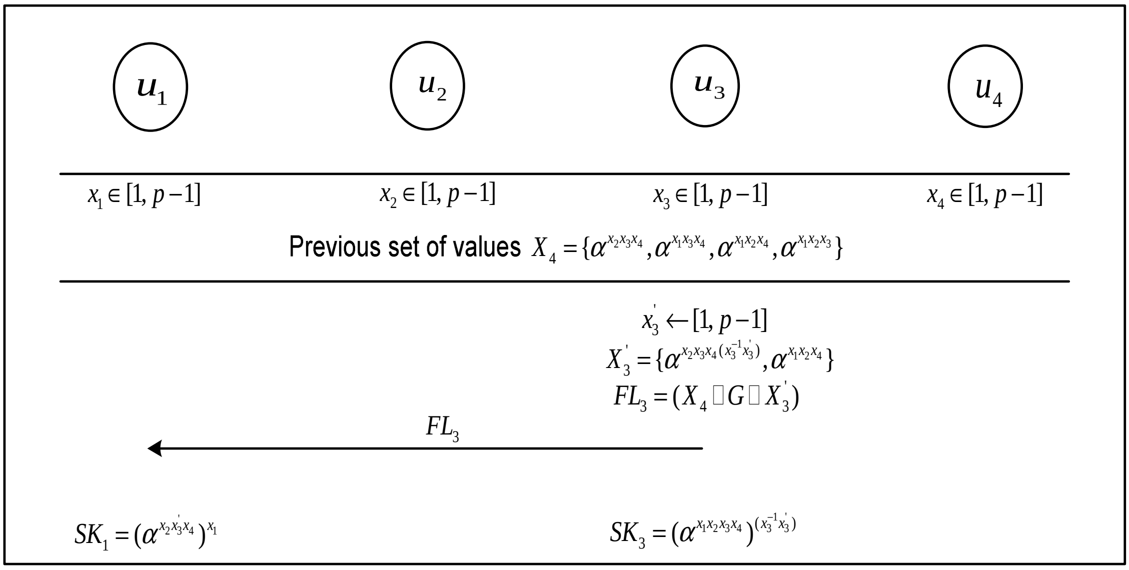

u_1
u_2
u_3
u_4
$x_1 \in [1, p-1]$
$x_2 \in [1, p-1]$
$x_3 \in [1, p-1]$
$x_4 \in [1, p-1]$
Previous set of values $X_4 = \{\alpha^{x_2 x_3 x_4}, \alpha^{x_1 x_3 x_4}, \alpha^{x_1 x_2 x_4}, \alpha^{x_1 x_2 x_3}\}$
$x_3^{'} \leftarrow [1, p-1]$
$X_3^{'} = \{\alpha^{x_2 x_3 x_4 (x_3^{-1} x_3^{'})}, \alpha^{x_1 x_2 x_4}\}$
$FL_3 = (X_4 \,□\, G \,□\, X_3^{'})$
FL_3
$SK_1 = (\alpha^{x_2 x_3^{'} x_4})^{x_1}$
$SK_3 = (\alpha^{x_1 x_2 x_3 x_4})^{(x_3^{-1} x_3^{'})}$

