ECDH-based Scalable Distributed Key Management Scheme for Secure Group Communication

Qiuna Niu
College of Information Science & Technology, Qingdao University of Science & Technology, Qingdao, China 266061
niuqiuna@eyou.com

Abstract—With the popularity of group-oriented applications, secure and efficient communication among all group members has become a major issue. An efficient key management mechanism is the base and critical technology of secure group communications. A distributed group-oriented key management scheme without the participation of third parties is proposed in the paper. The scheme deploys Elliptic Curve Diffie-Hellman (ECDH) which is more lightweight compared to regular Diffie-Hellman. The approach includes group key establishment and rekeying algorithms when there are membership changes. By using a distributed architecture, the load of key management is reduced. Specifically, the scheme can be extended to hybrid architecture to provide better scalability. Consequently, the extended scheme is both fault-tolerant and efficient in terms of integrity and confidentiality. In all protocol suites, the shared group key is calculated by scalar multiplication. According to performance comparisons with other schemes, the proposed scheme dramatically reduces communication overhead and computational costs. Security analysis indicates that the proposal provides a number of desirable security properties, including group key secrecy, forward secrecy and backward secrecy.

Index Terms—Distributed key management, group communication, ECDH

I. INTRODUCTION

As the development of wireless and wired networks technologies, it is becoming increasingly necessary to design and implement large scale networks to support group-oriented applications such as content distribution, multiparty teleconferencing or videoconferencing. Multicast group communication is preferably for sending the messages to a specific group of members. Because a host can receive transmitted data from any multicast groups, security is much more important especially for distributed and collaborative networks over insecure surroundings such as the Internet. To ensure the secrecy and integrity of the transmitted data, there is an urgent need to find a proper and efficient key management scheme.

Considering the performance overhead in a secure group communication, it is more popular by sharing secret keys in a group rather than directly distributing peer-to-peer private/public keys through key distribution center. How to manage and maintain these secret keys is called group key management [1].

Group key management algorithms can be divided into two types, one is centralized group key management, and the other one is distributed group key management. Both approaches have their own advantages and disadvantages, the centralized approach has the advantages of efficiency of the symmetric key encryption/decryption, but it also suffers from the fact that servers need great computation power, large communication bandwidth, and considerable storage. The distributed approach allowing every participant to take part in the interactive computation of the group key, it distributes the load of the key management to all the group members, so it is fault-tolerant in terms of integrity and confidentiality. Concerning the above issues, the distributed approach is more attractive.

The rest of the paper is as follows, Section II presents the related works about group key management. Section III introduces the background. Section IV presents the proposed group-oriented key management schemes. Section V evaluates the performance and prove the security. Finally Section VI concludes the paper.

II. RELATED WORKS

Two-party Diffie-Hellman (DH) key agreement [2] is the basis for great number of subsequent cryptographic schemes. Most distributed group key establishment protocols are based on generalizations of two-party DH or its extensions [3-8].

Group Diffie-Hellman (GDH) consists of GDH.1, GDH.2 (included in CLIQUES [3]) and GDH.3, they occupies large memory especially with the increase of group members. Y. Kim et al. Reference [4] proposed Tree-based Group DH (TGDH) protocol. TGDH has reduced the modular exponentiation from \(O(n)\) to \(O(\log n)\) during initial establishment, but it requires a few rounds for key computation, moreover each member maintains a set of keys arranged in a hierarchical binary tree.
Reference [5] utilizing Elliptic Curve Cryptosystem (ECC), optimizes CLIQUES and TGDH etc. from the perspective of ad hoc networks. The modified versions are called μCLQUIES and μTGDH, which decrease redundancies of the original protocols. Reference [6] presented an authenticated group key establishment protocol, it arranged users into binary tree structure. But in the protocol the shared key did not depend on the contribution of all members. Reference [7] proposed a two-party authenticated key agreement protocol for mobile ad hoc networks, but the scheme is based on the mix of ECC and RSA, which require more computational overhead. Reference [8] provides a hybrid group key management protocol based on TECH. The scheme is suitable for a military scenario, allows switching between centralized and distributed methods. It reduces communication costs and computational overheads.

