



A Novel Technique for Asymmetric Group Key Agreement for Dynamic Open Group Connection Networks

R. Siva Ranjani*
Research Scholar,
Dept. of CS&SE,
Andhra University,
Visakhapatnam,
Andhra Pradesh, India.

D.Lalitha Bhaskari
Associate Professor,
Dept. of CS&SE,
Andhra University,
Visakhapatnam,
Andhra Pradesh, India.

P.S.Avadhani
Professor,
Dept. of CS&SE,
Andhra University,
Visakhapatnam,
Andhra Pradesh, India.

Abstract— *In this piece of work, we proposed an authenticated asymmetric group key agreement (A^2GKA) protocol using bilinear mapping for dynamic networks. This protocol is mostly suitable for multicasting (or broadcasting) system without relying on the group controller to distribute secret keys to the group members. Our proposed protocol is equipped with various security issues including perfect forward security. Also, we revised the group key agreement definition and distinguished the symmetric key agreement from AGKA protocol. The protocol A^2GKA enables the group members to negotiate a common encryption key which is accessible to the public, by holding their secret decryption key. Conventional AGKA is secured against passive attacks only, whereas our proposed one is impregnable against both active as well as passive attacks.*

Keywords— *Asymmetric Group key agreement, Group communication, Dynamic group, bilinear map, broadcast encryption*

I. INTRODUCTION

A group key agreement (GKA) protocol allows users to establish a common secret key through open connection networks. The main goal is to establish a secured channel among the group members. GKA protocol allows a group of users to get involved in the key generation to come up with a common secret key from which a common session key can be derived. Nowadays, GKA protocols are likely used in group communications, audio conferences, video conferences, social networks and collaborative communication etc.

The popular, conventional GKA requires two or more rounds to establish a secret key; Burmester-Desmedt [7] protocol is the best known protocol. In 2009 Wu et al [4] proposed a static one-round ASGKA protocol from scratch, since that the participant does not hold any secret value. One round key agreement protocol is far more beneficial than any two or three round key agreement. In their resolution the group members negotiate a common encryption key instead of a common secret key, and each user will be allowed to hold their secret decryption key. Therefore, the outsiders can broadcast the encrypted messages to the group members by using a public encryption key.

A. Motivation and Contribution.

The protocols developed from the scratch are efficient, but they are secured only against passive adversaries, who just listen to the communication. But in the real world, attackers are usually active adversaries, who can control the communication channel to emplace powerful attacks. In particular, an active adversary is able to do man-in-middle attack. Hence, it is imperative for ASGKA to secure communication against active adversaries. Authenticated key agreement protocols allow only authorized participants to compute the secret keys which were used in group communication. In the basic papers of [13, 6,9,10, 11] an authenticated version of the static ASGKA protocol was proposed without involving additional rounds. The analysis shows that, protocol is equipped with the security properties like known key secrecy, unknown key share, key compromise impersonate, perfect forward secrecy and key control. Similarly in [4] protocol, for a group of n participants each participant is publishing $O(n)$ group elements. When the group size is becoming larger, then the transmission overhead is very high. From the analysis of the algorithm [4], it is clear that the user is only sending the Private Key, the public key and identification of the user to the certificate authority for acquiring the certificate. The Certificate authority only checks for validity and issues the master key, which is used in the private key generation. But in our proposal, along with the private key, public key and ID, the user should generate a signature on all the parameters and then he sent it to Group Controller. The Group Controller will check the validity of the received parameters by comparing them with decrypted signature. Along with this feature, the proposed protocol calculates the hash value of the message m and then passes it to the other group members, it helps the other users to take the decision whether to accept or reject the received message.

In this paper, a generic construction of dynamic authenticated asymmetric group key agreement without a Group Controller is proposed, by combining a conventional group key agreement and public key encryption. Our construction is similar to the authenticated group key agreement

Thus, the contributions of the work are as follows:

- Allows the groups with dynamism without compromising the security.
- Allow the mutual authentication between the group members and Group Controller.
- Performance and efficiency are compared with existing methods.
- Allows the users to broadcast public information by hiding their secret information. Common group key is derived from collecting public information.

B. Related Work.

Diffie-Hellman[8] proposed the first solution to the key agreement. Later, Joux[3] proposed a one rounded tri partite key agreement protocol. Indeed, many attempts have been made to extend both Diffie Hellman and Joux protocol to n users. Among them, the famous GKA protocol Burmester-Desmedt protocol [7], which required two rounds and is most efficient existing GKA protocol without constraints on n value.