Fig. 3. Illustration of group member withdrawal algorithm

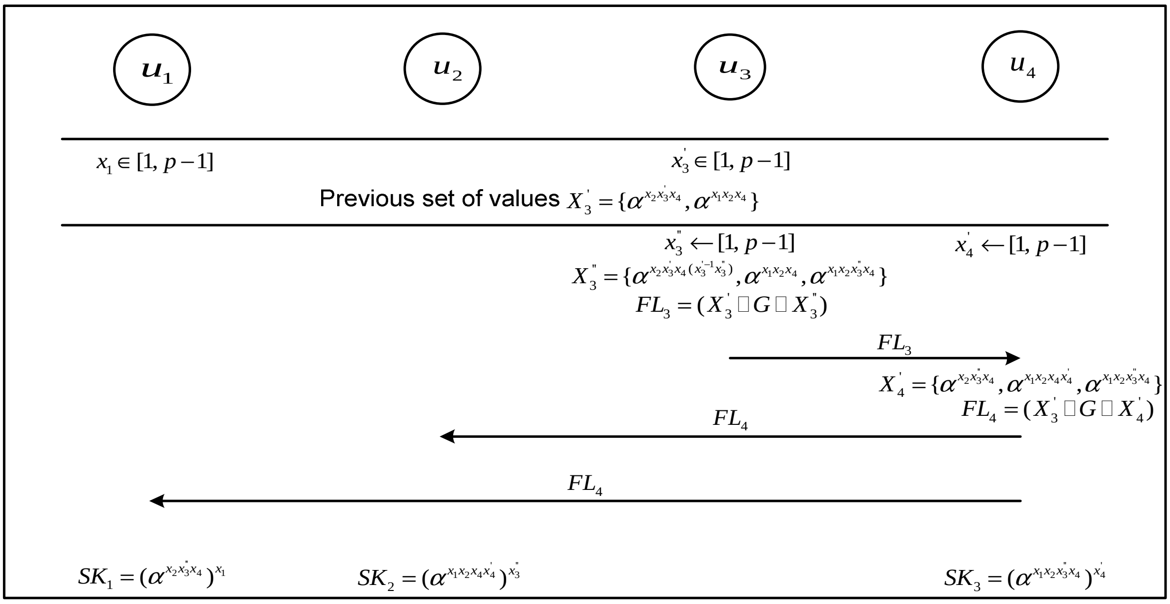

u_1
u_2
u_3
u_4
$x_1 \in [1, p-1]$
$x_3^{'} \in [1, p-1]$
Previous set of values $X_3^{'} = \{\alpha^{x_2 x_3^{'} x_4}, \alpha^{x_1 x_2 x_4}\}$
$x_3^{"} \leftarrow [1, p-1]$
$x_4^{'} \leftarrow [1, p-1]$
$X_3^{"} = \{\alpha^{x_2 x_3^{'} x_4 (x_3^{'-1} x_3^{"})}, \alpha^{x_1 x_2 x_4}, \alpha^{x_1 x_2 x_3^{"} x_4}\}$
$FL_3 = (X_3^{'} \,□\, G \,□\, X_3^{"})$
FL_3
$X_4^{'} = \{\alpha^{x_2 x_3^{"} x_4}, \alpha^{x_1 x_2 x_4 x_4^{'}}, \alpha^{x_1 x_2 x_3^{"} x_4}\}$
$FL_4 = (X_3^{'} \,□\, G \,□\, X_4^{'})$
FL_4
FL_4
$SK_1 = (\alpha^{x_2 x_3^{"} x_4})^{x_1}$
$SK_2 = (\alpha^{x_1 x_2 x_4 x_4^{'}})^{x_3^{"}}$
$SK_3 = (\alpha^{x_1 x_2 x_3^{"} x_4})^{x_4^{'}}$

Fig. 4. Illustration of new group member inclusion algorithm

6. Experimental results

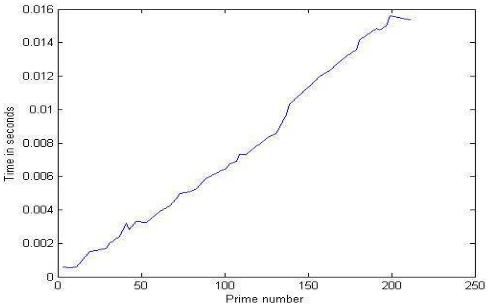


Fig. 5. Computation time for obtaining primitive root of prime number

The implementation of the proposed work is done using MATLAB tool. Figure 5 shows the importance of prime number *p* in an efficient key management of the group key. It is to be noted that computational time for obtaining the primitive root of the prime number increases gradually with the increase in the prime value range. Prime numbers are mainly used in the cryptography, since it takes considerable time to determine whether it is prime number or not. For the hackers it takes more time to break the algorithm, rendering it inappropriate. The Prime number with more computation time to obtain the primitive root of a prime number indicates that, the group key management algorithm becomes more secure and safe when the algorithm agrees on the largest possible prime value *p*.