To ensure secure group communication in specified environments, many distributed approaches have been proposed [9-15]. A number of schemes are proposed to meet the needs of different characteristics of mobile ad hoc group networks in [9-12]. Reference [13-15] present group key agreement protocols for wireless sensors networks.

An interesting overview over this family of cryptographic schemes can be found in [16]. The survey evaluates a number of the constant round group key agreement protocols proposed so far, the evaluation of efficiency and energy consumption of the protocols are briefly presented.

III. BACKGROUND

A. Motivation

All distributed group key agreement protocols apply public key cryptography, since there are no other methods to distribute a shared key over a public channel. Public key computational techniques including DH key exponentiation are expensive. Especially when distributing additional shared keys to users that have constrained bandwidth or capabilities of computation and storage, the computational overhead of management schemes should be taken into consideration.

According to [8], ECC keys have much fewer bits than applications based on the integer factorization problem (IFP) and the discrete logarithm problem (DLP), which leads to greater increase in computational cost. This also causes a large difference in the running times between ECC and other conventional cryptosystem such as RSA and DH. This makes ECC more attractive particularly for constrained environments. Table I compares the recommended key size between ECC and other classical asymmetric cryptosystem such as RSA and DH.

Consequently, the paper replaces DH with Elliptic Curve Diffie-Hellman (ECDH), which is more lightweight as compared to regular DH. Our approach also increases the numbers of participators to multi-parties, and provides a flexible group key management solution for dynamic groups.

<table>
<thead>
<tr>
<th>Security(Bits)</th>
<th>RSA/DH key length</th>
<th>ECC key length</th>
<th>MIPS year to attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>1024</td>
<td>160-223</td>
<td>10⁻²</td>
</tr>
<tr>
<td>112</td>
<td>2048</td>
<td>224-255</td>
<td>10⁻³</td>
</tr>
<tr>
<td>128</td>
<td>3072</td>
<td>256-383</td>
<td>10⁻⁴</td>
</tr>
<tr>
<td>192</td>
<td>7860</td>
<td>384-511</td>
<td>10⁻⁷</td>
</tr>
</tbody>
</table>

Considering a hybrid of centralized and distributed scheme may be required in secure group with many members, the proposed scheme is scalable, when the group members are relatively less, the distributed protocol can be used, and otherwise the extended hybrid protocol can be implemented.

B. ECDLP

The security of Elliptic Curve Cryptosystem (ECC) relies on the difficulty of solving the ECDLP (Elliptic Curve Discrete Logarithm Problem) on the EC group. Solving the ECDLP is harder than solving both, the IFP, as well as the DLP.

The equation of ECC over a finite Field Fp, denoted as E(Fp), is as follows:

\[ Y^2 = X^3 + AX + B \text{ (mod } p) \]  

where \( p > 3 \), is an odd prime, and \( A, B \in F_p \), satisfy

\[ 4A^3 + 27B^2 \neq 0 \text{ (mod } p) \]  

Fix an elliptic curve \( E(F_p) \). All points \( (P, Q) \) which satisfy the above equation (1) and the point at infinity lies on the elliptic curve. Find a random integer \( x \) such that \( xP = Q \), where \( xP \) represents the point \( P \) on elliptic curve added to itself \( x \) times, and the operation is called a scalar-point multiplication. The ECDLP is to determine \( x \) given \( P \) and \( Q \). It is relatively easy to calculate \( Q \) given \( x \) and \( P \), but it is very hard to determine \( x \) given \( Q \) and \( P \). Choosing elliptic curves as cryptosystems, suppose that \( Q \) is the public key and \( x \) is the private key.

