Ternary Tree based Group Key Agreement Protocol Over Elliptic Curve for Dynamic Group

Abhimanyu Kumar
Department of Computer Science and Engineering
Indian School of Mines,
Dhanbad-826004, Jharkhand, India

Sachin Tripathi
Department of Computer Science and Engineering
Indian School of Mines,
Dhanbad-826004, Jharkhand, India

ABSTRACT
Secure group communication is an active area of research and its popularity is fuelled by the growing importance of group oriented internet applications such as voice & video conferences, pay per view, etc. Several groupware applications like video conferences, distributed computations, etc requires secure transmission while communicating over the Internet. For secure communication, the integrity of the messages, member authentication, and confidentiality are must be provided among group members. To provide message integrity all group members must be agreed up on a common group key to encrypt and decrypt the messages. This paper proposes an efficient and contributory group key agreement protocol and also support dynamic operations like join, leave, merge, etc. by using ECC based Diffie Hellman key exchange. This protocol employs ternary tree like structure instead of binary tree in the process of group key generation. The performance of the proposed scheme is compared with that of several others existing schemes in literature and it is found that the proposed one is performs well in terms of communication and computation cost. In addition, the formal security validation is done using AVISPA tool that demonstrated that the proposed protocol is safe against passive and active attacks.

Keywords
ECC, group key agreement, ternary tree, ECC based Diffie-Hellman, AVISPA.

1. INTRODUCTION
For several groupware applications like voice & video conferences, distributed computation etc. over the insecure network like Internet, it is require to develop an efficient group key agreement protocol for secure communication. Consider a group of N members in a network would like to discuss on secrete common concern. These N members must able to communicate among themselves over a public channel in a secure manner such that any user other than the group must not be able to listen in to the conversation between legitimate members. The general aim of secure group communication is to construct a common secrete key among the group members for confidential communication. Although several tree based group key establishment technique like CCEGK [27], EGK [1], TGDH [11], STR [10], etc. are available in literature, all of which employ a binary tree for computing group key and uses two parties Diffie Hellman key exchange as the basic operation. However [24] introduces ternary tree in their protocol and uses GDH.2 [23] as the basic operation for the group of restricted size 3^k where k is any integer. In order to improve the efficiency of the cryptographic technique the ECC based cryptosystem can play an important role since it offers similar level of security which can be achieve with shorter keys size than existing methods which are based on difficulties of solving discrete logarithms over integers or integer factorization. The use of elliptic curve in public key cryptography was independently proposed by Koblitz and Miller in 1985 [12] and since then, an enormous amount of work has been done on elliptic curve cryptography. This paper proposes ECC based contributory key computation for secure group communication in dynamic environment. The proposed technique organises the key generation process in ternary tree like structure in which every node can have at most three children as in [24], but there is no restriction of the no of group members (not necessary 3^k as in [24]). The advantages of using ternary tree instead of binary tree are already justified in [24]. The protocol uses three parties ECC based Diffie Hellman key exchange which is discussed in section 3.3 as its basic operation. Along with the ternary tree, since the proposed technique uses ECC approach which has low computation cost and smaller key size, the overheads are reduces and the performance of the protocol improves significantly.

The rest of the paper is organized as follows. Section 2 summarises the related works. Section 3 describes the Preliminaries of the entire paper. Section 4 discusses the proposed protocol followed by a pictorial representation of step by step group key generation process between the members. Section 5 discussed the suggested authentication scheme for proposed protocol. In section 6 security analysis of proposed protocol are discussed followed by its verification result using AVISPA tool in Section 7. Section 8 provides analytical performance comparison with other existing protocols and finally section 9 concludes this paper.

2. RELATED WORK
Several approaches have been proposed for group key generation in current literature. These approaches can be classified into three categories: Centralized, Decentralized and distributed approaches [5, 6, 19, 21].

In Centralized approaches [8, 14, 21] an entity usually called key server, is responsible for generation and distribution of group key to all members of the group a trusted third party (TTP) can make this possible. However the main problem with this approach is the TTP must be constantly available and in every possible subset of group there must be a TTP available in order to support continued operation in the event of network partitions.

In decentralized approaches [13, 17, 18] the whole group is split into small subgroups. Each subgroup is managed by Subgroup Controller (SC) which minimizes the problem of concentrating the work on a single point. The failure of one SC will not escort to the failure of the whole group. But also

in most of the decentralized technique, the SC may become a bottleneck because the SC must decrypt the group messages and then re encrypt it using the sub key [13].

In distributed approaches, the group key is generated in a contributory fashion, where all members contribute their own share in computing the group key. Steiner et al. [23] extended the Diffie Hellman protocol, which is the first pioneering two party key agreement protocols, to multi-party scenario. The Group Diffie Hellman key agreement protocol (GDH) require N rounds to agree up on a common session key for a group of N members. In particular, the number of rounds may be crucial in a large number of group members environment, because members can't communicate until the other members finish the foregoing round protocol.

