Queue and tree based Group Diffie Hellman Key Agreement Protocol for E-Commerce Security

Ved Parkash*1, Roshan Lal*2

*Computer Science and Engineering Department, CDLSIET Panniwala Mota, Haryana, India.
*Department of Computer Science, Pt. Chiranjii Lal Sharma Govt. P.G. College Karnal, Haryana, India.

vpbhukal24@gmail.com
roshanhiranwal@gmail.com

Abstract — E-commerce security is required in each and every corner of the information and communication Technology (ICT). There are many uses of applications that involves group communications like secure conferencing, video games, video chatting etc. are rapidly increasing day by day. Many works are done in providing secure communication in the field of cryptography and network security. A group key agreement protocol helps to agree upon a common session key which is shared among the members of the group over a public network. This paper presents efficient group key agreement protocol based on queue with a tree based Diffie Hellman structure. So, with the help of queue most of the computation is performed by a high performance member in order to improve efficiency. We have designed bilinear pairing free group key agreement protocol. The computations in bilinear pairing are very complex. The proposed protocol consists of a group of users or participants and Group Controller Server, which validates each user or participants. Group of users give their contribution to form common group key in different levels of the tree. The security analysis of the proposed protocol shows that it is secure against active and passive attacks.

Keywords— E-commerce, Tree, Group Diffie Hellman Key, Agreement Protocol, Security

I. INTRODUCTION

Several groupware applications need a cryptographic key to establish secure communication over insecure channels. The computation of group key is very important for secure group communication over open networks. There are several efforts taken by researchers to enhance the efficiency of existing group key agreement protocol. In group key agreement protocol every member contribute in the construction of the group key. This paper focuses on establishing an efficient group key based on queue and with ternary tree structure. In the proposed protocol we have chosen to use queue as it helps us to filter out high performance member and low performance member, so that most of the computation is given to a high performance member. We have used the tree based group Diffie Hellman for constructing the cryptographic key.

Although several trees based group key establishment technique like CCEGK [1], EGK [2], TGDH [3], STR [4], etc. are available in literature, in which binary tree and two parties Diffie Hellman key exchange is used for computing group key. However [5] introduces ternary tree in their protocol and uses GDH.2 [4] as the basic operation for the group of restricted size 3k where k is any integer. ECC based cryptosystem improves the efficiency of the cryptographic technique and offers same level of security. ECC uses shorter key sizes than existing methods. ECC is based on some intractable problems like discrete logarithm problem, Computational Diffie Hellman problem, Decisional Diffie Hellman problem. The use of elliptic curve in public key cryptography was independently proposed by Koblitz and Miller in 1985. Since then a lot of work has been done in ECC to improve the efficiency of generating a group key.

The rest of this paper is organized as follows. In Section II some related works on group key agreement protocols based on queue, tree structure and the protocol proposed are discussed. The preliminaries related to proposed work are addressed in Section III. The Section IV proposes the protocol, while Section V provides its security analysis. Section VI compares the performance of proposed protocol with some existing one, followed by a conclusion section.

II. RELATED WORK

E-commerce security is required in all the communications during the data transfer and formal ways of communication. Therefore, a security model is required to maintain the security during the communication. Many group key establishment
protocols have been proposed by different authors. The group key generation can take place in three ways: centralized, Decentralized, Distributed [6].

- In centralized way of group key generation, key generation center trusted third party is responsible for constructing and distributing group key to the participants. One problem with this is that trusted third party must be available all the time to support the participants in the generation of the key.

- In decentralized way group key generation, a number of participants are divided into subgroups and these subgroups are managed by subgroup controller. There is problem with this is that if any of the subgroups fails, then whole group key generation fails.

- In distributed way of key generation, the group key is generated with the contribution of all members. Each member contributes their own share to compute the group key. In this individually no one can generate the group key.

S. Hong [7] has used queue based structure for the generation of group key. In this low performance member is filtered out by Group Controller Server (GCS) and stored in queue in such a way that high performance members stored in the front and low performance members are stored in the rear of the queue. In this fifty percent of members considered as high performance members. GCS makes subgroups of two members (one is a high performance member and another is a low performance member) in each round and then apply basic Diffie Hellman key exchange protocol. With this efficiency of group key generation is improved. Kumar et al. [6] has proposed ternary tree based group Diffie Hellman key exchange protocol in which a subgroup of three participants is made. Each member of subgroup performs three parties Diffie Hellman and agrees to a common point. Now, the x-coordinate on common key is used in the next round of the generation of key. We have taken the idea of a queue with the structure of the tree [8,13,15,18]. As in queue fifty percent is considered as high performance while in our protocol (1/3)rd member of the subgroup is considered as a high performance member and rest (2/3)rd as a low performance member.

III. PRELIMINARIES

In this section we will see the preliminaries of elliptic curves and Diffie Hellman key exchange protocol, which are required for the proposed protocol.