In open networks the key agreement protocol is essential to resist attacks from the active antagonist. However, the basic three protocols Diffie-Hellman, Joux and Burmester-Desmedt do not authenticate the communication. Hence, they are not suited for counteracting active attacks. Many protocols are designed by including authentication to their protocols [6,9,10,11]. In [10], Katz and Yung proposed a scalable compiler which transforms a secured authenticated GKA into the secured authenticated GKA protocol. Their proposal is a modified version of Burmester-Desmedt protocol to achieve active security by adding one round. Choiet.al [9] proposed a GKA protocol based on the bilinear version of Burmester-Desmedt protocol in the context of identity based public key cryptosystem[7]. Dutta and Barua[6] presented an authenticated GKA, which is a variant of Burmester-Desmedt with considerably better efficiency. Quan et.al[3] proposed an one round asymmetric group key agreement protocol for open networks using bilinear mapping, achieves active security and passive security. It also equipped with the potent to deal with security issues like perfect forward secrecy, key control and known key security.

Lei et.al[5] proposed a one round identity based authenticated GKA protocol, using an identity based batch signature to provide security properties like key compromise, impersonate resilience and malicious participants identification. The limitation with this protocol is, that it allows joining with in the group when the rows in the billboard-II are free. Any number of users can be joined in the group in the proposed protocol. .

C. Paper Outline.

We organize the rest of the paper as follows. Section 2 reviews bilinear maps and some complexity assumptions. Section 3 defines the security of A²GKA protocols. Our identity based batch multi-signature is presented in Section 4. Section 5 proposes and analyzes our A²GKA protocols. In Section 6, we show that our protocols are also secure against some insider attacks. In section 7, the way how we reduced the tradeoff in communication overhead. Finally, we conclude in Section 8.

II. PRELIMINARIES

In this section, we put forward the notations, definitions that we used in the discussion of the forth coming sections.

A. Bilinear maps.

We review the basic notations of the bilinear maps [12,14] under our proposal, Let $(G_1,+)$, $(G_2,+)$ and (G_T, \cdot) are the three multiplicative group of prime order $q > 2k$ for a security parameter $k \in \mathbb{N}$. We say that, there exists a bilinear map $\hat{e} : G_1 \times G_2 \rightarrow G_T$ satisfies the following properties:

1. Bilinearity: $P, Q \in G_1, a, b \in \mathbb{Z}, \hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$. This can be restated in the following way.
For $P; Q; R \in G_1; \hat{e}(P + Q; R) = \hat{e}(P; R)\hat{e}(Q; R)$ and $\hat{e}(P; Q + R) = \hat{e}(P; Q)\hat{e}(P; R)$
2. Non-degeneracy: $P \in G_1, \hat{e}(P, Q) = 1 \quad Q \in G_2$ iff $P = 1 \in G_1$.
3. Computability: $P \in G_1$ and $Q \in G_2$ then $\hat{e}(P, Q)$ is efficiently computable.

It is called symmetric bilinear map groups if $G_1=G_2$ and $P=Q$. A bilinear map is defined as a probabilistic polynomial time algorithm (E) that takes a security parameter k and returns a uniformly random tuple $(G_1, G_T, \hat{e}, g, q)$ of bilinear parameters, where g is the generator of G_1 and \hat{e} : is the bilinear map.

Consequences of pairings:

Pairings have important consequences on the hardness of certain variants of the Diffie-Hellman problem. For instance, symmetric pairings lead to a strict separation between the intractability of the Computational Diffie-Hellman problem and the hardness of the corresponding decision problem. The security of our proposal is based on the hardness of the computational Diffie Hellman(CDH) problem, Divisible computational Diffie Hellman and K-Bilinear Diffie Hellman exponent, which is described in next:

Computational Diffie-Hellman (CDH) : given g, g^α, g^β for unknown $\alpha, \beta \in \mathbb{Z}_q$, compute $g^{\alpha\beta}$

CDH Assumption: Let E be an algorithm which has an advantage in solving CDH problem. The assumption states that $\text{Adv}[E]$ cannot be negligible for any polynomial time algorithm E.

$$\text{Adv}[E] = \Pr[E(g, g^\alpha, g^\beta) = g^{\alpha\beta}]$$

Divisible Computational Diffie-Hellman (DCDH) Problem[17] : given g^α, g^β for unknown $\alpha, \beta \in \mathbb{Z}_q$, compute $g^{\alpha/\beta}$.

DCDH Assumption: The assumption states that, no polynomial time algorithm (E) that can solve the DCDH problem with the non negligible problem.

$$\text{Adv}[E] = \Pr[E(g^\alpha, g^\beta) = g^{\alpha/\beta}]$$

k-Bilinear Diffie-Hellman Exponent (k-BDHE) Problem [14]: Given $g; h$, and $y_i = g^{\alpha^i}$ in G_1 for $i = 1, 2, \dots, k, k+2, \dots, 2k$ as the input and computes $\hat{e}(g, h)\alpha^{k+1}$. Since the input vector lacks $g^{\alpha^{k+1}}$ term, the bilinear map does not seem to help to compute $\hat{e}(g, h)\alpha^{k+1}$.

k-BDHE Assumption: Let E be an algorithm which has an advantage in solving k-BDHE problem. The assumption states that, there is no polynomial-time algorithm that can solve the k-BDHE problem with non-negligible probability.

$$\text{Adv}[E] = \Pr[E(g, h, y_1, y_2, \dots, y_k, y_{k+2}, \dots, y_{2k}) = \hat{e}(g, h)\alpha^{k+1}]$$

B. Group key agreement Security Requirements.

Some of the security requirements of group key agreement protocols [18, 19, 20] include group key secrecy, backward key secrecy, known key security, key compromise impersonation, key control, unknown-key share and perfect forward key secrecy. The definitions of those requirements are used for evaluating the security of the conventional group key agreement protocols. In order to evaluate the security of A^2GKA those requirements are defined in table I.