Kim, Perrig, Tsudik [11] proposed a tree based group key agreement protocol called TGDH. TGDH computes a group key derived from contribution of all group members using a binary tree. Compared to GDH, TGDH just needs constant rounds to agree up on a common group key and has some computational advantages. Now a days a number of tree based group key agreement protocols have been proposed based on TGDH, including those in [10, 20, 24, and 27]. [24] Introduces first time a ternary tree approach and uses GDH.2 [23] as the basic operation to establish a contributory group key among the group members. This results significant reduction in height of the tree and reduces the computation and communication overhead. But the main drawback of this approach is that it support only for the no. of group members in the form of 3k where k is any integer. Moreover this protocol did not specify how to manage the resultant members after a single join or leave operation when the no. of group members essentially becomes other than 3^k.

3. PRELIMINARIES

Since the proposed paper based on ternary tree approach with ECC based GDH as its basic operation, the ECC and ECDH techniques are described in this section. The use of ternary tree instead of conventional binary tree has been already justified in [24].

3.1 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) was discovered in 1985 by Victor Miller(IBM) and Neil Koblitz (University of Washington) as an alternative mechanism for implementing public-key cryptography. The primary benefit promised by ECC is a smaller key size, reducing storage and transmission requirements i.e., an elliptic curve system could provide the same level of security afforded by an RSA-based system with a large modulus and correspondingly larger key e.g., a 256-bit ECC public key should provide comparable security to a 3072-bit RSA public key.

In ECC non-singular type of Elliptic curves over the real number are used. The elliptic curve over real numbers takes the general form as:

$$y^2 = x^3 + ax + b$$

In cryptography, variables and coefficients of elliptic curve equation are restricted to elements in a finite field. Thus for above equation x, y are co-ordinates of GF(p), and a, b are integer modulo p, satisfying $4a^3 + 27b^2 \neq 0 \ (mod\ p)$ (for non singular elliptic curve).Where p is a modular prime integer which make the EC of finite field. An elliptic curve E over GF(p) consist of points (x, y) defined by above two equations, along with an additional point called O (point at infinity or zero point) in EC. The 'O' point plays the role of identity element for EC group.

Usually an elliptic curve is defined over two types of finite fields: the prime field F_p containing p elements (prime curve) and the characteristic 2 finite field containing 2^m elements (binary curve).This paper focuses on the prime finite field as the prime curve are best suit for software applications [22].

Elliptic Curve Arithmetic

Cryptographic schemes based on ECC rely on scalar multiplication of elliptic curve points. Given an integer k and a point P ∈ $E(F_p)$, scalar multiplication is the process of adding P to itself k times. The result of this scalar multiplication is denoted k × P or kP.

Points addition and point doubling form the basis to calculate EC scalar multiplication efficiently using the addition rule together with the double-and-add algorithm or one of its variants. The detail description of ECC (including its point addition rule) can be found in various papers including [12, 20, 26].

The security of ECC based protocols are based on intractability of Elliptic Curve Discrete Logarithm Problem(ECDLP). ECDLP state that: Given P, Q ∈ E , find an integer k ∈ Z^*p such that Q = kP. It is relatively easy to calculate Q given k and P, but it is relatively hard to determine k given Q and P.

3.2 Two Parties Elliptic Curve Diffie-Hellman Protocol

Similar to DLP-based Diffie-Hellman key exchange agreement, a key exchange between users A and B using Elliptic Curve Diffie-Hellman (ECDH) can be accomplished also discussed in [26] as follows:

1) *A* selects an integer $a_1 \in Z^*p$, this is *A*'s private key. *A* then generates a public key $X = a_1P$; the public key is a point in Ep(a, b).

2) *B* similarly selects a private key $a_2 \in Z^*p$, and computes a public key $Y = a_2P$.

3) *A* generates the secret key $K = a_1 Y$.

4) *B* generates the secret key $K = a_2 X$.

3.3 Three Parties Elliptic Curve Diffie-Hellman Protocol

Three parties Elliptic Curve Diffie-Hellman Protocol based on GDH.2 discussed in [24] implemented with elliptic curve for the group of three parties (*A*, *B* & *C*) as follows:

1) *A*, *B* and *C* chooses their own private keys a_1, a_2, $a_3 \in Z^*p$ respectively and keep it secret.

2) *A* calculate $X = a_1 P$ and send to *B*.

3) *B* calculates $Y_1 = a_2 P$; $Y_2 = a_2X$ and construct the set {X (as received from *A*), Y_1, Y_2 } which is then transmitted to *C*.

4) *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 broad cast remaining $\{Z_1,Z_2\}$ to the user *A* and *B*.

5) On receiving from *C* each member can calculates same group key as

A: $K = a_1Z_1$; and
B: $K = a_2Z_2$.

On completing all three members have a common point in the elliptic curve i.e. $K = a_3Y_2 = a_1Z_1 = a_2Z_2 = a_1a_2a_3P$. If this secret key is to be used as a session key, a single integer must be derived. There are two categories of derivation: reversible and irreversible [26]. If the session key is also required to be decoded as a point in elliptic curve, it is reversible. Otherwise, it is irreversible. The reversible derivation will result in a session key which doubles the length of the private key. In the irreversible derivation, we can simply use the X-coordinate or simple hash function of the X-coordinate as the session key [26].

