Design of Queue-Based Group Key Agreement Protocol Using Elliptic Curve Cryptography

Priyanka Jaiswal, Abhimanyu Kumar and Sachin Tripathi

Abstract Secure group communication is an important research issue in the field of cryptography and network security, because group applications like online chatting programs, video conferencing, distributed database, online games etc. are expanding rapidly. Group key agreement protocols allow that all the members agree on the same group key, for secure group communication, and the basic security criteria must be hold. The design of secure group communication can be very critical for achieving security goals. Many group key agreement protocols such as Tree-based Group Diffie-Hellman (TGDH) Kim et al. (ACM Trans Inf SystSecur (TISSEC) 7(1):60–96, (2004)) [1], Group Diffie-Hellman (GDH) Steiner et al. (IEEE TransParallel Distrib Syst 11(8):769–780, (2000)) [2], Skinny Tree (STR) Wong et al. (IEEE/ACMTrans Netw 8(1):16–30, (2000)) [3] etc., have been established for secure group communication, but they have suffered from unnecessary delays as well as their communication cost increased due to increased exponentiation. An alternative approach to group key agreement is the queue based group key agreement protocol that reduces unnecessary delays, considers member diversity with filtering out low performance members in group key generation processes. We propose a novel queue based group key agreement protocol that uses the concepts of elliptic curve cryptography. The proposed protocol gives better results than the other existing related protocols and it also reduces computational overheads.

Keywords Group diffie hellman · Elliptic curve cryptography (ECC) · Queue based group key agreement

P. Jaiswal (✉) · A. Kumar · S. Tripathi
Department of Computer Science and Engineering, Indian School of Mines, Dhanbad 826004, Jharkhand, India
e-mail: priyanka_jais4@yahoo.co.in

A. Kumar
e-mail: abhi_a1ks@yahoo.co.in

S. Tripathi
e-mail: var_1285@yahoo.com

© Springer India 2015

J.K. Mandal et al. (eds.), *Information Systems Design and Intelligent Applications*, Advances in Intelligent Systems and Computing 339,
DOI 10.1007/978-81-322-2250-7_17

1 Introduction

Most of the groupware applications such as video conferencing, online chatting, online game, net gambling, etc. are increasing day by day over internet. Security is the major concern in maintaining such groupware application. Basic security services such as privacy, integrity and authentication, are necessary for groupware application as well as key management is very important in group key agreement protocols for security purpose. Group key can be managed by one of the three ways, as centralized, distributed and contributory group key management [4]. In centralized group key management a single entity or a set of entity is involved in the generation and distribution of group key for group members via a pair-wise secure channel established with each group member. Centralized group key management is a simple group key management as it involves a single (or a set) of the entity. However, Centralized group key management is inappropriate for peer group communication as it involved a trusted third party or online key generation center for supporting the group operation every time. Continuous availability of an entity can be addressed as fault tolerance and replication. However, Centralized group key management work well for one-many multicast network scenarios. Distributed group key management is more suitable in peer group communication as it involved dynamically selecting a group member for distribution of group key. In contrast, in contributory group key management all members equally contribute in generation of group key. This type of protocols is appropriate for dynamic peer groups. This approach avoids the problem of single point of failure.

Group key plays major role in establishing secure group communication. So, key generation is a major task in group key agreement through secure way. Many of group key agreement protocols have been developed earlier for secure group communication [1, 5], but they have some disadvantages. Since the group generation processes takes many modular exponentiations and long time in generation of group key. For achieving higher security, group key protocol should be dynamic, means it should change for each new join or leave member, so that new member have not any knowledge about prior information [6]. Therefore group key management protocol focusing on the group key generation efficiently [1, 7–9]. Modular exponentiation is very expensive in computation of group key [1]. The number of exponentiations for membership depends on group size as when the group size increased the number of exponents will also increase. Tree Based Group Diffie-Hellman (TGDH) uses the concept of Diffie-Hellman key exchange with logical tree structure to achieve efficiency. The efficiency of TGDH is $O(\log_2 n)$, where n is the group size. However, some extra overhead occurred in maintaining a perfect key tree balance. Skinny tree has lower communication overhead, but it increases computation. Burmester–Desmedt (BD) distributes and minimizes computation by using more messages broadcast. All these protocols using similar security properties including group key independence. TGDH, STR, and other key management

are under a homogeneous computing and network environment. However, one of the problems associated with the tree structure is the balancing of the tree, when the members are changing the group.