TABLE I: A^2GKA Requirements

Term	Definition
Group key secrecy	Guarantees that it is computationally infeasible for a passive adversary to discover any group key.
Backward key secrecy	Guarantees that a passive adversary who knows a contiguous subset of group keys cannot discover preceding group keys.
Known key security	For each run of the session, protocol establishes a unique decryption key for each group member and a unique common group encryption key. Even, if the leakage of group decryption key(s) of one session should not help in estimation of other sessions.
Key compromise impersonation	The accord of A's private key will allows the active adversary to impersonate A, but it should not allow the adversary to impersonate other users in the presence of A.
Key control	No participant in the group communication is able to force the group encryption key or decryption key to a preselected value.
Unknown key share	An Active adversary is unable to make one participant believe that the key is shared with one party when it is shared with other party.
Perfect forward key secrecy	Ensures that a session key derived from a set of public and private keys will not be compromised if one of the private keys is compromised in the future. Perfect Forward Secrecy has the additional property that an agreed key will not be compromised even if agreed keys derived from the same long-term keying material in a subsequent run are compromised.

III. Authenticated Asymmetric Group Key Agreement Protocol (A^2GKA)

We propose A^2GKA , motivated by the Asymmetric Group Key Agreement [4]. In this protocol, each participant is having a public private key pair for authentication purpose. In the protocol all the users who want to involve in the communication have to register with the Group Controller (U_0). Our proposal can be described in following seven algorithms:

- Group setup:** In this step, if any user U_i wants to take part in the communication first, he has to get the permission from the Group Controller by sending their contribution to U_0 , who collects all U_i contributions to computing the master secret value. This is done by performing the following operations.

a. User U_i contribution Preparation and signature generation

Each user U_i having identity ID_i chooses a random message $(x_i) \in \mathbb{Z}_q^*$ and takes the current time stamp T . The values of $(ID_i; ID_0; x_i; T)$ are encrypted using U_0 's public key (PU_0).

$$e = PU_0\{ID_i; ID_0; x_i; T\}$$

Also calculates the signature ($sign_i$) by encrypting $(ID_i; ID_0; x_i; T)$ using his private key (PR_i)

$$sign_i = PR_i\{ID_i; ID_0; x_i; T\}$$

Each user U_i sends $\{e, sign_i\}$ to Group Controller (U_0)

$$U_i \rightarrow U_0: \{e, sign_i\}$$

b: U_i message reception and Verification by U_0

The U_0 receives all the messages from all the group members and decrypts them. U_0 then verifies the signatures of corresponding user U_i 's. It also checks for the validity of timestamp. U_0 accepts the U_i 's message if their signature and timestamp are valid.

c: master key generation

The Group controller collects all the x_i contributions from the group members and then computes a master secret key $K = XOR(x_1, x_2, \dots, x_n)$. On the input security parameter l , the Group Controller (U_0) generates a random tuple (G_1, G_T, g_1, q) . The two multiplicative groups G_1 and G_T with the prime order q , where G_1 is generated by g and $\cdot : G_1 \times G_1 \rightarrow G_T$. The U_0 also selects $K_1, K_2, \dots, K_n \in G_1$, where n is number of users are involved in that session for group communication, a master secret key

$K \in \mathbb{Z}_q^*$ and sets $g = \mathbb{Z}_q^*$ and also chooses hash functions $H_1: \{0,1\}^* \rightarrow G_1$. Finally, U_0 publishes the system parameters list $\mathcal{P} = (G_1, G_T, \hat{e}, g, q, K, K_1, K_2, \dots, K_n, H_1)$. where n is the largest group size that the system can support.

2. Private key Extraction:

In this step, each participant involved in group communication will collect the master key K coming from the U_0 and used it in private key generation. The user U_i chooses a random number S_i and then computes the private key for the group communication $PR_i = S_i \cdot K$.

3. Key Establishment:

In this stage, the participant generates and publishes the messages which will be used in generation of used in generation of group encryption and decryption keys. Without loss of generality, all the participants $U_1, U_2, \dots, U_n, n < P$, A user U_i holding her private key PR_i performs the following steps:

- i. Randomly chooses $h_i \in G_1, r_i \in \mathbb{Z}_q$ and compute $x_i = g^{-r_i}$ and $A_i = \hat{e}(h_i, g)$.
- ii. For all the users in the group $1 \leq j \leq n$, U_i computes $\Delta_{i,j} = h_j^{r_i}$.
- iii. Generate a signature δ_i on M_i using the private key PR_i .