A. Elliptic curve group

An elliptic curve E over a prime finite field E is defined by Equation 1:

\[ Y^2 \mod p = (x^3 + ax + b) \mod p. \]  

Where \( a, b \in F_p \) Elliptic curves are non-singular and condition for non-singularity is defined by Equation 2:

\[ \Delta = (4a^3 + 27b^2) \mod p \neq 0. \]

B. Two Party Diffie-Hellman

This protocol helps us to establish shared cryptographic key between the two users which helps in the encryption and decryption of the message during communication over a public channel. Key exchange is done in the following way:

- A and B chooses their own private keys \( a_1, a_2 \in Z_p^* \) respectively and keep it secret.
- A will compute \( X = a_1P \) and sends to B.
- In the same way B will compute \( Y = a_2P \) and sends Y to A. B computes key as \( K = a_2X \).
- Now A will compute key as \( K = a_1Y \). A and B uses K as shared cryptographic key for encryption and decryption.

C. Three Party Diffie Hellman

Three parties Elliptic Curve Diffie-Hellman Protocol is based on GDH.2 implemented with elliptic curve for the group of the three parties (A, B, C) as follows:

- A, B and C choose their own private keys \( a_1, a_2, a_3 \in Z_p^* \) respectively and keep it secret.
- A calculate \( X = a_1P \) and send to B.
- B calculates \( Y_1 = a_2P; Y_2 = a_2X \) and construct the set \( X, Y_1, Y_2 \) which is sent to C.
- C calculates \( K = a_3Y_2; Z_1 = a_3Y_1 \) and \( Z_2 = a_3X \). It keeps secret K as the contributory group key and the broadcast remaining \( Z_1, Z_2 \) to the user A and B.
- On receiving from C, A and B calculates same group key as \( A: K = a_1Z_1 \) and B: \( K = a_2Z_2 \).
- Thus, all three members use \( K =a_1Y_2 = a_2Z_1 = a_2Z_2 = a_1a_2a_3P \) as the secret key used for encryption and decryption.

D. Group Controller Server
Group Controller Server (GCS) maintains the database containing information of each member of the group. It stores the public key of all the members of the group in the queue in order of their arrival. The fastest members kept in front and slowest members are kept at the rear side of the queue. GCS selects two fastest member and one slowest member and forms a group of three members. These members perform three party Diffie Hellman key exchange protocol and agree at some common point. Among the groups of three members, one fastest member is chosen and again forms group of three members. This is repeated until only one group is left so that every member will agree on common key. GCS is similar to virtual synchrony that runs the client Daemon program to manage membership [7]. The Virtual Synchrony (VS) daemon runs on each machine at participants side and synchronizes with other’s VSs to update membership whenever membership changes so that every member is able to know the other member’s status due to VS [7].

IV. PROPOSED PROTOCOL

This section explains the proposed protocol for group key agreement protocol with the help of queue and ternary tree based group Diffie Hellman protocol. In this consider a group of members (U₁, U₂,…,Uₙ) and a Group Controller Server(GCS). GCS contains the information like user IDs, Passwords, and Blind Key Queues (BKQ) of the members of the group. Blind key is same as the public key of the user. A queue based Key agreement is shown in Fig. 1.

 Initialization

- Each member Uᵢ logsins to the GCS and the GCS will validate his ID, password by checking member’s information in the database.
- Each user Uᵢ chooses a random number aᵢ ∈ Zₚ * and keep it secret.
- After validation each members, Uᵢ starts to send their public key as aᵢP to GCS where P is the base point of elliptic curves.
- GCS will collect all the public keys received from the members of the group and store them in the queue in order of arrival so that GCS is able to identify fastest member (high performance member) and slowest member (low performance member).
- The fastest members are stored in front and slowest members are stored in the rear of the blind key queue.
- The GCS will decide which member goes to the next level for the generation of the key.

Key Generation Phase

Now the computation of group key is done with the help of key tree is as follows:

- GCS will invite the participants who will go to perform at the next level.
- GCS will make subgroups G₁,G₂,…,Gₘ of three members consists of two slowest member and one fastest member.(If n is not multiple of three then remaining 1 or 2 members will go in the next level for the group key generation and they will not do anything in current level. This is same in all levels).
The members of group \(G_1, G_2, \ldots, G_m\) performs three party Diffie Hellman key exchange protocol and agree on a common point as \(K_i = a_{i1}a_{i2}a_{i3}P\), where \(a_{i1}, a_{i2}, a_{i3}\) are private keys of members \(U_{i1}, U_{i2}, U_{i3}\) of subgroups \(G_1, G_2, \ldots, G_m\) with \(i = 1, 2, \ldots, m\).

Now GCS will choose one of the fastest members as Group controller (GC) from each subgroup for the next level. In this way every subgroup is represented by GC and treated as new node.

In the second round GCS again makes subgroup of these nodes having set of three participants of each and calculate their subgroup’s common point as in the previous.

At this time GCs choose \(x\)-coordinate \(K_{ij}\) of common point \(K_i\) of each subgroup \([G_1, G_2, \ldots, G_m]\) is used and multiply with the base point \(P\) as \(K_{ij}P\).

