International Journal of Computer Applications (0975 - 8887)
Volume 171 - No.3, August 2017

A Provable Secure and Escrow-able Authenticated Group Key Agreement Protocol without NAXOS Trick

Shaheena Khatoon
School of Studies in Mathematics
Pt.Ravishankar Shukla University
Raipur - 492010 (C.G.), India

Tejeshwari Thakur
School of Studies in Mathematics
Pt.Ravishankar Shukla University
Raipur - 492010 (C.G.), India

Balwant Singh Thakur
School of Studies in Mathematics
Pt.Ravishankar Shukla University
Raipur - 492010 (C.G.), India

ABSTRACT

We present an ID-based escrow-able authenticated group key agreement (AGKA) protocol which is provably secure in random oracle model. Additionally, the proposed protocol neither involve NAXOS trick nor uses gap assumption. And the security is proven in stronger eCK model. To our best knowledge, the proposed protocol will be first provable Secure and escrow-able ID based authenticated group key agreement protocol without NAXOS trick in eCK model.

Keywords

Group key agreement, identity based, escrow-able, NAXOS trick

1. INTRODUCTION

Authenticated group key agreement(AGKA) protocol provides the participant with a shared group key which is used in different collaborative and group oriented application, to achieve confidentiality as well as authentication. Mostly, group application communication take place over a public(open) network which are usually insecure network. So,a strongly secure AGKA protocol is essential requirement for a group oriented applications. The first group key agreement was proposed by Ingemarsson [15] in the year 1982. After, that a number of group key agreement has been proposed and their security has been extensively studied.Bellary and Rogaway in the year 1993 were first to give the formal security model for key agreement protocols popularly known as BR model. Then there has been several modification and extension to the BR model [2], [3], [8] etc. The most note-able one is the CK model given by CanettiKrawczyk [8], later it was extended as eCK model by LaMacchia et al [16].
But, there exist two big problem in the security proof of the models:(1) Most of these models use Gap assumption [20](artificial oracle) in their security analysis. But,the gap assumption is not practical at all, as it is not possible to construct such a decisional oracle in the real world. Cash et al [10] gave solution to this problem they define the twin Deffie-Hellman problem whose core is the trapdoor test. The trapdoor technique enabled us to implement decisional oracle without use of artificial oracle (2) LaMacchia et al [16] used NAXOS trick to hide the exponent of ephemeral public key, but it is leaked by side-channel attack. Many of the secure AGKA protocol uses NAXOS trick. Hence, should be avoided as recommend by [19] and [22].

So, we propose an AGKA protocol without using gap assumption and NAXOS trick. Further, the proposed protocol is an ID-based escrow-able AGKA protocol. As discussed in [18], the key escrow is desirable property in some close group application where audit trial is legal requirement. An ID-based encryption was first given by Boneh and Franklin [4], since then it is widely used for its simplicity. Hence, to our best knowledge we present a strongly secure ID based AGKA protocol without gap assumption and NAXOS trick in stronger eCK model.

2. OUR CONTRIBUTION

We present an ID-based escrow-able authenticated group key agreement (AGKA) protocol which is provably secure in random oracle model. Additionally, the proposed protocol do not involve NAXOS trick. The proposed protocol has the following features:

—we use the trapdoor test [10] and [13] for security analysis which avoids the Gap Bilinear Diffie Hellman (GBDH) assumption which weakens the security assumption.

—The protocol is based on Computational Bilinear Diffie Hellman (CBDH) assumption in the considerably strong eCK model.

—Most of the key agreement protocols [12], [21] and [23] use signatures for mutual authentication, which considerably degrades the performance of the protocol.The Proposed protocol do not use signature for mutual authentication.

—The proposed protocol is escrow-able and has a better performance especially when we consider certain pre-computations which can be performed off line.

—To our best knowledge this protocol is the first provably secure id based AGKA protocol in eCK model without using gap assumption and NAXOS trick.