Where $\delta_i = E_{PR_i}\{M_i\}$ and

$$M_i = \{ \Delta_{i,1}, \Delta_{i,2}, \dots, \Delta_{i,i-1}, \epsilon, \Delta_{i,i+1}, \dots, \Delta_{i,n}, x_i, A_i, ID_i \}$$

- iv. Finally, publish the parameters

$$\{ \Delta_{i,1}, \Delta_{i,2}, \dots, \Delta_{i,i-1}, \epsilon, \Delta_{i,i+1}, \dots, \Delta_{i,n}, x_i, A_i, \delta_i, ID_i \}$$

After completion of this stage, each participant can get the message as shown in the table II. In the table $\Delta_{i,i} = h_i^{r_i}$, only known to U_i and will not be published to other users in the group.

TABLE II: MESSAGES RECEIVED BY VARIOUS PARTICIPANTS

User	U_1	U_2	U_3	----	U_n	All
U_1	$\Delta_{1,1} = \epsilon$	$\Delta_{1,2}$	$\Delta_{1,3}$	----	$\Delta_{1,n}$	$(x_1, A_1, \delta_1, ID_1)$
U_2	$\Delta_{2,1}$	$\Delta_{2,2} = \epsilon$	$\Delta_{2,3}$	----	$\Delta_{2,n}$	$(x_2, A_2, \delta_2, ID_2)$
U_3	$\Delta_{3,1}$	$\Delta_{3,2}$	$\Delta_{3,3}$	----	$\Delta_{3,n}$	$(x_3, A_3, \delta_3, ID_3)$
.	
.	
.	
U_n	$\Delta_{n,1}$	$\Delta_{n,2}$	$\Delta_{n,3}$	----	$\Delta_{n,n} = \epsilon$	$(x_n, A_n, \delta_n, ID_n)$
Decryption key	d_1	d_2	d_3		d_n	(X, R)

4. Common Group encryption key Derivation:

Any user in the group can compute the group encryption key (X, R) using the formulae

$$X = \prod_{i=1}^n x_i, R = \prod_{i=1}^n A_i = \prod_{i=1}^n \hat{e}(h_i, g) = \hat{e}(\prod_{i=1}^n h_i, g)$$

After checking the validity of all users signatures $\delta_1, \delta_2, \dots, \delta_n$ only the group key (X, R) is accepted.

5. Individual decryption key Derivation:

All the user U_i 's in the group communication can calculate their decryption key (d_i) $d_i = \prod_{j=1}^n \Delta_{j,i}$, accepts the d_i if all users signatures $\delta_1, \delta_2, \dots, \delta_n$ are valid. The attacker cannot compute d_i since $\Delta_{i,i}$ is not published. After successful completion of this instance last row (decryption key) is generated as output shown in table. Expand the decryption key corresponding to the group key encryption key, we get $d_i = \prod_{j=1}^n \Delta_{j,i} = \prod_{j=1}^n h_j K_i^{r_j} = \prod_{j=1}^n h_j K_i^{\sum_{j=1}^n r_j}$

6. Encryption:

After knowing the public parameters and the group encryption key (X, R) , the message $m \in G_T$ can be encrypted by performing the following steps:

- a. Select a random number $t \in \mathbb{Z}_q$
- b. Compute the parameter $C_1 = g^t, C_2 = X^t, C_3 = m, C_4 = H_1(C_1, C_2, m)$
- c. Sends the cipher text $C = (C_1, C_2, C_3, C_4)$

7. Decryption:

After cipher text C reception, receiver U_i can decrypt and recover the message m by performing following steps:

- a. Calculate m by computing $\frac{C_3}{d_i(C_1)^t \hat{e}(C_2, C_4)}$

$$\begin{aligned} \text{i.e } m &= \frac{m \cdot R^t}{d_i(C_1)^t \hat{e}(C_2, C_4)^t} \\ &= \frac{m \hat{e}(\prod_{j=1}^n h_j, g)^t}{\prod_{j=1}^n h_j K_i^{\sum_{j=1}^n r_j} \hat{e}(K_i, g)^t} \\ &= \frac{m \hat{e}(\prod_{j=1}^n h_j, g)^t}{m \hat{e}(\prod_{j=1}^n h_j, g)^t} \\ &= \frac{\prod_{j=1}^n h_j \hat{e}(K_i^{\sum_{j=1}^n r_j}, g)^t}{\prod_{j=1}^n h_j \hat{e}(K_i, g)^{\sum_{j=1}^n r_j} \hat{e}(K_i, g)^{\sum_{j=1}^n r_j} t} \\ &= \frac{\hat{e}(\prod_{j=1}^n h_j, g)^t}{\prod_{j=1}^n h_j \hat{e}(K_i, g)^t} = m \end{aligned}$$

- b. Calculate the hash value V on received C_1, C_2 and computed m in the previous step.

$$V = H_1(C_1, C_2, m)$$
- c. Finally user U_i compares V with C_4 , accepts the message m if valid. Otherwise rejected the message.