\(GC_1\) calculates \(x_{1j}\) and unicast to \(GC_2\), \(GC_2\) calculates \(x_{2j}\) and \(x_{3j}P\) and broadcast \([x_{1j}P, x_{2j}P, x_{3j}P]\) to all members of third subgroup. The members of the third subgroup now can calculate the common key as \((x_{1j}x_{2j}x_{3j}P)\) and keep it secret. \(GC_1\) additionally calculates \([x_{1j}P, (x_{2j}x_{3j})P]\) and broadcast to all members of its sibling groups. All sibling subgroup members calculate group key by multiplying their own private value.

Note: \(GC_1, GC_2, GC_3\) is the group controllers and \(x_1, x_2, x_3\) are their \(x\)-coordinates of common point i.e. \(K_{ij}\).

For the next round of key generation, GCS will again choose the fastest member from each subgroup \(G_1, G_2, \ldots, G_m\) and again form subgroups of 3 members and these subgroups compute common point as in the previous.

Similarly, in each round partial key is computed. In the last round there are two possibilities that either two members remained for making group or three members remained for making group. For three members’ three parties Diffie Hellman key exchange protocol is used and for two member’s two parties Diffie Hellman protocol is used.

At the end group key is computed as \(S_1 = k_{11}k_{12}k_{13}\ldots k_{im}P\)

V. SECURITY ANALYSIS

In this section we are going to prove that our proposed protocol meets the desirable security attributes under Elliptic Curve Discrete Logarithm Problem. Cryptanalysis involves determining \(x\) given \(Q\) and \(P\) where \(P\) is a point on the EC and \(Q = xP\) that is \(P\) added to itself \(x\) times. The security tolerance of the proposed protocol is discussed below:

- **Outsider attack**: If an attacker is an outsider, then GCS will not contain the information about that attacker in the database. Thus GCS will not validate it and will not select him for generation of group key.

- **Interior Collecting Attacks**: If a group member negotiates with one of its ancestors (parent) by knowing the key then there is no relation parameter among any of the ancestor nodes, thus it is not possible to obtain the key.

- **Known session key security**: At every level of key generation, each subgroup’s common agreed key changes. If the adversary compromises one session key, then he should not be able to compromise other session keys, thus this protocol provides known session key security.

- **No key control**: The group key is determined by the points which are computed by each subgroup. Each time for the next level of key generation subgroups changes. Thus nobody can individually generate the group key. The contribution of each member of the group key generation is mandatory.

VI. PERFORMANCE COMPARISON

This section shows a comparison of the proposed protocol with some other existing Group Key Agreement protocols on the basis of its performance in terms of communication and computation costs. The result is shown in Table I (where \(n\) is the total number of participants). The following notations are used for comparison:

- **PM**: number of Scalar point multiplications.

- **Message**: Total number of message overheads during group key generation process (including unicast and broadcast).

- **\(n\)**: Total number of participants.

- **Pairings**: number of bilinear pairing computations needed in the key agreement process (zero in case of our proposal).
• $h = \log_2 n$: The height of the original key tree in proposed technique

### Table 1 Comparison with the existing state of the art models

<table>
<thead>
<tr>
<th>Protocols</th>
<th>Rounds</th>
<th>Message</th>
<th>PM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TGEC DH [12]</td>
<td>$h = \log_2 n$</td>
<td>$2^h(n-1)$</td>
<td>$n^h(n-1) + n^h$</td>
</tr>
<tr>
<td>Maria et al. [13]</td>
<td>$h = \log_2 n$</td>
<td>$2^h(n-1)$</td>
<td>$n^h(n-1) + n^h$</td>
</tr>
<tr>
<td>GDH.2[14]</td>
<td>$n$</td>
<td>$n$</td>
<td>$n(n+3)/2 - 1$</td>
</tr>
<tr>
<td>GDH.3</td>
<td>$n+1$</td>
<td>$2n - 1$</td>
<td>$5n - 6$</td>
</tr>
<tr>
<td>BD[14]</td>
<td>$n$</td>
<td>$2n$</td>
<td>$n^*(n+1)$</td>
</tr>
<tr>
<td><strong>Proposed protocol</strong></td>
<td>$h = \log_2 n$</td>
<td>Floor$[3^*(n-1)/2]$</td>
<td>$5^<em>(n-1)/2 + h</em>n$</td>
</tr>
</tbody>
</table>

### VII. Conclusions

This paper proposes a pairing free tree based group key agreement protocol with the help of the queue. Group key management in today’s world is very important. The low performance machine takes more time to compute group key to establish secure communication for e-commerce. So, we should make efficient group key agreement protocol which reduces computation overhead. With these assumptions we have proposed this protocol. This protocol is designed to decrease the computation load in such a way that the low performance member will not have computation overhead. The GCS makes group of high performance member and low performance member, with this burden of computation decreases on each member. This protocol uses queue structure to differentiate high performance member and low performance member. We have implemented elliptic curve on queue structure. Thus the cost of the various group operations is compared with others existing techniques which shows its efficient performance than the existing protocols.

### REFERENCES


[16] K. M. S. Maria Celestin Vigila. Ecc based contributory group key computation scheme using one time pad.