4. PROPOSED PROTOCOL

The proposed protocol chooses a k-bit prime p and determine following public parameters:
{Fp, E/Fp, G, P}.
where E/Fp: Elliptic curve over Fp.
G: Cyclic additive points group formed by points on E/Fp.
P: Generator of G.
The protocol describes operation to generate common Session key among n members (it is not important whether n is equal to 3^k or not) called Initialization operation along with others group operations like Join, Leave, Merge, etc. for dynamic group.

4.1 Initialization

Let us suppose all members identified by $M_1, M_2, ..., M_n$ are arranged as the leaf nodes of a ternary tree. Now each member Mi randomly chooses a secrete $a_i \in Z_p^*$ (for i =1to n) and keep it safe. The sequence of operations in each round are follows.

1) In first round all members are arranged in $\lfloor n/3 \rfloor$ subgroups having set of three members in each. (If n is not the multiple of 3 then remaining one or two members supposed to forward in next round and they does nothing in current round. The same condition is in every round) Member in every set form their own common EC points by using ECC based Three Parties Diffie Hellman key exchange as discussed in section 3.3.

   At the end of first round every subgroup has its own secret key (a point in EC group) in the form of $(a_{xi}.a_{yi}.a_{zi}.P)$ for i=1... $\lfloor n/3 \rfloor$. Where a_{xi}, a_{yi} & a_{zi} are private keys of first ,second and third member of i'th subgroup.

   One member from every subgroup comes forward as the group controller (GC) for the next round. In this way we treat every subgroup as a new node controlled by their GC.

2) In second round There are total $\left\lceil \frac{n}{3} \right\rceil$ nodes (along with the remaining node coming from previous round) form the subgroups having set of three participants of each and calculates their secrete subgroup key as in previous. This time GCs uses x-co-ordinate of their own subgroup keys as the private key. GC_1 calculate $(x_1.P)$ and unicast to GC_2. GC_2 calculates $(x_2.P)$ and $(x_2.\ x_1.P)$ and broadcast $\{(x_1.P)$,$(\ x_2.P)$,$(\ x_2.x_1.P)\}$ to the all members of third subgroup. The members of third subgroup now can calculate common key as $(x_3.\ x_2.x_1.P)$ and keep it secret .GC_3 additionally calculates $\{(x_3.x_1.P), (x_3.x_2.P)\}$ and broadcast to the all members of its sibling groups. All sibling subgroup members calculates common key by multiplying their own private value.

   Note that GC_1, GC_2 and GC_3 are group controllers and x_1, x_2 and x_3 are their x co-ordinates of common points of first, second and third subgroups respectively.

3) Repeats the above process in subsequent rounds .In every round no. of nodes becomes (1/3) of the previous round. After $\lceil log_3^n \rceil$ rounds we have a single group which includes all the members, each sharing the group secret key.

4) If in last round the no of participants remains only two then instead of three parties it employs two parties ECC based Diffie Hellman for calculating final group key.

An example to initialize 12 ($12 \neq 3^k$) members by above protocols is illustrated in Figure 1. Members of the group are represented by leaf node $M_1,..,M_{12}$. In First round four groups are formed with their own group key shown as the parent nodes $G_1 \ldots G_4$ of corresponding members. Only one group G_5 is possible in second round with participants G_1, G_2 and G_3 they form a common key which is shared by all the members of G_1, G_2 and G_3. Group G_4 doing nothing in this round forwarded to the next round. In third round remain only two participants G4 and G_5 .They employ two parties ECC based Diffie Hellman as suggested in our protocol to compute final group key which is shared by all the members $M_1,..,M_{12}$.
The worst case cost of initialization operation are summarised in following table:

Table1. The Worst case cost of Initialization Operation

Communication:		**Computation:**
Rounds	**Messages (Unicast + Broadcast)**	**Point Multiplications:**
$h = \lceil log_3^n \rceil$	$\left\lceil 3\frac{(n-1)}{2} \right\rceil$	$\frac{5(n-1)}{2} + h.n$