3. PRELIMINARIES

The present section briefly defines some of the properties of the bilinear pairing related mathematical problems and the trapdoor test.
Bilinear Pairing: Let $\langle G_1, + \rangle$ be a cyclic additive group generated by P, whose order is a prime p and $\langle G_2, . \rangle$ be a cyclic multiplicative group of the same order p. A bilinear pairing e is a map defined by $e : G_1 \times G_1 \rightarrow G_2$ and have the following properties:

(1) Bilinear: This means that, for given $(P, Q) \in G_1$, $e(aP, bQ) = e(P, Q)^{ab}$, for any $a, b \in {Z_p}^*$.
(2) Non-degenerate: This means that, there exists $(P, Q) \in G_1$ such that $e(P, Q) \neq 1$, where 1 is the identity of G_2.
(3) Computability: This means that, there is an efficient algorithm to compute $e(P, Q)$ for all $(P, Q) \in G_1$.

The discrete logarithm problem (DLP) is hard in both G_1 and G_2. Two pairing are extensively used for cryptographic use. They are Weil pairing and their modification and Tate pairing. A full description can be found in [5], [6],[7] and [9]. Define $BDH(aP, bP, cP) = e(P, P)^{abc}$, where $a, b, c \in Z_q$.

Computational Bilinear Diffie-Hellman (CBDH) Assumption: For any probabilistic polynomial time (PPT) algorithm A,

$$Pr[A(q, G_1, G_2, P, aP, bP, cP) = BDH(aP, bP, cP)] \leq \epsilon(k)$$

, where $\epsilon(k)$ is negligible and value k denotes the security parameter. The probability is taken over the coin tosses of A, the choice of q, P and the uniformly random choices of $a, b, c \in Z_q$. And the advantage of an algorithm C in solving the CBDH problem to be the probability that, given input $aP, bP, cP \in G_1$, C returns $BDH(aP, bP, cP)$.

Trapdoor Test:[10] Given a bilinear pairing $e : G_1 \times G_1 \rightarrow G_2$, where G_1 and G_2 are same as described above. Suppose $B_1 \in G_1, y, z \in Z_q$ are randomly chosen independent variables and, define $B_2 = yP - zB_1$. Further, randomly choose $A, C \in G_1$ and $T_1, T_2 \in G_2$ where each of these are some function of $B_1 and B_2$.Then we have:

—B_2 is distributed uniformly over G_1
—B_1 and B_2 are independent.
—If $B_1 = b_1P$ and $B_2 = b_2P$, then the probability that the truth value of

$$T_1^z.T_2 = e(A, C)^y$$

does not agree with the truth value of

$$T_1 = e(A, C)^{b_1} and T_2 = e(A, C)^{b_2}$$

is at most $1/q$, moreover, if equality (2) holds, then equality (1) certainly holds.

For, further explanation for trapdoor technique reader are referred to [10], [13], [17].

4. SECURITY MODEL

4.1 Desirable Security Attributes

An escrow-able ID-based AGKA protocol should fulfill the following security property.Let $U_i, 1 \leq i \leq n$ be the set of n user in the group agreement protocol.

(1) Basic impersonation resilience: An adversary A can impersonate any user only if it knows its private key.
(2) Known-key security (K-KS): Each time the protocol is executed, it will produce a unique and independent secret session key. The compromise of one session key should not affect the secrecy of keys established in other sessions.
(3) Forward secrecy (FS): Forward secrecy is define in three ways: (1) Partial forward secrecy: compromising of some of the participants long-term private keys should not disclose session keys established previously; (2)Perfect forward secrecy (PFS): compromising of all long-term private keys of the user in the group should not disclose session keys previously established; (3) KGC forward secrecy (KGCFS): compromising the master private key of the KGC should not disclose session keys previously established(while KGCFS implies PFS in the ID-based setting,it is a particular property defined for ID-based AKA protocols in the escrow-less mode). However, if the adversary is actively involved with the choice of the DH values X, Y at a session, no two-message AKA protocol can achieve full forward secrecy, according to the result of HMQV [14]. Thus, we use the weak form of full forward secrecyweak forward secrecy (WFS) which can be applied to practice.
(4) Key-compromise impersonation(K-CI)resilience: If an adversary knows the private key of any user than it can impersonate only that identity but not any other user in the group.
(5) Unknown key-share (UK-S) resilience: Any entity U_i cannot force into sharing a key with any entity U_k while U_i believes that he is sharing the key with another entityU_i.
(6) No key control: No user in the group can force the session key (or any portion of the session key) to any preselected value.
(7) Ephemeral secrets reveal resistance (ESRR): The protocol should prevent the leakage of ephemeral secrets of any user. However, if the ephemeral secret is leaked it should not compromise the security of sessions where the ephemeral secret was not used.