C. ECDH and the Extension

The Elliptic Curve Diffie-Hellman (ECDH) key agreement scheme is a variant of the DH key agreement protocol. Let point \( P \in E(F_p) \) of high prime order \( n \), A and B are two entities that wish to share a secret key. Both A and B generate a pair of public/private key \((k_AQ_A)\) and \((k_BQ_B)\) respectively, where \( Q_A = k_AP \), \( Q_B = k_BP \), \( k_A, k_B \in [1, n-1] \). Then A and B perform a two-party Diffie-Hellman key exchange and produce a shared session key as,

\[ k_{AB} = k_AQ_B = k_BQ_A \]  

The paper is based on the extension of ECDH to the multiparty case [10]. Assuming in a three-party case, group member A, B and C all holds key pairs \((k_AQ_A), (k_BQ_B)\) and \((k_CQ_C)\). According to scalar- points multiplication in prime field, every shared session key is
calculated through intermediate public contributions. In this three-party case, \( A \) calculates as follows:

\[
k_{A,B,C} = k_2, k_{A,B} = k_3(k_2Q_C) = k_4(k_3(k_2P))
\]

B computes the key through,

\[
k_{A,B,C} = k_2k_{A,B} = k_3(k_2Q_C) = k_4(k_3(k_2P))
\]

C calculates the key via,

\[
k_{A,B,C} = k_2k_{A,B} = k_3(k_2Q_C) = k_4(k_3(k_2P))
\]

Then \( k_{A,B,C} \) is the shared session key of \( A, B \) and \( C \). In similar ways, we can also perform the scalar-point multiplication in a \( n \)-party case, which is used in the group key establishment of the proposed scheme.

IV. PROPOSED ECDH-BASED GROUP KEY MANAGEMENT SCHEME

A. Group Key Establishment

The scheme supports two kinds of applications, distributed operations of a single group or extension to a hybrid architecture involving a few more subgroups. In the two cases, a new member initiates the joining operation to the target group. The proposed scheme is based on the extended CLIQUES key agreement algorithm. Notations used in this paper are listed in Table II.

<table>
<thead>
<tr>
<th>( E(F_p) )</th>
<th>Elliptic Curve Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>numbers of group communication</td>
</tr>
<tr>
<td>( m_i )</td>
<td>( i )-th group member; ( i \in [1, n] )</td>
</tr>
<tr>
<td>( P )</td>
<td>prime base point, ( P \in E(F_p) )</td>
</tr>
<tr>
<td>( s_i )</td>
<td>secret key of ( m_i ), a random integer</td>
</tr>
<tr>
<td>( xP )</td>
<td>public key of ( m_i ), calculated through the scalar multiplication operation</td>
</tr>
<tr>
<td>( m_i \to )</td>
<td>( m_i ) unicasts information to another member</td>
</tr>
<tr>
<td>( m_i \to # )</td>
<td>( m_i ) multicasts messages to all members in the same group</td>
</tr>
</tbody>
</table>

**Initiation:** Choosing elliptic curves as cryptosystems, both \( E(F_p) \) and \( P \) are shared by all group members. Logic sequence of nodes should be arranged when initialization. The last member \( m_n \) usually acts as the controller by default. Controller maintains a member list which includes the address and sequence of connecting nodes of its group.

Assume there are four members in the group \( m_1, m_2, m_3, \) and \( m_4 \). Each chooses a secret random integer \( x_i \) as their secret key denoted as \( x_1, x_2, x_3, x_4 \). Then they each computes public key which is \( x_1P, x_2P, x_3P, x_4P \) respectively.