2 Related Work

Groupware application like video conferencing, online gaming, e-chatting, etc. may have different settings. To provide secure group communication, secure key distribution and efficient key management are very necessary to maintain integrity, confidentiality, and authentication. For secure group communication, group key management is responsible for generating the group key and distributing it to all intended recipients in a secure way over an insecure channel [4]. Group key management can be categorized into centralized, distributed and contributory group key management. In a centralized approach, a single entity or a set of entities is involved in generation and distribution of group key. The centralized group key management protocol is not suitable for peer group because it involves a continuous availability of trusted third party (TTP) for generation and distribution of group key, that may cause of single point of failure. However, the Distributed group key management involves dynamically selecting the group key and distributing it to other group member, which is more suitable for dynamic peer group communication. Distributed group key management involves distributing key in a decentralized way. In contrast to centralized approach, contributory group key management involves each group member to equally contribute, to generate the group key. This avoids the problems of single point of failure. Contributory group key management is most suitable for peer group communication, because each member has an equal opportunity to generate a group key. Therefore the proposed protocol, uses the contributory group key management approach to generate the group key. It uses the elliptic curve cryptographic technique to reduce the exponentiation. Some of the other related protocol like Burmester–Desmedt (BD) [10], Group Diffie–Hellman (GDH) [2], Skinny Tree (STR) [3] and Tree Based Group Diffie Hellman (TGDH) [1] and Queue Based Group Diffie Hellman (QGDH) [5], have some limitations. The Burmester–Desmedt (BD), protocol support dynamic operation and uses modular exponentiation to reduce communication overhead, but it requires more message exchange to generate the group key. Group Diffie Hellman (GDH) provides better security, but it requires more computation and communication overhead. The Skinny Tree (STR) Protocol is more suitable for member joining group operation, it has relatively low communication cost, but it does not work well for exclusion of members. The Tree Based group Diffe Hellman (TGDH) provides efficient group key agreement protocol from above related protocols. The Queue Based Group Diffie Hellman (QGDH) uses decentralized group key management and contributory group key distribution mechanism to improve efficiency, however modular exponentiation increases computational overhead. Therefore, we have proposed a new elliptic curve based group key agreement protocol.

3 Preliminary

3.1 Elliptic Curve Over Finite Field (F_p)

Let $p \geq 3$ be a prime number. Let $a, b \in F_p$ be such that $4a^3 + 27b^2 \neq 0$ in F_p. An elliptic curve E over F_p is defined by the equation $Y^2 \bmod p = (x^3 + ax + b) \bmod p$ where (x, y), $x, y \in F_p$, together with an extra point O, called point identity. The set of points $E(F_p)$ forms an abelian group with the following addition rules.

- Identity: $P + O = O + P = P$ for all $P \in E(F_p)$.
- Negativity: If $P(x, y) \in E(F_p)$ then $(x, y) + (x, -y) = O$, The point $(x, -y)$ is defined as $-P$ called negative of P.
- Point addition: Let $P(x_1, y_1)$, $Q(x_2, y_2) \in E(F_p)$, then $P + Q = R \in E(F_p)$ and coordinate (x_3, y_3) of R is given by $x_3 = \lambda^2 - x_1 - x_2$ and $y_3 = \lambda(x_1, x_3) - y_1$ where $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$.
- Point doubling: Let $P(x_1, y_1) \in E(K)$ where $P \neq -P$ then $2P = (x_3, y_3)$ where $x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1$ *and* $y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1$.

3.2 Elliptic Curve Discrete Logarithm Problem (ECDLP)

Given an elliptic curve E defined over a finite field F_p, a *point* $P \in E(F_p)$ of order n, and a point $Q \in \langle P \rangle$, find the integer $l \in [0, n - 1]$ such that $Q = LP$ The integer L is called the discrete logarithm of Q to base P, denoted $L = \log_p Q$.

3.3 Elliptic Curve Diffie Hellman (ECDH)

Elliptic Curve Diffie Hellman is one of the key exchange protocol used to establishes a shared key between two parties. ECDH protocol is based on the additive elliptic curve group. ECDH selecting the underlying field (F_p) or $GF(2^k)$, the curve E with parameters a, b and the base point P is equal to n. The standards often suggest that we select an elliptic curve with prime order and therefore any element of the group would be selected and their order will prime number n. At the end of the protocol the communicating parties end up with the same value K which is the point on the curve.