4.2 The Security Model

We adopt the original eCK model given by LaMacchia et al [16] in ID based setting from original PKI based setting. Let $U = ID_1, ..., ID_n$ be a set of user in participating in key agreement with each party ID_i representing a probabilistic polynomial-time (PPT) Turing machine. The protocol executes between any two of these parties. Each party ID_i can execute finite number of protocol instances (sessions) at a time. Public key for each party is derived from its identity string. The Key Generation Center (KGC) generates static private keys of the parties through a secure channel. The adversary A is also modeled as a PPT Turing machine has full control of the network over which the user communicates, and at his will he can eavesdrop, delay, replay, alter and insert messages at will. $\prod_{i,j}^{T}$ denotes the T-th protocol session between the user ID_i(the owner) and ID_j(the peer).A session $\prod_{i,j}^{T}$ enters an accepted state when it computes a session key $sk_{i,j}^{T}$. It should be noted that a session may be terminate without ever entering into an accepted state. The information that a session has terminated with acceptance or without acceptance is public. The session $\prod_{i,j}^{T}$ assigned a partner ID $prid = (ID_i, ID_j)$.Here,comms represents the transcript of the messages exchanged between the owner and the peer during the session. Two sessions $\prod_{i,j}^{T}$ and $\prod_{i,j}^{W}$ are said to be matching if they have the same comms (and prid). The game is executed in two phases. In the first phase , the adversary A can issue the following query in any order:

(1) Ephemeral Secret Reveal (ESR): An adversary get control to the owner party $ID_i's$ ephemeral secret for any session $\prod_{i,j}^{T}$. Ephemeral secret is usually get revealed if secret information is stored in any insecure memory device or if the random number generator of the party is corrupted.
(2) Session Key Reveal: In this case it reveals the accepted session key, otherwise if the session is not accepted it outputs empty string. And the session is called open against which the adversary issues this query. Note that Session Key Reveal query is

valid if none of the following conditions hold: (1)$\prod_{i,j}^{T}$ does not exist or it is not accepted. (2) Revealing the session key of $\prod_{i,j}^{T}$ breaks the freshness of the test session chosen by a challenger C. If either of the first two conditions hold, C returns $\perp$. If the last condition holds, C aborts the game.

(3) Static Key Reveal: An adversary get access to party $ID_i's$ static private key.

(4) Establish Party(ID_i): In this query an adversary can arbitrarily register as a legal user on behalf of any party ID_i. So that the adversary can completely control the party ID_i. A party is called honest if the adversary does not issue this query to him.

(5) Send query: The adversary sends the message m to party ID_i executing session $\prod_{i,j}^{T}$ on behalf of party ID_i and get a response according to the protocol specification. This query let adversary to order ID_i to start session $\prod_{i,j}^{T}$ with ID_j and to provide communications from ID_j to ID_i.

Once the adversary A decides that the first phase is over, it starts the second phase by choosing a fresh session $\prod_{i,j}^{T}$ and issuing a Test query, where the fresh session and test query are defined as follows:
Freshness: A session namely, $\prod_{i,j}^{T}$ is said to be fresh if:(1) User ID_i and ID_j are honest. (2) Session $\prod_{i,j}^{T}$ has accepted and is not opened.(3) There is no opened session $\prod_{j,i}^{W}$ which has a matching conversation to $\prod_{i,j}^{T}$ (4) None of the following conditions hold:

—ID_j is engaged in session $\prod_{j,i}^{W}$ matching to $\prod_{i,j}^{T}$ and the adversary A reveals either both the static private key of ID_i and the ephemeral secret of $\prod_{i,j}^{T}$ or both the static private key of ID_j and the ephemeral secret of $\prod_{j,i}^{W}$.