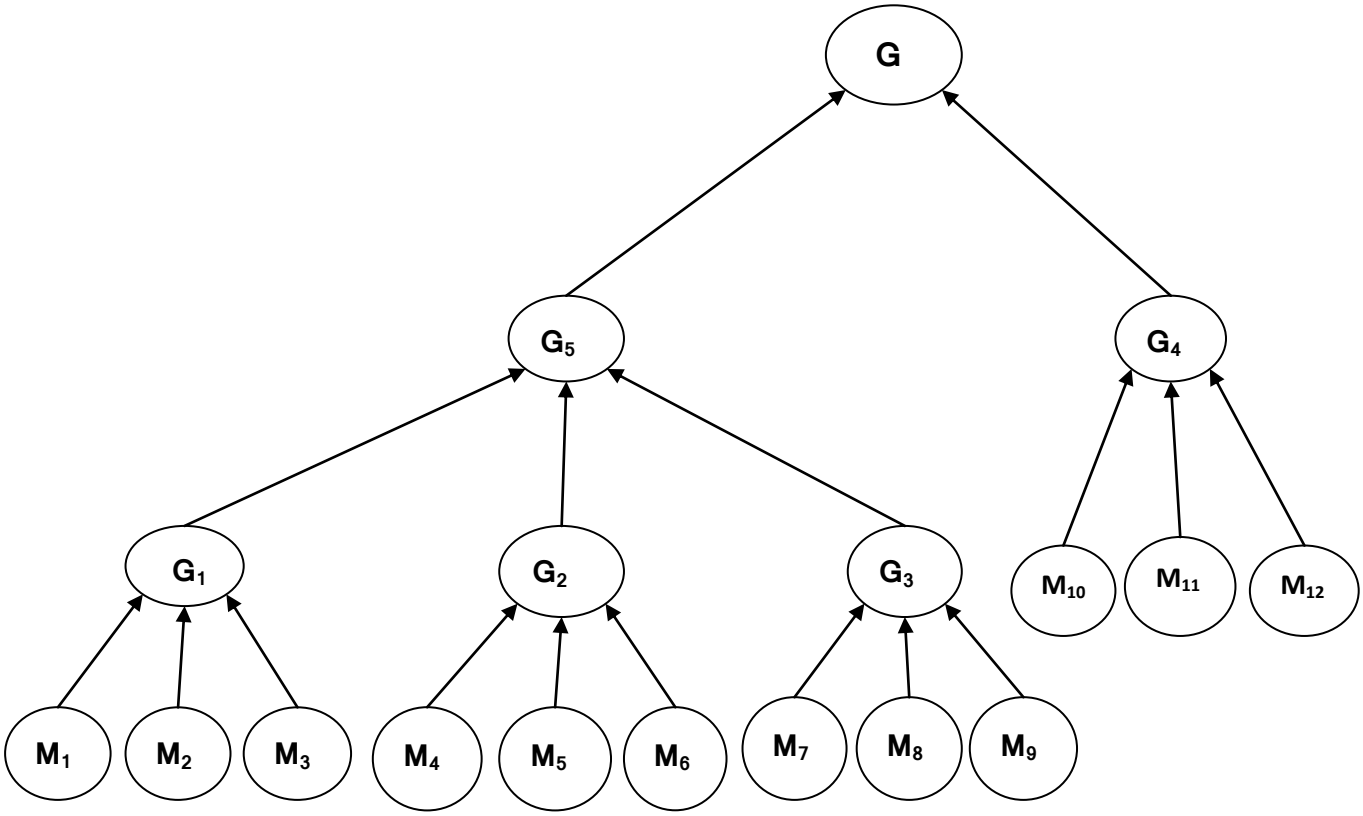

Figure 1: Initialization of 12 Members

4.2 Join Operation:

Efficient Join and Leave algorithms are very important to the dynamic group key agreement protocols, since any member can leave and join the group at any time. The proposed protocol handles the join operation in two ways: single join when only one new member wants to join the group and mass join when multiple new members want to join the existing group. When a join request is come to the GC it wait till the predefined thresholds (a small time slot) if more join requests are come within the threshold then these requests are handled by mass join operation. On the other hand in case of single request it is handled by single join operation.

4.2.1 Single join

For single join the GC of initial group do the two parties Diffie Hellman Key exchange with the new member. So there are only two messages are required for single join. Initially joining member choose a secrete value anew ∈ Zp* and calculate blind key as (anew.P) and broadcast to the members of current group. All members of current group (including GC) can calculates new group key Knew = (x.anew.P) and keep it secret .Now GC of current group send (x.P) to the new member and then the new member can calculates Knew = (anew.x.P). Note that 'x' is the group secret of existing group. The total no. of point multiplications require in single join is: (n+2).

4.2.2 Mass Join

Suppose there are 'm' new members want to join the current group with 'N' members having common group key K then:

a) First initialize 'm' members to form their own independent group by initialization operation discussed in section 4.1 and form their own group key say K_{new} .

b) Merge the new group to the old existing group.

c) To merge these two groups, sponsor(GC) of each group broadcasts their blind key to the all members of other group and form a new group key ($x_1.x_2.P$) just like as two parties key exchange. Where x_1 , x_2 are group secrets of old and new group respectively.