4 Proposed Queue Based Group Key Agreement Protocol

In the proposed protocol there are (M_1, M_2, M_3, …, M_n) n number of members in the group, and there is a group controller server (GCS) responsible for every member authentication and key generation. The group controller server (GCS) manages all the consisting group member and contain information about the group members such as current login list of members, information about the registered member, current session key etc. GCS also manages the BKQ according to the arrival of the member. GCS requests all members to generate blind key, then GCS creates a BKQ and stores the blind key in BKQ into the order of the arrival. The highest performance member blind key is always stored on the front end of the BKQ, whereas the low performance member blind key is stored on the rear end of the BKQ. Figure 1 shows an example of structural Blind Key Queue (BKQ), in which the first spot blinded key is computed with the last spot blinded key ($x_{n1}x_{n2}B$), and the member with the second spot is computed key with the member in last second spot ($x_{n2}x_{n3}B$). The proposed algorithm can be categorized into setup phases and key generation phase, in the setup phase KGC initializes public parameters, and in second phase KGC and users are contributed to generate group key.

4.1 Setup

(By KGC) This algorithm takes a security parameter $k \in Z^*$ and does the following:

KGC chooses a k bit prime $p \in Z_p^*$ and determine the tuple F_p, E/F_p, B.

where

E/F_p: Elliptic curve over F_p.

B: Base point on elliptic curve.

KGC publishes parameters = {F_p, E/F_p, B} as system parameters.

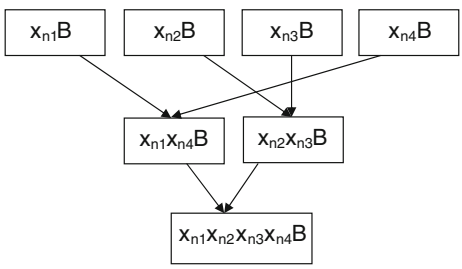

Fig. 1 Blind key queue structure

4.2 Key Generation

- Each user select an integer x where $x \in Z_p^*$.
- Each user calculates Q_i as a product of private key and a base point (B). $Q_i = x_i * B$, Q_i is the public key of member i, where $1 \leq i \leq n$.
- Each member sends Q_i to group controller server (GCS).

When GCS got many requests, GCS arranges those requests in order of their arrival and performs the following operations.

- GCS stores Q_i in their respective position in queue in order of their arrival from different users.
- GCS arranges each request coming from the group and when a threshold number of members are registered, then GCS says front end and rear end members to generate group key means member with A_1 spot will generate a group key with member in A_n spot. Member at A_2 spot will generate with member at $A_{(n-1)}$ and so on.
- After that GCS waits for threshold time, and if any group key arrives, then GCS will recreate public key Queue (PKQ) in order of their arrival. After the threshold time, if group key from certain group is absent, then GCS gives them last chance to generate group key together or independently and send to GCS in the given time. If they still fail then GCS ignore them. They can join later using 'one by one' algorithm.
- The last step is repeated until level $\log_{2n+1}$ and the session key is generated and stored in current session key (CSK) register. The current session key (CSK) = $x_1x_2x_3\ldots x_i\ldots x_{(n-1)}x_nB$.

5 Member Join

Join and leave operation is an important part in dynamic group key agreement protocol. Since any time a member or a set of members can join or leave the group.

Whenever a new member wants to join the group, the group controller broadcast a control message to others member to update their public key and then sends back to group controller, then group controller arranges all the keys in order of their arrival as well as with the new member key. It includes the following steps.

- When a new member wants to join the group, it sends their public key to GCS.
- GCS stores member's public key and identity in the database.
- GCS updates, current session key $CSK = CSK' + Q_i$ (using elliptic curve point addition).
- GCS selects a random number N between $[0, p - 1]$.
- Calculate new current session key $CSK1 = N*CSK$.
- GCS broadcasts $CSK1$ as new session key to all the members including new member and updated CSK.

6 Member Leave/Mass Leave

If a member or set of member wants to leave the group then the group session key of the resulting group must be updated to provide the forward secrecy. The leave operation includes following steps.

- When a member wants to leave the group, it sends leave request to GCS, by sending a public key as Q_i.
- GCS updates, current session Key $CSK = CSK - Q_i$.
- GCS selects a random number N on $[0, p - 1]$.
- Calculate new current session key $CSK1 = N * CSK$.
- Update CSK to $CSK1$.
- Broadcast the new CSK to all members except leaving member.

7 Security Analysis and Comparison with Other Protocols

This protocol maintains security due to elliptic curve discrete logarithm problem. Adversary wants to find out the value of x by the given value of B and Q where B is the base point of the elliptic curve and $Q = x \cdot B$, where B is x times added to itself to generate Q. However, it is computationally infeasible to find out the value of x due to elliptic curve discrete logarithm problem. The responses of the proposed protocol from the various attacks are addressed as follows.