—No session matching to $\prod_{i,j}^{T}$ exists and the adversary A reveals either the static private key of ID_j, or both the static private key of ID_i and the ephemeral secret of $\prod_{i,j}^{T}$

Test query: The session $\prod_{i,j}^{T}$ must be fresh. A bit $b \in \{0,1\}$ is randomly selects. If $b = 0$, the adversary A is given the session key, otherwise a uniformly chosen random value from the distribution of valid session keys is returned to A.Adversary A can issue test query only once in a game. After Test query on the session $\prod_{i,j}^{T}$ has been issued, the adversary can continue querying except with the condition that the test session $\prod_{i,j}^{T}$ remains fresh. At the end of the game, the adversary outputs a guess b'. If $b' = b$,the adversary wins.The adversary's advantage in winning the game is defined as

$$Adv_A(k) = |Pr[Awins] - 1/2|.$$

Secure AGKA Protocol: We say a AGKA protocol is secure(in the eCK model) if matching sessions compute the same session keys and for any adversary A has the negligible advantage in winning the above game.

It should be noted here that the above model does not support the adversarys Master Private Key Reveal queries since revealing the master private key makes anyadversary able to trivially break an ID-based AGKA protocol in the escrow mode.

5. AN ID-BASED ESCROW-ABLE AGKA PROTOCOL

System Setup: Let k be the security parameter and KGC be the key generation center. The KGC chooses two groups G_1 and G_2 of prime order q, an admissible pairing $e : G_1 \times G_1 \to G_2$ and a random generator P of G_1. It also defines three cryptographically secure hash functions $H : \{0,1\}^{*2} \times G_1^2 \times G_1^2 \to H_1, H_2 : \{0,1\}^* \to G_1$ Then KGC chooses $s \in Z_q^*$ as the master secret key (MSK) and computes $P_{pub} = sP$ as its public key. The system parameters params are published as $\{G_1, G_2, e, P, P_{pub}, H, H_1, H_2\}$.
Extract: On input params and identifier $ID \in \{0,1\}$, KGC computes the private key of ID as $\langle S_{ID}^{(1)} = sQ_{ID}^{(1)}, S_{ID}^{(2)} = sQ_{ID}^{(2)} \rangle$ where $Q_{ID}^{(1)} = H_1(ID), Q_{ID}^{(2)} = H_2(ID)$. Then KGC communicates $\langle S_{ID}^{(1)}, S_{ID}^{(2)} \rangle$ secretly and public key are $\langle Q_{ID}^{(1)}, Q_{ID}^{(2)} \rangle$.
Key Agreement: We assume each user u_i pre-computes and store following non-interactive share secrets:

$$K_i^1 = e(Q_j^1, S_i^1)$$

$$K_i^2 = e(Q_j^2, S_i^2)$$

$$T_i^1 = e(Q_j^1, P_{pub})$$

$$T_i^2 = e(Q_j^2, P_{pub})$$

for all $i = 1, 2, ...nandi \neq j$ Assume that $u_1, u_2, ..., u_n$ is a group of members who will establish a group session key. The protocol is as follows:
Round 1: Each user i chooses random $r_i \in Z_q^*$ computes and broadcasts $P_i = r_i P$ where $i = 1, 2, ...n$.
Round2: Each u_i verifies $P_j \in G_1^*$ if so u_i chooses randomly $k_i \in \{0,1\}^k$ and computes $H(k_i)$, $M_{i,j}^1 = (K_i^1)[e(P_J, P_{pub})]^{r_i}[T_i^1]^{r_i} e(P_i, S_i^1) = e(P_j + Q_j^1, r_i P_{pub} + S_i^1)$, $M_{i,j}^2 = (K_i)^2[e(P_J, P_{pub})]^{r_i}[T_i^2]^{r_i} e(P_i, S_i^1) = e(P_j + Q_j^2, r_i P_{pub} + S_i^2)$ $K_{i,j} = H(M_{i,j}^1 \| M_{i,j}^2) \oplus k_i$. Then broadcast $H(k_i), K_{i,j}, (1 \leq j \leq n, j \neq i)$.
Key Computation: on receiving $K_{j,i}$, u_i computes $k_j^{'} = H(M_{i,j}^1 \| M_{i,j}^2) \oplus K_{j,i}$ and verifies $H(k_j) = H(k_j^{'})$, if verification fails u_i aborts, otherwise it sets $sid_i^v = P_1 \| P_2 \| .... \| P_n$ and computes the group session key as $sk_i^v = H(K_{i1} \| K_{i2} \| ... \| K_{in} \| pid_i^v \| sid_i^v)$.