**Mechanism for Key Setup:** Key setup establishes a shared common group key for secure group communications, the steps is as following.

i> \( m_1 \to m_2 \): information \( x_1P \)

ii> \( m_2 \) computes the intermediate public contribution \( x_2(x_1P) \) and ,

iii> \( m_2 \to m_3, x_1P, x_2P \) and \( x_2(x_1P) \)

iv> \( m_3 \to m_4 : x_3P, x_3P, x_3P, x_3P \) and \( x_3(x_2P) \) As a controller of the group, \( m_4 \) firstly computes three intermediate public contributions, \( x_3x_4(x_3P) \), \( x_3(x_3P) \), and \( x_4(x_2P) \), then,

v> \( m_4 \to \# : x_4(x_3P), x_4(x_3P) \) and \( x_4(x_4P) \)

vi> Each member \( m_i \) performs scalar-point multiplication through its secret key \( x_i \) to produce the final shared key \( k_{1,2,3,4} \) where,

\[
k_{1,2,3,4} = x_i(x_1(x_2(x_3(x_4)))) = x_i(x_i(x_i(x_i(x_i(x_i(x_i(x_i)))))))(8)
\]

**Mechanism for Rekeying:** In order to provide the forward secrecy and the backward secrecy, any change in the group membership will induce group key refreshing. Consequently, there are two cases under which the group key should be updated: when a new member JOINs the group; and when a member LEAVES the group.

- **Mechanism for member joining:** When a member joins the group, to provide backward security, updating the session key is very essential so that the new member can't access any of the past group communications. Let \( m_5 \) need join the group, it will act as a new group controller to generate a new key for the subgroup. The rekeying operation is as follows.

i> \( m_1 \to \# \): Connection Request piggybacking its public contribution \( x_1P \).

ii> Receiving the request \( m_4 \) changes its secret key to \( x_4^' \) and

iii> \( m_1 \to m_5 : \{ x_1P, x_2P, x_3(x_1P), x_3P, x_3(x_2P) \} \) \( x_3P \) and \( x_3(x_2P) \)

iv> \( m_5 \): selects its secret key \( x_5 \) operates elliptic curve scalar multiplication and,

v> \( m_3 \to \# : x_5(x_3P), x_3P, x_3P, x_3P \)

vi> Eventually, each member \( m_i \) receives the intermediate values and \( m_i \) produces a new shared key as \( k_{1,2,3,4} = x_i(x_1(x_2(x_3(x_4)))) = x_i(x_i(x_i(x_i(x_i(x_i(x_i(x_i))))))) \).

- **Mechanism for member leave:** In the case that a member leaves the group, the group key must be updated to preserve the forward secrecy. The
algorithm requires the leaving member firstly to send a *Quit Request* to the group controller. Then the controller deletes leaving member’s address information from the sequence list, decreases the members count number and performs rekeying scheme to update a new group key. Suppose \( m_3 \) is going to leave, \( m_3 \) informs the controller \( m_4 \) by a *Quit Request*. After changing its private key value to \( x_4' \) and computing the public value, \( m_4 \) multicasts the public value to all the remaining members. Each member performs rekeying by putting its contribution to public value and generates the new subgroup key.

i> \( m_4 \rightarrow m_4 \): *Quit Request*

ii> \( m_4 \): changes member list to \( m_1, m_2 \) and \( m_4 \) and varies its secret key to \( x_4' \), and,

iii> \( m_4 \rightarrow \ast : x_4 \ (x_2P), x_4' \ (x_1P) \) with the new member list.

iv> Eventually, \( m_1 \) and \( m_2 \) add their public key separately and produce a new shared session key \( x_4' \ (x_2(x_1P)) \).

On the other side, if it is a group controller \( m_n \) going to leave, it sends *Quit Request* with stored intermediate public contribution to the previous member \( m_{n-1} \) who will act as a new controller. Similar with the above rekeying procedure:

i> \( m_n \rightarrow m_{n-1} \): *Quit Request*, \( x_1P, x_2P \)

ii> \( m_{n-1} \): changes member list to \( m_1, m_2, m_3 \) and changes \( x_3' \)

iii> \( m_{n-1} \rightarrow \ast : x_3 \ (x_2P), x_3' \ (x_1P) \) with the new member list.

iv> \( m_1 \) and \( m_2 \) separately produce a new shared session key \( x_3'x_2' (x_1P) \).