A. Joining / Mass Joining.

When a user U_j wants to join in the group communication first, he has to register with the group controller. Then Group Controller will check for the validity of the user by following step 1(a&b). After checking the validity, On the input security parameter l , the Group Controller (U_0) generates a random tuple $(G_1, G_T, \hat{e}, g, q)$. The two multiplicative groups G_1 and G_T with the prime order q , where G_1 is generated by g and $\hat{e}: G_1 \times G_1 \rightarrow G_T$. The GC also selects $K_1^1, K_2^1, \dots, K_{n+1}^1 \in G_1$, where $n+1$ is number of users after join in the new session for group communication, a random master key $K^1 \in Z_q^*$ and sets $g = g_1^{K^1}$ and also chooses hash functions $H_1\{0,1\}^* \rightarrow G_1$. Finally, GC publishes the system parameters list $\pi^1 = (G_1, G_T, \hat{e}, g, q, K_1^1, K_2^1, \dots, K_{n+1}^1, H_1)$. Once the system parameters were published, the users in the group communication will again choose a new private key generation (step 2) and follows the remaining steps for key agreement, group encryption key generation, and group decryption key generation. The step 6 encryption algorithm is used for encrypting the plain text, uses step 7 to decrypt the received cipher text.

B. Leaving / Mass Leaving.

When a user U_j wants to leave from the group communication, he has to inform the group controller. Then Group Controller will check for the validity of the user by following step 1(a&b). After checking the validity, On the input security parameter l , the Group Controller (U_0) generates a random tuple $(G_1, G_T, \hat{e}, g, q)$. The two multiplicative groups G_1 and G_T with the prime order q , where G_1 is generated by g and $\hat{e}: G_1 \times G_1 \rightarrow G_T$. The GC also selects $K_1^{11}, K_2^{11}, \dots, K_{n-1}^{11} \in G_1$, where $n-1$ is number of users after join in the new session for group communication, a random master key $K^{11} \in Z_q^*$ and sets $g = g_1^{K^{11}}$ and also chooses hash functions $H_1\{0,1\}^* \rightarrow G_1$. Finally, GC publishes the system parameters list $\pi^{11} = (G_1, G_T, \hat{e}, g, q, K_1^{11}, K_2^{11}, \dots, K_{n-1}^{11}, H_1)$.

IV. SECURITY ANALYSIS

In this section, proposed protocol is furnished with all the security requirements defined in group key security requirements section.

1. **Known key Security:** In each session, U_i randomly chooses different h_i and r_i , which are used in group encryption and decryption key generation in that session. Hence, the group encryption and decryption keys are used in the current session is independent of other sessions. Therefore, the leakage of group decryption key(s) in one session should not derive the group decryption key(s) in other session, so known key security property is satisfied.
2. **Key compromise impersonation:** We can say a key agreement protocol is key compromise impersonate is compromising a participant A's private key, it will allows the active adversary to impersonate the only A not other than A. So, if an adversary wants to impersonate a participant C other than A, he should know the private key of the user C to generate the valid signature. Hence, impossible for the adversary to forge a participant C, to whom he does not know the private key. Therefore, our protocol has the key compromise impersonate property.
3. **Key control:** In our protocol, all participants are taking a role in the generation of common encryption key. It is easy to see that no gp member
4. **Unknown key share:** Each participant in our protocol should generate their signature () on M_i . Therefore, other users can verify the authentication of the user U_i . This means that the protocol is successful in checking of all the participants are real users or faked users. That result, only true participants should be allowed in group key negotiation.
5. **Group key secrecy:** The active group members in the session, who are involved in key establishment by publishing their individual parameters onto other. Even, the passive adversary knows the part of the parameters he cannot estimate the common group encryption key, because each user is contributing their random parameter (h_i) in the calculation of A. Even the passive adversary knows all the A_i 's of all active users, without knowledge of user own contribution he cannot estimate the decryption key(d_i). Therefore, Group key secrecy is achieved.
6. **Perfect forward secrecy:** without the knowledge of users random contribution h_i and r_i of U_i the adversary should break the K-BDHE assumption. Hence, even the adversary collects the private keys of all the group participants, the adversary cannot recover the parameter $_{i,i}$. Since the adversary does not know $_{i,i}$ cannot derive the previous decryption keys and the protocol is satisfying the perfect forward secrecy.
7. **Perfect Backward secrecy:** As the master secret key (K) is regularly updated with group updating. without the knowledge of users random contribution h_i and r_i of U_i the adversary should break the K-BDHE assumption. Hence, even the adversary collects the private keys of all the group participants, the adversary cannot recover the parameter $_{i,i}$. Since the adversary does not know $_{i,i}$ cannot derive the future decryption keys and the protocol is satisfying the perfect backward secrecy.