Correctness: By the property of bi-linearity we can briefly check the correctness of the protocol, $M_{i,j}^1 = e(P_j + Q_j^1, r_i P_{pub} + S_i^1) = e(r_j P + Q_j^1, r_i sP + sQ_i^1) = e(P_i + Q_i^1, r_j P_{pub} + S_j^1) = M_{j,i}^1$ and $M_{i,j}^2 = e(P_j + Q_j^2, r_i P_{pub} + S_i^2) = e(r_j P + Q_j^2, r_i sP + sQ_i^2) = e(P_i + Q_i^1, r_j P_{pub} + S_j^2) = M_{j,i}^2$.
Escrow: By the property of bi-linearity we get, $M_{i,j}^1 = M_{j,i}^1 = e(P_i + Q_i^1, P_j + Q_j^1)^s$ and $M_{i,j}^2 = M_{j,i}^2 = e(P_i + Q_i^2, P_j + Q_j^2)^s$. Thus, the KGC can recover all the session keys using MSK and the public data such as $P_i's$. Hence, the protocol is escrow-able.

6. SECURITY ANALYSIS

THEOREM 1. *Assuming that H, H_1, H_2 are random oracle and for (G_1, G_2, e, P) the CBDH assumption holds then the protocol ϕ proposed in section 5 is a secure escrow able ID based AGKA protocol in eCK model.*

PROOF. In accordance with the definition 4.2, we will show that their exist no efficient PPT adversary against the proposed protocol

ϕ having a non-negligible advantage in winning the game describe in section 4.

Here, k is the security parameter. Let us assume that their exist PPT adversary A against the protocol ϕ having an advantage $Adv_A^{\phi}(k)$ in wining the game given in section 4. Let $n_s(k)$ denotes the maximum number of sessions that any user in the group may have, further A can activate at-most $n_p(k)$ honest parties and can make at-most $n_h(k)$ distinct query to the oracle H. Note that $n_s(k), n_p(k), n_h(k)$ are bounded polynomially in k and $Adv_A^{\phi}(k)$ is non-negligible. Since H is a random oracle, after making test query A has only three possibilities to differentiate between a tested session key from a random string. These possibilities are as follows:

(1) Guess attack: The adversary A can correctly guess the session key.

(2) Key replicating attack: In this attack A forces a non-matching session to posses the same group session key with the test session. By doing so, A can know the group session key by querying the non-matching session.

(3) Forging attack: If $\prod_{i,j}^{T}$ be any test session, A can query on H with at leat one value $(M_{i,j}^1 \| M_{i,j}^2), (1 \leq i,j \leq n, i \neq j.)$ in the test session owned by any ID_i communicating with some ID_j. And in this attack A calculates the values $(ID_i, ID_j, P_i, P_j, M_{i,j}^1, M_{i,j}^2)$.

Out of the three attacks mentioned above, the success probability of guess attack and key replicating attack are negligible. As we know H is a random oracle so probability of guessing the output of H is $o(\frac{1}{2^k})$ which is negligible. Further two non- matching session cannot have the same identity and the same ephemeral public key. hence the success probability of key replicating attack is also negligible. Now, we will show protocol ϕ is secure against forgery attack also.