**B. Extended ECDH-based Group Key Agreement Scheme**

When the number of a group member expands too large, it will increase the computational performance especially for the group controller. There is an extended strategy to the proposed scheme. Hence the scheme becomes hybrid architecture accordingly instead of distributed.

The maximum allowed number of members in a group should be limited, and the last member in the member list acts as a subgroup controller. Once an additional user requests to join a group, the requested controller will inform this user to initiate a new subgroup. That means a group is partitioned into two independent subgroups.

Subsequently the new joiner originates a new round of group key agreement. Moreover, controllers of subgroups also need to produce a globally common key shared by all of the group members. Take 11 members in a group as example, restrict 4 as maximum number of members in each subgroup, then there are 3 subgroups and consequently 3 sub-controllers. The subgroup structure is shown in Fig. 2.

After key agreement in a single subgroup, the globally shared session key agreement procedure between three sub-controllers is presented as Fig. 3, where the intermediate public contribution \( x_3' (x_2(x_1P)) \) is simplified as \( g_3P, x_3' (x_2(x_1P)) \) as \( g_2P \) and \( x_2' (x_1(x_0P)) \) as \( g_1P \). After producing the globally shared key, the controller of the entire group, let be \( m_{n-1} \) in this example, multicasts the necessary public contribution with the completed member lists to each authorized member. And the key tree of all members is given in Fig. 4. Just as shown in the figure, each authorized member owns the global group key \( x_3' (x_2(x_1P)) \).

To reduce the memory requirements of storing keys, each ordinary subgroup member only need store the global group key besides its own secret/public key pairs. The public key of each member in the subgroup still need to be reserved by the subgroup controller since any dynamic membership change needs the information to refresh the globally shared group key.
Let $m_{12}$ be going to join a subgroup, whose subcontroller is $m_{11}$. $m_{12}$ should act as the new sub-group controller. At first, $m_{12}$ multicasts a Connection Request piggybacking its public contribution $x_{12}P$. Receiving the request, $m_{11}$ changes its secret key to $x_{11}'$ and sends the stored intermediate public contribution to this new member. $m_{12}$ chooses its secret key $x_{12}$, as a sub-group controller, it recalculates the intermediate public contribution and multicasts the information to the members of the same subgroup. $m_{12}$ acts as a subcontroller simultaneously, it multicasts the needed intermediate public contributions to other sub-controllers, which revokes new session key updating algorithm between common members in other subgroups. Eventually, each member generates the common session key $x_{12}(x_{11}'(x_{10}(x_{9}...x_{2}(x_{1}P))))$.

Similarly, when rekeying in the subtracting case, the new subgroup controllers also changes its secret key and then recalculates and multicasts the intermediate public values to the remaining members.

V. DISCUSSION

A. Performance Evaluation

In this section, the goal of the analysis is to estimate the communication and computational costs for the most common three group events, i.e., set up, members join and leave. CLIQUES [3] and TGDH [4] have been known to be the most common two contributory group key managements. We also compare our protocol to the protocols.

To evaluate the communication overhead, two metrics are used: the number of unicasted/multicasted messages including multicasted between subgroups and the entire group, in addition, the number of rounds is used to measure the group rekeying time.

The metrics used to evaluate computational complexity are the total times of modular exponentiations or the scalar-point multiplication performed by all of the group members.

Let the total number of members be $n$, the height of the key tree be $h$, numbers of subgroups be $k$. Table III summarizes the communication and computational costs of the three protocols.

From the Table III, with respect to time (i.e., number of rounds), during the setting-up period all protocols scale linearly with group size. The proposed extended scheme mainly depends on the key tree height. But for joining and leaving events, the proposed hybrid scheme is ahead of the rest, no matter members or sub-controllers joining or leaving the group, only one round is needed to generate a main group key.