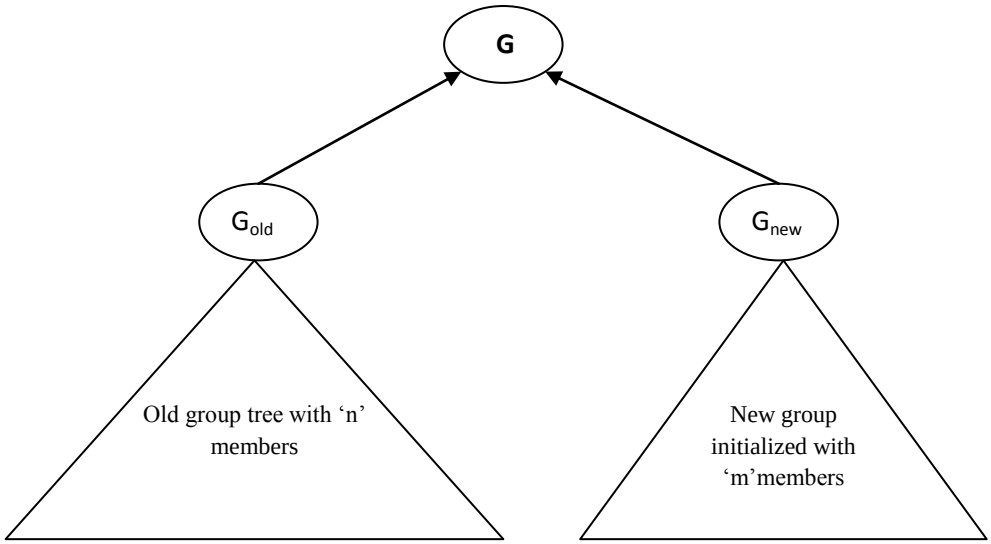

Figure 1 Mass Join

The total no of messages involved in mass join operation are:

$= \left\lceil 3(\frac{m-1}{2}) \right\rceil + 2.$

Total no. of point multiplication can be calculated as:

$$\frac{5(n-1)}{2} + h.n + \frac{5(m-1)}{2} + h'.m + n + m + 2$$
$$= n\left(h + \frac{7}{2}\right) + m\left(h' + \frac{7}{2}\right) - 3$$

Where h and h' are heights of initial and new group tree respectively. Note that the discussion of point multiplication cost for all operations of in this paper is worst case cost in practical scenario it is much less than the worst case.

Note that although we are using merge operation of CCEGK for mass join in current paper but we can use any method like mass join-iterative [1], mass join-simultaneous, etc. as discussed in [27].

4.3 Leave Operation

If no. of leaving member is one or very few then it can be handle few times calling of single leave operation which require just one broadcast message. But if the no. of leaving members are very large then Mass Leave operation is perform which is suggest that it is better to reinitialize the tree with other than leaving members.

4.3.1 Single Leave:

In case of single leave, key path from the leaving member to the root of the tree must be updated. If the leaving member is group controller (GC) then choose some other member as the group controller but no need to change other subgroup controller.Leave operation in proposed protocol is very similar to that in TGDH [11] .Suppose that we have n members in the group and assume that member Md leaves the group. First, each member updates its key tree by deleting the leaf node corresponding to Md. If Md have only one sibling, the former sibling of Md is promoted to replace Md's parent node. Leave operation in proposed protocol is very similar to that in TGDH [11] the sponsor generates a new key share, computes all {key & blind key contribution with others siblings} on the key-path up to the root, and broadcasts the Messages required to initialization of 'm' members + 2

new set of blind key contributions. This allows all members to compute the new group key. So there is only one round and only one message is require in leave operation. The no. of point multiplication is depends on the location of the leaving member and tree structure. In worst case it is (n + $3h$ -2) when the leaving node locates in deepest level. Note that the sponsor in this case is the rightmost leaf node of the sub tree rooted at the leaving member's sibling node.

4.3.2 Mass Leave:

If the no. of leaving members are very large then it is better to reinitialize the tree with other than the leaving members. So if n be the total no. of members in initial group and m is the no. of leaving members from current group then

The message cost of the mass leave operation is: $\left\lceil 3\frac{(n-m-1)}{2} \right\rceil$

Total no. of point multiplication is: $\frac{5(n-m-1)}{2} + h''(n-m)$

Note that after numerous group operations like join, leave, mass join etc. the resultant key tree becomes quit unbalanced. So as suggested in [27] when the key tree reaches a certain imbalance point, we should rebalance the tree and treat the rebalance operation as the group operation required. The imbalance point can vary depending on network and efficiency requirement. The proposed paper does not provide any new rebalancing scheme however protocol can use any scheme discussed in [27]. Also sometimes it is more suitable to reinitialize the tree instead of rebalancing.

5. AUTHENTICATION

Although the proposed protocol is based on elliptic curve Diffie Hellman which is secure and not easy to break, the entire system is vulnerable if the keys are not securely distributed. Therefore, we should implement an authentication algorithm in the protocol. There are several ways to authenticate a group key exchange, such as centralized authentication, implicit authentication, and pairwise authentication [3, 4, 9, 11, 16]. In all operations of the protocol any of the above authentication schemes can be used. The advantages and limitations of different authentication schemes are discussed in CCEGK [27].

6. SECURITY ANALYSIS

The Security of ECC is due to the discrete logarithm problem over the points on the elliptic curve. Cryptanalysis involves determining x given Q and P where P is a point on the EC and Q = x P that is P added to itself x times. The best known algorithm to break the elliptic curve points is the pollard – rho algorithm which is a fully exponential algorithm and difficult to solve. Forward and Backward secrecy are maintained as each session. Also the proposed protocol is analyzed by AVISPA tool and it is found to be safe against the various attacks. The detail output of AVISPA tool and role specification in HLPSL are shown in next section. In this section we address the possible types of attacks. Security tolerance of the proposed scheme in response to the various attacks is discussed in the following subsections.