Let C be a challenger who will use the advantage of an adversary A to differentiate between a tested session key from a random string into an advantage in solving the CBDH problem. Here, $Adv_C^{CBDH}(k)$ denotes the advantage of C in solving the CBDH problem using the adversary A. C has the instance $(aP, bP, cP) \in G_1^{*3}$ of the CBDH problem and the challenge is to compute $e(P,P)^{abc}$. C simulates the game. Before the beginning of the game C tries to guess the test session and the strategy A will adopt. For this C will first randomly choose $i, j \in \{1, ...n_p(k)\} : i \neq j$ where it represents ith and jth honest party initially chosen by A and C also selects $T \in \{1, ...n_s(k)\}$ and determine the test session $\prod_{i,j}^{T}$ which is correct with the probability greater than $\frac{1}{n_s(k)[n_p(k)]^2}$. C simulates the game in accordance with his guess and abort the game whenever it finds that it has missed the guess. Otherwise C has the following four choices in accordance with A's strategy:

(1) A neither have the knowledge of the ephemeral secret of ID_j nor he knows the static private key of ID_i.

(2) A neither have the knowledge of the ephemeral secret of ID_i nor he knows the static private key of ID_j.

(3) A neither have the knowledge of the static private key of ID_i nor of ID_j.

(4) A neither have the knowledge of the ephemeral secret of ID_i nor of ID_j.

The probability that C had guessed both the test session and the strategy of A before the beginning of game is $\frac{1}{4n_s(k)[n_p(k)]^2}$, (as A has four strategies). The group session key can be generated by querying H on any $(ID_i, ID_j, r_iP, r_j, M_{i,j}^1, M_{i,j}^2)$:$(1 \leq i,j \leq n, i \neq j)$ where,$M_{i,j}^1 = e(r_jP + H_1(ID_j), r_iP + H_1(ID_i))^s$ and $M_{i,j}^2 = e(r_jP + H_2(ID_j), r_iP + H_2(ID_i))^s$. C performs the simulation and aborts the game if the test session chosen by A does not matches with C's guess. Otherwise, C randomly choose a value $\mu \in \{0,1\}^k$ and returns it to A. C maintains empty list namely $H_1^{list}, H_2^{list} and H^{list}$ to record entries and to simulate the oracle $H_1, H_2 and H$ in the following way:

The H_1 Oracle: C simulates the H_1 oracle by using an initially empty list H_1^{list}. C randomly selects $l_i \in Z_q^*$ records it, and sets $H_1(ID_i) = l_iP$. Before the game begins, For Strategy 1 and 3, C sets $H_1(IDi) = bP$ for Strategy 2, $H_1(ID_j) = bP$ and for Strategy 3, it additionally sets $H_1(ID_j) = cP$, here bP and cP are given CBDH problem instance. As$bP, cP \in G_1^*$, this modification is indistinguishable for any adversary.

The H_2 Oracle: C simulates the H_2 oracle by using an initially empty list H_2^{list}. Instead of choosing $H_2(ID_i)$ at random from G_1, C randomly selects $y_i \in Z_q^*$ at, records it, and sets $H_2(ID_i) = yiP$.For Strategies 1-3, C randomly selects $y_{trap_1}, z \in Z_q^*$, records them, and the H_2 oracle is patched before the game starts: for Strategy 1 and 3, $H2(ID_i) = y_{trap_1}P - zbP$; for Strategy 2, $H_2(ID_j) = y_{trap-1}P - zbP$. For Strategy 3, C again randomly selects $y_{trap_2}, z^{'} \in Z_q^*$, records them,and additionally patched the H_2 oracle and sets $H_2(ID_j) = y_{trap_2}P - z^{'}cP$,here bP and cP are given CBDH problem instance. As the pre-patched values are completely re-randomized, this modification is indistinguishable for any adversary.