### Table III. Comparison of Communication Overhead and Computational Cost

<table>
<thead>
<tr>
<th>Protocols</th>
<th>Protocol</th>
<th>Round</th>
<th>Communication Cost</th>
<th>Computation Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Unicast</td>
<td>Multicast</td>
</tr>
<tr>
<td>TGDH</td>
<td>Session</td>
<td>$h$</td>
<td>0</td>
<td>$2n+2$</td>
</tr>
<tr>
<td></td>
<td>Join</td>
<td>2</td>
<td>0</td>
<td>$2n +2$</td>
</tr>
<tr>
<td></td>
<td>Leave</td>
<td>1</td>
<td>0</td>
<td>$2n -4$</td>
</tr>
<tr>
<td>CLIQUES</td>
<td>Session</td>
<td>$n+1$</td>
<td>$2n+3$</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Join</td>
<td>2</td>
<td>$n+1$</td>
<td>$n+1$</td>
</tr>
<tr>
<td></td>
<td>Leave</td>
<td>1</td>
<td>0</td>
<td>$n -1$</td>
</tr>
<tr>
<td>Proposed</td>
<td>Session</td>
<td>$h$</td>
<td>$n-k$</td>
<td>$k+1$</td>
</tr>
<tr>
<td>Scheme</td>
<td>Join</td>
<td>1</td>
<td>1</td>
<td>$3+k$</td>
</tr>
<tr>
<td></td>
<td>Leave</td>
<td>1</td>
<td>1</td>
<td>$2+k$</td>
</tr>
</tbody>
</table>

B. Experiment Results

The communication overhead becomes greater when the number of member joining and leaving the group increases. Next, we evaluate the number of sent packets in the transmission of unicast and multicast when dealing with the join event and leave event in the three protocols above.

Assume the group size is 32, 64, 128, 256 and 512 members respectively, and restrict the maximum number of a subgroup member is 50. Fig. 5 depicts measurement for join, and Fig. 6 describes the result for leave. As expected, our scheme is much lesser than TGDH and CLIQUES, since the approach depends only on the number of subgroups. Compared to TGDH and CLIQUES, our approach consumes about 5% of memory in average and 10% at most. Fig. 6 further shows in TGDH the most number of messages are required, and
Cryptography relies on expensive operations such as exponentiations and scalar-point multiplications. Especially for exponentiations, it costs more time than scalar multiplications [10]. To demonstrate the cryptographic overhead, the number of serial exponentiations for TGDH, CLIQUES and scalar-point multiplications is measured for common events.

Still suppose there are totally 512 members in a group and the maximum number of subgroups in our approach is 50. From Fig. 7, the proposed hybrid scheme requires less cost as compared to TGDH and CLIQUES in terms of the number of cryptographic operations. CLIQUES is the worst performer due to many modular exponentiations. TGDH costs are in the middle and depend on the key tree height \( h \) in all cases. When protocols being set up, our hybrid scheme requires a linear number of scalar-point multiplications relative to the group size. But in the other cases, the computational overhead of scalar multiplications depend on the number of subgroups \( k \).

C. Security Analysis

The ECDH-based group-oriented key agreement scheme is a variant of the DH key agreement protocol, which is based on the difficulty of solving the DLP, i.e. ECDLP mentioned above. Consequently, benefiting from the same security properties, in the presented scheme member \( m_i \)'s key \( x_i \) is secure, since only the authorized members know their secret key \( x_i \), and the random values \( x_i \) is protected by the ECDLP. The security of this scheme is shown in the following lemma.

**Lemma 1.** Choosing elliptic curve \( E(F_p) \), given a prime \( p \neq 0 \), let \( Q_0=x_0(x_1P), P \in E(F_p), x_i \in (0: n-1) \) and \( Q_0=x_0(x_1=x_2=(x_2(x_1P))) \). According to the known \( Q_0 \) and \( Q_{0i} \), it is difficult to compute \( x_i \).