V. Performance Evaluation

Our scheme is constructed from one rounded asymmetric group key management scheme, used ZSS scheme in signature generation and L.Zhang [13] encryption scheme. The proposed system is evaluated by comparing it with AGKA by Qin.B[4], Robust authentication protocol by L. Zhang[13], ASGKA for open networks by Qianhong Wu[5] and constant round key by R. Dutta[6]. The performance comparison of our scheme is listed in the following table III and table IV. To establish a session key, most of the existing protocols require two or more rounds. Therefore, all the participants in the group should connect concurrently. Our protocol also requires two rounds, one is for a group set up

and the other is for group common key derivation. In our protocol, the master secret key is derived from the contributions of group members, but in other protocols the GC himself decides the master key. Our protocol is providing all the features of [5] and additionally supporting the mass join/leaving in the group. The comparison shows that our proposal is comparable to scheme with identical structure [5] on the performance, needs additional computation (C_4) at the encryption. The limitation with the [5] protocol is limitation in board size, the user is allowed to join when a free spaced is existed on board II, otherwise the user request for join is denied. But, our protocol is dynamically allowing single / multiple users to join in the group, because the when any group member is join /leaving the group, he should first register with the Group Controller. After that, the U_0 is generating a new master key and again the entire algorithm is executed.

TABLE III: FEATURES COMPARISON WITH PREVIOUS WORK

	[4]	[6]	[12]	[5]	Our scheme
Constant Rounds	Yes	Yes	Yes	Yes	Yes
Dynamic single join/leave	No	Yes	Yes	Yes	Yes
Mass join	No	No	No	No	Yes
Authenticated scheme	Yes	Yes	Yes	Yes	Yes
Asymmetric scheme	Yes	No	Yes	Yes	Yes
Broadcasting supported	Yes	Yes	No	Yes	Yes

TABLE IV: PERFORMANCE COMPARISON WITH PREVIOUS WORK

Feature	[4]	[6]	[12]	[5]	Our scheme
Structure	Line	Circle	Circle	Line	Line
No of rounds	1	2	3	1	2
Computation cost at key agreement	$(n-1)sign+(n-1)ver+2nM+nM_T$	$2sign+(n+1)ver+2nM+3E$	$3sign+2nver+(n-1)M+4E$	$(n-1)sign+(n-1)ver+2nM+nM_T$	$(n-1)signver+2nM+nM_T$
Exponentiation	0	3	4	0	0
Computation cost at join	0	$(m+5)ver+(2m+4)M+2E$	$(n+2m-1)ver+(w+n-1)M+1E+1sign$	$mver+(n+m-1)M+1sign$ $m<(n+1)$	$mver+(n+m-1)M+1sign$
Computation cost at leave	0	$(n-m)ver+(2w-2n-1)M+1sign$	$\leq (n+m-1)ver+(n+m-1)M+1E+1sign$	$\leq mver+(n+m-1)M+1sign$	$\leq mver+(n+m-1)M+1sign$
Transmission cost at key agreement	$nG+G_T$	$2G+2\sigma$	$6Zq+5G$	$nG+G_2$	$nG+G_T$
Transmission cost at Join	0	0	$1Zq+1G$	$2mG+1Zq$	1G
Transmission cost at Leave	0	$(w-n-1)G$	1G	$mG+1Zq$	1G
Storage overhead	-	-	-	$(n+1)^2$	1

n: the total number of users participated in the group communication

m: the number of users who want to join/leave the group

G: element in the G.

G_T : element in the G_T

P: pairing in the G.

Zq: element in the Zq

M: Multiplication/Division in G

E: Exponentiation in G

M_T : Multiplication/Division in G_T .

Sign: signature signing operation

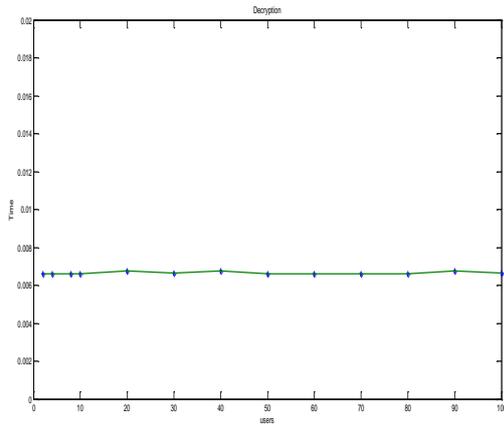
Ver: signature verification operation

In protocol [6], sender and participant have to verify $n+1$ signature and compute $2n-2$ multiplications and 3 exponentiation operations. In our protocol, the sender has to generate group encryption keys, verify n signatures and compute $2n$ group multiplication operations. Participant needs to verify $n-1$ signatures and performs $2n$ multiplications. This computation overhead is less compared to protocol in [6].