Exterior Collecting Attack
The first potential attack is from an outsider. If an attacker is outsider, it means no idea about what EC or base point is being used is known and hence more difficult to attack. Therefore, the proposed scheme restricts intrusion from outsiders.

Interior Collecting Attacks
If a group member has many ancestors and if it negotiates with one parent also by knowing the key as there is no relation parameter among any of the ancestor nodes it is not possible to obtain the key.

Known Session Key Security In each session, each member Mi randomly chooses an private key ai ϵ Zp* and the generated group session key depends on each member's private key ai . The adversary that compromises one session key should not compromise other session keys, so this protocol can provide known session key security.

No Key Control The group session key in the protocol is determined by all members' private keys a_i , for i=1,2,3,……n so that neither party alone can control the outcome of the session key. So it is fully contributory technique.

Forward and Backward Secrecy The new coming member does not know what was the group key earlier because they receives only a point in EC which is generating point multiplied with the group secrete. New member cannot get the secret due to ECDLP. Also the leaving member cannot compute the new group key because all blind key contribution from leaving member to the root are updated by leave controller and its share is no longer part of the new group key.

7. FORMAL SECURITY VERIFICATION USING AVISPA TOOL

Recently, AVISPA tool [25] is widely used by many researchers [15] for the automated validation of Internet security protocols and applications. The AVISPA is a pushbutton tool designed by University of Geneva, Italy using the concept of Dolev and Yao intruder model [7], where the network is controlled by an intruder (Active and passive); however he is not allowed to crack the underlying cryptography. The AVISPA tool supports High Level Protocol Specification Language (HLPSL) based on which the cryptographic protocols are to be implemented and analyzed. It has four model checkers/back-ends, called OFMC (On-the-fly Model-Checker), CL-AtSe (Constraint-Logic-based Attack Searcher), SATMC (SAT-based Model-Checker) and TA4SP (Tree Automata-based Protocol Analyzer). The details description about AVISPA and HLPSL can be found in [2]. The role specification of each party involved in key generation just for three parties and the results of OFMC & CL-AtSe are shown below:

Role Specification:

```
role party1 ( P1,P2,P3:agent,G: text, SND, RCV :
channel(dy),K1,K2,K3: public_key)
played_by P1
def= local
State : nat,
A1 : text,
X,Y,Z,Ya3,Za2,Xa2,Xa3 :public_key,
K : symmetric_key
% knowledge(P1) = {inv(K1)}
init State :=0
transition

1.State=0/\RCV(start)=|>

    State' := 1/\ A1':= new()
  /\ X':=exp(G,A1')
  /\SND({{P1.X'}_inv(K1)}_K2)
  /\ witness(P1,P2,party2_party1_x,X')
2.State=1 /\ RCV({{Xa3'.Ya3'}_inv(K3)}_K1)=|>
State':= 2
/\request(P1,P3,party1_party3_ya3,Ya3')
/\ K':= exp(Ya3',A1)
/\secret(K',k,{P1,P2,P3})
end role

role party2(P2,P3,P1: agent, G : text,SND,RCV:
channel (dy),K1,K2,K3 : public_key)
played_by P2 def=
local
State :nat,
X,Y,Z,Ya3,Za2,Xa2,Xa3 :public_key,
K : symmetric_key,
A2 :text
init State :=0
% knowledge(P2) = {inv(K2)}
Transition
1.State =0 /\RCV({{P1.X'}_inv(K1)}_K2)=|>
State':=1/\request(P2,p1, party2_party1_x,X')
/\ A2':= new()
/\ Y':= exp(G,A2')
/\ Xa2' := exp(X',A2')
/\ SND({{P1.P2.X'.Y'.Xa2'}_inv(K2)}_K3)
/\ witness(P2,P3,party3_party2_x,X')
/\ witness(P2,P3,party3_party2_y,Y')
/\ witness(P2,P3,party3_party2_xa2,Xa2')
2.State =1 /\ RCV({{Xa3'.Ya3'}_inv(k3)}_K2)=|>
State':=2
/\ request(P2,P3,party2_party3_xa3,Xa3')
/\ K' := exp(Xa3',A2)
/\ secret(K',k,{P1,P2,P3})
end role
```

```
role party3(P3,P1,P2: agent,G:text,SND,RCV: channel(dy),K1,K2,K3 : public_key)
played_by P3 def=
local
State :nat,
X,Y,Z,Ya3,Za2,Xa2,Xa3 :public_key,
K :symmetric_key,
A3 : text
init State := 0
transition
% knowledge(P3) = {inv(K3)}
1.State = 0 /\
RCV({{P1.P2.X'.Y'.Xa2'}_inv(K2)}_K3)=|>
State':= 1/\ request(P3,P2,party3_party2_x,X')
/\ request(P3,P2,party3_party2_y,Y')
/\ request(P3,P2,party3_party2_xa2,Xa2')
/\ A3' := new()
/\ Z':=exp(G,A3')
/\ K' := exp(Xa2',A3')
/\ secret(K',k,{P1,P2,P3})
/\ Xa3' := exp(X',A3')
/\ Ya3' := exp(Y',A3')
/\ SND({{Xa3'.Ya3'}_inv(K3)}_K1)
/\ SND({{Xa3'.Ya3'}_inv(K3)}_K2)
 %.{P1.P2.P3.X'.Y'.Z'}_K
/\ witness(P3,P1,party1_party3_ya3,Ya3')
/\ witness(P3,p2,party2_party3_xa3,Xa3')

end role
```