7.1 Known Session Key Security

In the proposed protocol, each member M_i randomly chooses a private key $x_i \in Z_p^*$ in the session. The session key depends on each member's private key. If an adversary compromise one session key, then it is not an easy task to compromise other's session key. So, it cannot find other's session key. So, this protocol provides known session key security.

7.2 No Key Control

The proposed protocol is fully contributive protocol, because the session key of the proposed protocol depends on each members blind key which are computed with each member's private key as $Q_i = x_i\mathrm{B}$.

7.3 Forward Secrecy

The coming member does not know about what was the group key earlier because they receive information about the generating point of the elliptic curve. They cannot guess the group key because the group key is computed with each one secrets and generating point of the group key. The secret is a random number, taken privately by each member. Random number is unknown to connecting a member. So, the new member cannot find the previous group key.

7.4 Backward Secrecy

The leaving member cannot compute the new group key because the share of the leaving member is no longer part of the group key. The group key is updated by GCS using the blind key contribution of all members except the leaving member. So, all the previous group keys are completely unknown to leaving member/s.

7.5 Key Independence

A member, who knows a set of key, cannot discover previous or future key. So the proposed protocol maintains key independence.

7.6 Comparison with Other Existing Protocols

This section compares the cost of major group key management operations of the proposed protocol with other existing group key agreement schemes. The comparison Table 1 shows the following notation for comparison.

n: Number of members in the group
h: Height of the original key tree
p: Number of leaving member.

8 Conclusion

We have proposed an efficient queue based group key agreement protocol. We have analyzed many prior group key agreement protocols like TGDH, STR, BD, QBDH etc., they provide better security but they takes more computational overheads. So, we have used elliptic curve cryptographic technique that removes exponentiation to

Table 1 Comparison with other protocols

Protocols		Messages	Exponentiation
TGDH	Join	3	3h/2
	Leave	1	3h/2
	Mass leave	2n	3h
STR	Join	3	4
	Leave	1	3n/2 + 2
	Mass leave	1	3n/2 + 2
GDH	Join	n + 3	n + 3
	Leave	1	n − 1
	Mass leave	1	n − p
BD	Join	2n + 2	3
	Leave	2n − 2	3
	Mass leave	2n − 2p	3
QGDH	Initial setup	2n − 2	$3(\log_2 n)/2$
	Join	2n − 2	$3(\log_2 n)/2$
	Leave	2n − 2	$3(\log_2 n)/2$
	Mass leave	2n − 2	$3(\log_2 n)/2$
Proposed protocol	Initial setup	2n − 2	0
	Join	2	0
	Leave	2	0
	Mass leave	2	0

reduce computational overheads. The comparison table shows that it provides better results than the other group key agreement protocols.

Acknowledgment This work is supported by UGC (University Grant Commission), Govt. of India under project No.—UGC(77)/2012-13/316/CSE. We would like to thank UGC for the support in this research work.

References

1. Kim, Y., Perring. A., Tsudik, G.: Tree-based group key agreement. ACM Trans. Inf. Syst. Secur. (TISSEC). **7**(1), 60–96 (2004)
2. Steiner, M., Tsudik, G., Waidner, M.: Key agreement in dynamic peer groups. IEEE Trans. Parallel Distrib. Syst. **11**(8), 769–780 (2000)
3. Wong, C., Gouda, M., Lam, S.: Secure group communication using key graphs. IEEE/ACM Trans. Netw. **8**(1), 16–30. (2000)
4. Amir, Y., Kim, Y., Rotaru, C.N., Tsudik, G.: On the performance of group key agreement protocols. ACM Trans. Inf. Syst. Secur. (TISSEC). **7**(3), 457–488 (2004)
5. Hong, S.: Queue based group key agreement protocol. Int. J. Netw. Secur. **9**(2), 135–142 (2009)

6. Diffie, W., Hellman, M. E.: New directions in cryptography. IEEE Tran. Inf. Theor. IT **22**(6), 644–654 (1976)
7. Hong, S., Benitez, N.L.: Enhanced group key computation protocol. In: International Conference on Security and Management-SAM'06. Las Vegas, USA, 26–29 June 2006
8. Hsu, C.F., Cui, G.H., Cheng, Q., Chen, J.: A novel linear multi-secret sharing scheme for group communication in wireless mesh networks. J. Netw. Comput. Appl. **34**(2), 464–468 (2011)
9. Bohli, J.M.: A framework for robust group key agreement. Comput. Sci. Appl. ICCSA, 355–364 (2006)
10. Burmester, M., Desmedt, Y.: A secure and efficient conference key distribution system. In: Advances in cryptology–Eurocrypt'94, pp. 275–286. Springer, Berlin (1994)