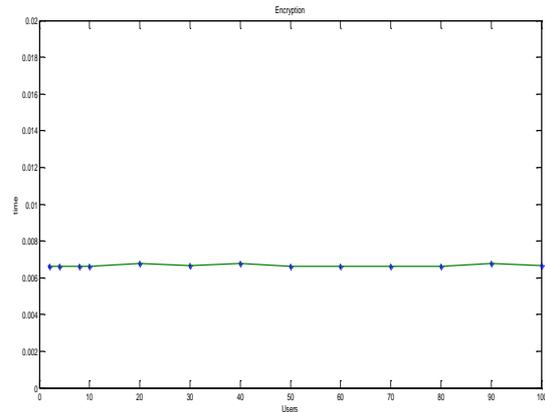
VI. Simulation

In this section, the simulation arrangement is explained to determine the performance of the above protocol. In simulation, we run the program using pairing based cryptography library [2] on the desktop using Intel(R)Core(TM) i5-2400 CPU@3.10GHZ, frequency 3.09 GHz and 2.91 GB of RAM. We vary the number of participants from 3 to 100 by setting the security parameter to be l, length of the group elements in G1 and GT are set to 171 and 1024 bits respectively.

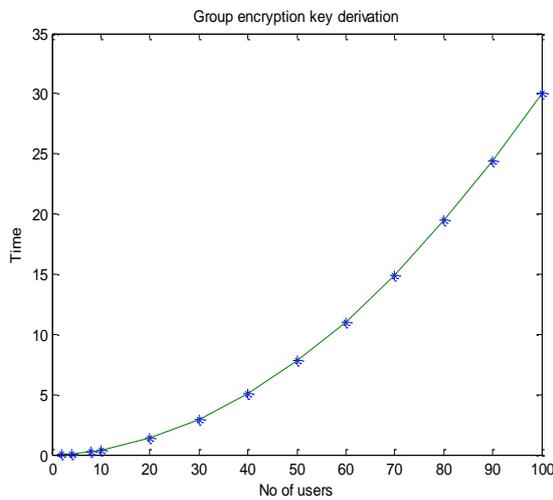
The ZSS[16] short signature scheme is selected for securing the protocol and having a length of bits. The average computation time for key establishment, common group encryption key derivation and decryption key derivation by is shown in the Fig 1.



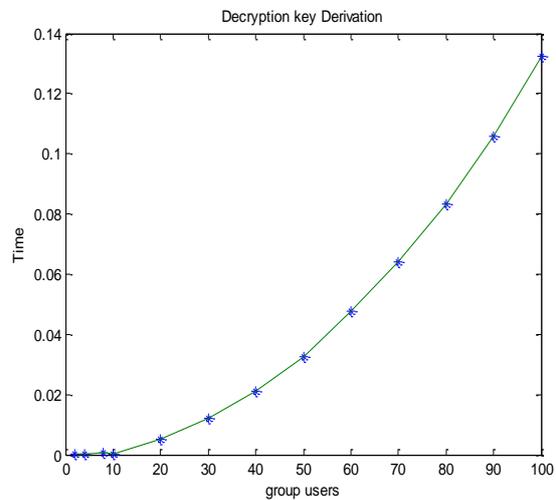
(a) For Message Encryption



(b) For Message Decryption



(c) Group Encryption key



(d) individual Decryption key

Fig 1: Average running time calculation

From the figure1(c), we can see that the time costs for the sender and participant to generate a group encryption key is linearly increasing as the number of participants in the group is increasing. We can also see that the time cost for a participant to generate an individual user group decryption key is ranging between 0.1 ms to 13 ms shown in fig 1(d). The time costs of generating cipher text and decrypting a cipher text are constant regardless of increasing in the group members is shown in fig 1(a) and fig 1(b).

VII. Trade off Between Communication and Ciphertext Size

In A²GKA protocol, each participant should have the transmission overhead of O(n) to publish group elements during group key establishment. To reduce the transmission overhead, a group can be further divided into subgroups. For instance, we can set the subgroup size to be log₂(n). The participants in each subgroup can run the proposed protocol to generate their sub group encryption key and decryption keys. By this, the user in the subgroup will generate log₂n encryption keys and will have one sub group decryption key. To send an encrypted message to all the other group members, he has to encrypt the message separately for each subgroup using the corresponding sub group encryption key, and generates the final cipher text by concatenating all the inherent individual ones. To decrypt the cipher text, the subgroup members should extract the part of the cipher text from the full ciphertext, and then decrypts the extracted sub ciphertext using their subgroup decryption key. With this approach the sender has to send a secret message to only w user at a cost of log₂w group elements during encryption.

We noticed that, above technique is similar to that in [15]. The technique in [15] is focused on broadcast encryption scheme, usually not suitable for group key agreement protocols. In group key agreement protocol if we divide the group into sub groups, each participant in the subgroup can run the conventional group key agreement protocol. Then, all the subgroups will have their own group encryption key and the decryption key. By using these keys, the users in the same subgroup will communicate with each other. However, in an ASGKA protocol any user in the sub group, who knows the group encryption key, can send the encrypted message to other subgroup. The other subgroup participants can

use their decryption keys to decrypt the received cipher text. Hence, adapting the technique [15] will reduce the trade-off in communication overhead.