Simulation result of our scheme on OFMC model checker:

% OFMC

% Version of 2006/02/13

SUMMARY

SAFE

DETAILS

BOUNDED_NUMBER_OF_SESSIONS

PROTOCOL

/home/avispa/web-interface-computation/./tempdir/workfileufnO9q.if

GOAL

as_specified

BACKEND

OFMC

COMMENTS

STATISTICS

parseTime: 0.00s

searchTime: 0.14s

visitedNodes: 16 nodes

depth: 4 plies

Simulation result of our scheme on CL-AtSe model checker:

SUMMARY

SAFE

DETAILS

BOUNDED_NUMBER_OF_SESSIONS

TYPED_MODEL

PROTOCOL

/home/avispa/web-interface-computation/./tempdir/workfileufnO9q.if

GOAL

As Specified

BACKEND

CL-AtSe

STATISTICS

Analysed : 9 states

Reachable : 0 states

Translation: 0.04 seconds

Computation: 0.00 seconds

8. COMPARISON WITH OTHER EXISTING PROTOCOLS

This section compares the costs of major group key management operations of proposed technique with other existing binary tree based group key technique .The comparison chart shown in Table 2 uses the following notation for comparison.

n: Number of members in a group.

m: Number of new member send requests to join/leave the original group.

h = $\lceil log_3^n \rceil$, which is the height of original key tree in proposed technique.

$h' = \lceil log_3^m \rceil$, which is the height of new key tree created by m new members in proposed technique.

$h'' = \left\lceil log_3^{(n-m)} \right\rceil$ Which is the height of new key tree with$(n-m)$ members in proposed technique after mass leave event.

$H'' = \left\lceil log_2^{(n-m)} \right\rceil$ Which is the height of new key tree with$(n-m)$ members in other existing binary tree based technique.

It is noted that Maria et al. [20] and TGECDH in

[26] do not provides the cost of initialization operation in their papers. So we first calculate no. of messages and point multiplications of these protocols based on the concepts discussed in their papers and compare with proposed protocols. Same for other operations in Maria et al.

Table2. Comparison Table

Protocol	Group Operation	Rounds	Messages	Point Multiplications
Proposed	Initialization	$h = \lceil log_3^n \rceil$	$\left\lceil 3(\frac{n-1}{2}) \right\rceil$	$\frac{5(n-1)}{2} + h.n$
	Join	1	2	n +2
	Mass Join	$h' + 1$	$\left\lceil 3\frac{(2m+1)}{2} \right\rceil$	$n\left(h + \frac{7}{2}\right) + m\left(h' + \frac{7}{2}\right) - 3$
	Leave	1	1	$n + 3h - 2$
	Mass Leave	$h'' = \left\lceil log_3^{(n-m)} \right\rceil$	$\left\lceil 3\frac{(n-m-1)}{2} \right\rceil$	$\frac{5(n-m-1)}{2} + h''(n-m)$
TGECDH [26]	Initialization	$H = \lceil log_2^n \rceil$	$2(n-1)$	$n(n-1) + nH$
	Join	2	3	$(n+3)$
	Leave	1	1	$2H + n - 3$
	Mass Leave	$H'' = \left\lceil log_2^{(n-m)} \right\rceil$	$2(n-m-1)$	$(n-m)(n-m-1) + (n-m)H''$
Maria et al[20]	Initialization	$H = \lceil log_2^n \rceil$	$2(n-1)$	$n(n-1) + nH$
	Join	1	2	$(n+3)$
	Leave	1	1	$2H + n - 3$
	Mass Leave	$H'' = \left\lceil log_2^{(n-m)} \right\rceil$	$2(n-m-1)$	$(n-m)(n-m-1) + (n-m)H''$

9. CONCLUSON

This paper proposed an efficient contributory group key agreement protocol for dynamic group. Here group is organized in a logical ternary key tree instead of binary tree. For key computation it uses ECC based three parties Diffie-Hellman based on GDH.2. The paper describes the implementation of major group key management operations. The security verification of the protocol is done by using AVISPA tool and found to be safe under various attacks. Finally The cost of the various group operations are compared with others existing techniques which shows its efficient performance than the existing protocols.

10. REFERENCES

[1] Alves-Foss, J. An e_cient secure authenticated group key exchange algorithm for large and dynamic groups. In IN PROC. 23 RD NATIONAL INFORMATION SYSTEMS SECURITY CONFERENCE (2000), pp. 254-266.