The H Oracle: C simulates the H oracle by using an initially empty list H^{list} with entries of the form $(ID_i, ID_j, P_i, P_j, M_{i,j}^1, M_{i,j}^2, h)$. The challenger C maintains consistency between the H oracle and Session Key Reveal queries so that it can succeed in its simulation. Further C, can not compute all necessary data to query the H oracle for a valid session key in some instances. For this, C additionally maintains an initially empty list R^{list} with entries of the form $(ID_i, ID_j, P_i, P_j, sk)$. Entries in R^{list} represents a session (and its matching session if the latter exists). If C is asked an H query, it first checks H^{list} for matching entry.If it exist C returns it to A, otherwise, C checks R^{list} for matching entry (C first check if this is a valid H query for a session key and if necessary S uses its decisional oracles that is explained later in the proof). If matching entry if found in R^{list}, C returns the matching sk to A and a new entry of the query data and the value sk is inserted into the H^{list} by C; otherwise, C chooses a value uniformly at random from the output domain of H, returns this value to A and adds a new entry of the query data and the chosen value to H^{list}. Valid Session Key Reveal query is answered in the same way, i.e. C first checks for matching entry in R^{list} for some sk if it exist, C returns sk to A; otherwise, S looks in H^{list} for matching Session Key Reveal query (C first check if this is a valid H query for a session key and if necessary S uses its decisional oracles that is explained later in the proof). If matching entry if found in R^{list}, C returns the matching sk to A and a new entry of the query data and the value sk is inserted into the H^{list} by C; otherwise,if matching is found in H^{list}, C returns the matching entry h and adds a new entry of the query data and the value h to R^{list}; otherwise, C randomly selects a value $sk \in_R \{0,1\}^k$, returns it to A and adds a new entry of the query data and the chosen value sk to R^{list}).

If ID_i and ID_j are not participating in the test session $\prod_{i,j}^{T}$, than C

handles the session key reveal query in the following way:Without loss of generality it is assumed that session is initiated by ID_i and it has the message $r_{j_{(A)}}$, here subscript $j_{(A)}$ indicate that that value may be controlled by an adversary. Due to the patched H_1 and H_2 oracles, C knows the values: $log_p H_1(ID_i) = l_i, log_p H_1(ID_j) = l_j, log_p H_2(ID_i) = y_i, and log_p H_2(ID_j) = y_j$ and C also knows $\{sH_1(ID_i) = l_i P_{pub}, sH_2(ID_i) = y_i P_{pub}\}$ and the ephemeral secret r_i of ID_i. So to obtain session key C has to query the H oracle on $(ID_i, ID_j, r_i P, r_{j_{(A)}} P), M^1_{i,j}, M^2_{i,j}$.So, the challenger C acting as the party ID_i knows the public values $ID_i, ID_j, r_i, r_{j_{(A)}}$ and can compute $M^1_{i,j} = e(r_{j_{(A)}} P + l_j P, r_i P_{pub} + l_i P_{pub})$ and $M^2_{i,j} = e(r_{j_{(A)}} P + y_j P, r_i P_{pub} + y_i P_{pub})$.

Now, we analyze the behavior of C according to the four strategies mentioned above. Without loss of generality, we assume ID_i is the initiator and any $ID_j, (1 \leq j \leq n : i \neq j)$.

Strategy 1 : For the Strategy 1, A neither have the knowledge of the ephemeral secret of ID_j nor he knows the static private key of ID_i. But, it can corrupt the static private key of ID_j and the ephemeral secret of ID_i. Now, C sets the $P_{pub} = aP$ and the patched value of H_1 and H_2 oracle gives $H_1(ID_i) = bP, H_2(ID_i) = y_{trap_1} P - zbP$. Further, C can generate the static private keys of all identities except ID_i as $sH_1(ID_i) = l_i a and sH_2(ID_i) = y_i aP, (1 \leq j \leq n : i \neq j)$ and it sets the ephemeral public key of participants $ID'_j s, (ID_j \neq ID_i)$involving in the test session $\prod^T_{i,j}$ as cP. As per the condition of strategy 1, C cannot make reveal query for $ID'_i s$ static private key, but it does not make much affect on overall success probability. Precisely, for strategy 1 challenger C guesses that the test session $\prod^T_{i,j}$ selected by it has matching conversation with some other session $\prod^W_{j,i}$ and it sets ephemeral public key of $\prod^W_{j,i}$ to cP. But, a problem occurs for C when A makes a session reveal query for some other session including ID_i and a party ID_j which may not be a honest party in the session $\prod^T_{i,j}$. To reply such a query C checks the H^{list} for the matching H query involving both ID_i and ID_j. If the matching record exist, C will use the twin bilinear Diffie-Hellman trapdoor to check the validity of H. And C construct the decisional oracle by extracting discrete logarithm $H_1(ID_j)$ and $H_2(ID_j)$ as l_i and y_i. These value are obtain from the patched H_1 and H_2 oracles. Then C extracts $M^1_{i,j}$ and $M^2_{i,j}$ from the matching session for which session key reveal query has been asked, and computes $M'^1_{i,j} = e(l_j aP, r^*_i P) e(r_j P, r^*_i aP) e(l_j aP, H_1(ID_i))$ and $M'^2_{i,j} = e(y_j aP, r^*_i P) e(r_j P, r^*_i aP) e(y_j aP, H_2(ID_i))$. Here, $r^*_i P$ is the outgoing ephemeral public key of session $\prod^T_{i,j}$ from C and $r_j P$ is the incoming ephemeral public key of session $\prod^T_{i,j}$ and verifies that,