VIII. Conclusion

We have proposed two rounded Authenticated Asymmetric Group Key agreement protocols which capture all the desirable security attributes of key agreement protocols. Scalable group key agreement is allowed in the proposed protocols. This protocol uses the bilinear map technique in common group key generation, which provides security against active as well as passive attacks. Evaluation of our protocols has more computation overhead and allows the group member to broadcast his contribution to other group members. For every group update (user join/leave) the group controller is changing the public parameters, which are used in the group public and private key generation, resulting in the provision of perfect forward secrecy and backward secrecy. Our system also attains perfect forward security as well as backward secrecy.

References

- [1] NS-2 Network Simulator. <http://www.nsnam.org/>
- [2] Pairing-Based Cryptography Library. <http://crypto.stanford.edu/pbc/>
- [3] Joux, A. "One Round Protocol for Tripartite Diffie-Hellman". *Journal of Cryptology* 17(4), pp. 263-276, 2004.
- [4] Wu, Q., Mu, Y., Susilo, W., Qin, B., Domingo-Ferrer, J.: "Asymmetric Group Key Agreement". In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 153–170. Springer, Heidelberg (2009)
- [5] Lei Zhang, Qianhong Wu, Bo Qin, Josep Domingo-Ferrer, Ursula Gonzalez-Nicolas "Asymmetric Group Key Agreement Protocol for Open Networks and Its Application to Broadcast Encryption", Elsevier September 18, 2011
- [6] R. Dutta, R. Barua. "Constant Round Dynamic Group Key Agreement". ISC 2005, LNCS 3650, pp. 74-88, Springer, Heidelberg, 2005.
- [7] M. Burmester, Y. Desmedt. "A Secure and Efficient Conference Key Distribution System." EUROCRYPT 1994, LNCS 950, pp. 275-286, Springer, Heidelberg, 1995.
- [8] Diffie, W., Hellman, M.: "New Directions in Cryptography". *IEEE Transactions on Information Theory* 22(6): 644-654 (1976)
- [9] K. Choi, J. Hwang, D. Lee. "Efficient ID-based Group Key Agreement with Bilinear Maps". PKC 2004, LNCS 2947, pp. 130-144, Springer, Heidelberg, 2004.
- [10] J. Katz, M. Yung. "Scalable Protocols for Authenticated Group Key Exchange". CRYPTO 2003, LNCS 2729, pp. 110-125, Springer, Heidelberg, 2003.
- [11] H. Kim, S. Lee, D. Lee. "Constant-Round Authenticated Group Key Exchange for Dynamic Groups." ASIACRYPT 2004, LNCS 3329, pp. 245-259, Springer, Heidelberg, 2004.
- [12] L. Zhang, B. Qin, Q. Wu, F. Zhang, "Efficient many-to-one authentication with certificateless aggregate signatures", *Computer Networks* 54(14)(2010) 2482-2491.
- [13] L. Zhang, Q. Wu, A. Solanas, J. Domingo-Ferrer. "A Scalable Robust Authentication Protocol for Secure Vehicular Communications". *IEEE Transactions on Vehicular Technology* 59(4), pp. 1606-1617, 2010.
- [14] D. Boneh, X. Boyen, E. Goh. "Hierarchical Identity Based Encryption with Constant Size Ciphertext." EUROCRYPT 2005, LNCS 3494, pp. 440-456, Springer, Heidelberg, 2005.
- [15] D. Boneh, C. Gentry, B. Waters. "Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys." CRYPTO 2005, LNCS 3621, pp. 258-275, Springer, Heidelberg, 2005.
- [16] F. Zhang, R. Safavi-Naini, W. Susilo, "An Efficient Signature Scheme from Bilinear Pairings and Its Applications," *Practice and Theory in Public Key Cryptography -- PKC'2004*, Singapore(SG), March 2004, *Lecture Notes on Computer Science* 2947, Springer-Verlag (2004), pp. 277--290.
- [17] F. Bao, R. Deng, H. Zhu. "Variations of Diffie-Hellman Problem." ICICS 2003, LNCS 2836, pp. 301-312, Springer, Heidelberg, 2003.
- [18] S. Blake-Wilson, D. Johnson, A. Menezes. "Key Agreement Protocols and Their Security Analysis." *Cryptography and Coding*, LNCS 1355, pp. 30-45, Springer, Heidelberg, 1997.
- [19] M. Burmester, Y. Desmedt. "A Secure and Efficient Conference Key Distribution System." EUROCRYPT 1994, LNCS 950, pp. 275-286, Springer, Heidelberg, 1995.
- [20] C. Mitchell, M. Ward, P. Wilson. "Key Control in Key Agreement Protocols." *Electronic Letters* 34(10), pp. 980-981, 1998.