[2] Armando, A., Basin, D., Boichut, Y., Chevalier, Y., Compagna, L., Cuellar, J., Hankes Drielsma, P., He_am, P.-C., Mantovani, J., Modersheim, S., von Oheimb, D., Rusinowitch, M., Santiago, J., Turuani, M., Vigan_o, L., and Vigneron, L. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In Proceedings of the 17th International Conference on Computer Aided Veri_cation (CAV'05), K. Etessami and S. K. Rajamani, Eds., vol. 3576 of LNCS. Springer, 2005. Available at http://www.avispa-project.org/publications.html.

[3] Ateniese, G., Steiner, M., and Tsudik, G. Authenticated group key agreement and friends. In Proceedings of the 5th ACM conference on Computer and communications security (New York, NY, USA, 1998), CCS '98, ACM, pp. 17-26.

[4] Burmester, M., and Desmedt, Y. E_cient and secure conference-key distribution. In Proceedings of the International Workshop on Security Protocols (London, UK, UK, 1997), Springer-Verlag, pp. 119-129.

[5] Challal, Y., and Seba, H. Group key management protocols: A novel taxonomy.

[6] Ching Chan, K., and h. Gary Chan, S. Key management approaches to offer data confidentiality for secure multicast. IEEE Netw (2003), 30-39.

[7] Dolev, D., and Yao, A. C. On the security of public key protocols. Information Theory, IEEE Transactions on 29,2 (1983), 198-208.

[8] Jun, Z., Yu, Z., Fanyuan, M., Dawu, G., and Yingcai, B. An extension of secure group communicationusing key graph. Information Sciences 176, 20 (2006), 3060-3078.

[9] Just, M., and Vaudenay, S. Authenticated multi-party key agreement. In Advances in Cryptology – ASIACRYPT '96, International Conference on the Theory and Applications of Cryptology and Information Security, Kyongju, Korea, November 3-7, 1996, Proceedings (1996), vol. 1163 of Lecture Notes in Computer Science, Springer, pp. 36-49.

[10] Kim, Y., Perrig, A., and Tsudik, G. Group key agreement efficient in communication. IEEE Transactions on Computers 53, 7 (2004), 905-921.

[11] Kim, Y., Perrig, A., and Tsudik, G. Tree-based group key agreement. ACM Trans. Inf. Syst. Secur. 7, 1 (Feb.2004), 60-96.

[12] Koblitz, N. Elliptic curve cryptosystems. Mathematics of Computation 48, 177 (Jan. 1987), 203-209.

[13] Mittra, S. Iolus: A framework for scalable secure multicasting. pp. 277-288.

[14] Ng, W. H. D., Howarth, M., Sun, Z., and Cruickshank, H. Dynamic balanced key tree management for secure multicast communications. IEEE Trans. Comput. 56, 5(May 2007), 590-605.

[15] Nicanfar, H., and Leung, V. C. M. Multilayer consensus ecc-based password authenticated key-exchange (mcepak) protocol for smart grid system. IEEE Trans.Smart Grid 4, 1 (2013), 253-264.

[16] Perrig, A. Efficient collaborative key management protocols for secure autonomous group communication. In In International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC 99 (1999), pp. 192-202.

[17] Peyravian, M., Matyas, S., and Zunic, N. Decentralizedgroup key management for secure multicast communications. Computer Communications 22,13(1999), 1183 -1187.

[18] Rafaeli, S., and Hutchison, D. Hydra: a decentralisedgroup key management. In Enabling Technologies: Infrastructure for Collaborative Enterprises, 2002. WET ICE 2002. Proceedings. Eleventh IEEE International Workshops on(2002),pp.62-67.

[19] Rafaeli, S., and Hutchison, D. A survey of key management for secure group communication. ACM Comput.Surv. 35, 3 (Sept. 2003), 309-329.

[20] S. Maria Celestin Vigila, K. M. Ecc based contributory group key computation scheme using one time pad.JOURNAL OF COMPUTING.

[21] Setia, S., Zhu, S., and Jajodia, S. A scalable and reliable key distribution protocol for multicast group rekeying,2002.

[22] Stallings, W. Cryptography and Network Security: Principles and Practice, 3rd ed. Pearson Education, 2002.

[23] Steiner, M., sudik, G., and Waidner, M. Diffie-Hellman key distribution extended to group communication. ACM Press, pp. 31-37.

[24] Tripathi, S., and Biswas, G. P. Design of efficient ternary tree based group key agreement protocol for dynamic groups. In Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International (2009), pp. 1-6.

[25] Vigan, L. Automated security protocol analysis with the avispa tool. In In Proceedings of MFPS05 (2006), p. 2006.

[26] Wang, Y., Ramamurthy, B., and Zou, X. The performance of elliptic curve based group diffe-hellman protocols for secure group communication over ad hoc networks. In Communications, 2006. ICC '06. IEEE International Conference on (2006), vol. 5, pp. 2243-2248.

[27] Zheng, S., Manz, D., and Alves-Foss, J. A communication-computation efficient group key algorithm for large and dynamic groups. Comput. Netw. 51,1(Jan. 2007), 69-93.