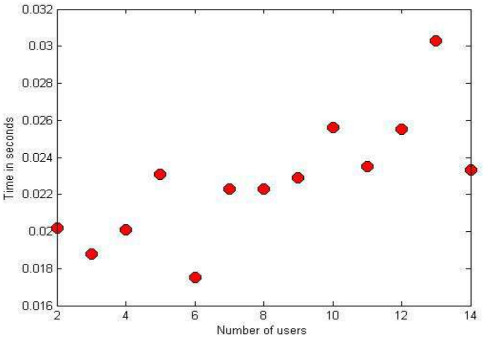


Fig. 6. Group key generation time versus number of users

The largest possible prime value is selected and the initial group setup algorithm is implemented by considering different number of users. Initially each user agrees on the random private key. Finally the common secret group key is computed by all the participants in the group. Figure 6 shows computation time taken by the algorithm for the generation of group key with varying group size. At some point it is to be noted that, with some group size, there is a sudden increase or decrease in the time consumption for performing group key generation. This is due to the fact that, the different users initially agree on the random selection of private key. Further varying mathematical computation of exponential powers leads to the fluctuation in implementation time.

When the group member withdrawal algorithm is implemented, the group key is updated with the successful removal of the participant. This implementation results in a new updated group secret key and it is also verified that the updated group key is entirely different from the previous group key.

The implementation of new group member inclusion algorithm also results in new updated group key with the addition of new participant to the group. Initially the new member is included to the group by randomly selecting the private key. Thereby the generated group key is completely different from the previous group key.

7. Conclusion

The Group key management provides fundamental security service in group communication. To provide the security of the data transmitted over an insecure channel, the group key is generated in reliable fashion and distributed among the group members. In this paper Group Diffie-Hellman Key Exchange algorithm is used to generate and distribute the secret group key. Similarly with the membership changes the group key is updated immediately. The proposed scheme is efficient in terms of providing authentication, confidentiality, integrity, forward secrecy and backward secrecy. Further secure communication in a group can be established in an efficient manner by performing symmetric encryption and decryption of the message/data with the help of group key. The group communication is secured against several possible attacks (Eavesdropping, Denial of service (DoS) attack etc.,) by the efficient management of group key.

References

[1] Sandro Rafaeli and David Hutchison, “A survey of key management for secure group communication,” ACM Computing Surveys, vol. 35, no. 3, pp. 309–329, September 2003.

[2] Sencun Zhu and Sushil Jajodia, “Scalable Group Key Management for Secure Multicast: A Taxonomy and New Directions”, Network Security, pp.57-75, June 2010.

[3] Emmaneul Bresson and David Pointcheval, “Provably secure authenticated group Diffie Hellman key exchange,” ACM Transactions on Information and System Security, vol. 10, no. 3, pp. 1–45, July 2007.

[4] Emmaneul. Bresson and Mark Manulis, “Securing group key exchange against strong corruptions,” ACM Symposium on Information, Computer and Communications Security, vol. 8, pp.249–260, March 2008.

[5] Mikhail J. Atallah, Marina Blanton, Nelly Fazio and K. B. Frikken, “Dynamic and efficient key management for access hierarchies,” ACM Symposium on Information and system Security, vol. 12, no. 3, pp. 1–43, January 2009.

[6] Giuseppe Ateniese, Alfredo De Santis Anna Lisa Ferrara, and Barbara Masucci, “Provably-secure time bound hierarchical key assignment schemes,” Journal of Cryptography, vol. 25, pp. 244–270, November 2010.

[7] Jason Crampton, “Practical and efficient cryptographic enforcement of interval-based access control policies,” ACM Transactions on Information and System Security, vol. 4, no. 1, pp. 1–30, May 2011.

[8] W. Stallings, “Cryptography and Network Security: Principles and Practice”, fifth ed., Prentice Hall, 2011.