**Proof.** Let \( Q_1=x_1P, X_{0i}=x_0 \times_0 \times_1 \times_2 \), and \( X_n=x_0 \times_1 \times_2 \). Then it is difficult to find \( X_0 \) if given \( Q_0 \) and \( P \).

Similarly, denoting \( X_{0i}=x_0 \times_0 \times_1 \times_2 \times_i \times_1 \times_2 \), so, \( Q_{0i}=X_{0i}(x_1P) \). This is also hard to determine \( X_{0i} \). Thus, the value \( x_i \) cannot be evaluated from the public contributions \( Q_0 \) and \( Q_{0i} \). This completes the proof.

Security properties of group key management include group key secrecy, forward secrecy and backward secrecy [17]. Group key secrecy guarantees that it is at least computationally infeasible for an adversary to discover any group key. Forward secrecy provides security for subtractive events, such as member-leaving, which guarantees that departing group members cannot discover the subsequent group keys. Similarly, backward secrecy provides security for additive events such as member-joining, i.e., it guarantees a new member or a passive adversary cannot discover the preceding group keys. To fulfill the secrecy requirements, in the proposed scheme, the group rekeying will be revoked through all of the authorized nodes in a timely fashion whenever group membership changes. As seen in the following theorem, the extended ECDH-based cryptosystem has the security properties.

**Theorem 1.** The proposed ECDH-based scheme provides the feature of computational group key secrecy.

**Proof.** Let \( A \) be a passive attacker, and it can intercept messages sent over a public channel and obtain intermediate public contribution multicasted by each group or sub-group controller. A cannot discover the group key unless it learns at least one secret key. Due to
the hardness of the ECDL problem the adversary is not able to reveal these values from their public counterparts. Hence, A can neither compute nor distinguish the group key.

**Theorem 2.** The proposed ECDH-based scheme provides the property of forward secrecy and the backward secrecy.

**Proof.** For member leave case. Suppose someone be going to leave the group, and then the subgroup controller involved in the changing membership first varies its secret key, additionally computes and sends a renewed group key to the remaining members. Consequently a departing group member is not able to obtain any subsequently used group key. Forward secrecy is provided.

For member join case. When a new member is going to join the group, the rekeying algorithm will be revoked. After changing its secret key, the former sub-controller computes new intermediate contributions and sends these updated values to the new member. Therefore a new member cannot obtain the previously shared group key. Backward secrecy is provided.

VI. CONCLUSION

In this paper, a distributed group key agreement scheme is proposed based upon ECDH, which is more lightweight than regular DH protocol. During the group key establishment, the shared group key is calculated through all of the members of a multicast group rather than distributed by a key distribution center. Compared to classic distributed schemes, the protocol can be extended to a hybrid structure and applied in large dynamic multicast environments, this helps to provide better scalability.

According to the performance comparison, communication overhead of our proposed approach depends on the number of subgroups and requires less sent messages. The communication overhead is significantly reduced. Therefore, the approach is more efficient than other schemes, such as CLIQUES and TGDH. In terms of computational cost, the proposal uses scalar multiplication, and costs less time in comparison with the modulus and exponent operation performed in CLIQUES and TGDH. Regarding the security properties, our scheme provides group key secrecy, forward secrecy and backward secrecy, which can meet the security goals of group communication excellently.

ACKNOWLEDGMENT

The author is especially grateful to Yongdiao Xu for his precious comments and contributions to this work. The author would like to thank the editors and anonymous reviewers for their constructive advices that greatly improved the paper.

REFERENCES


Qiuna Niu was born in Henan, China in 1977. She has completed her Bachelor of industrial automation in Changchun University of Science and Technology, Changchun, China in 2001, and has her M.S in Communication and Information System from Shandong University, Jinan, China 2004. She is a Ph.D. candidate of Ocean University of China.

From 2004 to now, she is a lecturer at the College of information Science & Technology, Qingdao University of Science & Technology. Her current research interests include information theory and coding, wireless communication and networks, and security issue in networks.