$$\left(\frac{M^1_{i,j}}{M^2_{i,j}}\right)^z \cdot \frac{M'^1_{i,j}}{M'^2_{i,j}} = e(aP, r_j P)^{y_{trap_1}}.$$

Similarly, if C is asked a new H query involving ID_i and ID_j, C uses the above decisional oracle to check the validity of H query which should be answered with a matching record from R^{list} (if such a record exists).Otherwise, C randomly selects a value $sk \in_R \{0,1\}^k$, returns it to A and adds a new entry of the query data and the chosen value sk to R^{list}).

As the attack that the adversary A mounts is Forging attack, if A succeeds, it must have queried oracle H with inputs of the correct form $M^1_{i,j} = e(P_i + H_1(ID_i), r_j sP + sH_1(ID_j)) = e(P_i + H_1(ID_i), caP + aH_1(ID_j))$ and $M^2_{i,j} = e(P_i + H_2(ID_i), r_j sP + sH_2(ID_j)) = e(P_i + H_2(ID_i), caP + aH_2(ID_j))$, where P_i is the outgoing ephemeral public key of the test session $\prod^T_{i,j}$ from C and $r_j P = cP$ is the incoming ephemeral public key of the test session$\prod^T_{i,j}$. To solve the CBDH problem for all entries in H^{list}, C randomly chooses one entry of $M^1_{i,j}, M^2_{i,j}$ and proceeds as follows: C first computes $M''^1_{i,j} = M^1_{i,j}/(e(P_i, l_j aP) e(H_1(ID_i), l_j aP)) = e(P_i + H_1(ID_i), caP)$ and $M''^2_{i,j} = M^2_{i,j}/(e(P_i, y_j aP) e(H_2(ID_i), y_j aP)) = e(P_i + H_2(ID_i), caP)$. After this C computes $\bar{M}_{i,j} = M''^2_{i,j}/M''^1_{i,j} = e(H_2(ID_i) - H_1(ID_i), caP) = e(y_{trap_1} P - zbP - bP, caP)$; and $\hat{M}_{i,j} = (\bar{M}_{i,j}/e(cP, aP)^{y_{trap_1}})^{\frac{-1}{z+1}} = e(cP, aP)^b = e(P, P)^{abc}$; then C gives $\hat{M}_{i,j}$ as the solution of the CBDH problem instance. The success probability of C is lower bounded by

$$Adv^{CBDH}_C(k) \geq \frac{Adv^{\phi}_A}{4n_s(k)[n_p(k)]^2 n_h(k)},$$

where $n_h(k)$ is the polynomial bound on the number of distinct H queries made by the adversary A.

Strategy 2: Strategy 2 is same as the strategy 1 (only ID_i and ID_j are exchanged) so it has the same probability as for strategy 1.

Strategy 3: A neither have the knowledge of the static private key of ID_i nor of ID_j. But, can corrupt the ephemeral secret of ID_i and of ID_j. To use the adversary A to solve the CBDH problem instance (aP, bP, cP), C sets $P_{pub} = aP$ and patches the H_1 and H_2 oracle as $H_1(ID_i) = bP, H_1(ID_j) = cP, H_2(ID_i) = y_{trap_1} P - zbP$ **and** $H_2(ID_j) = y_{trap_2} P - z^{'} cP$. Again C, uses the same technique as in strategy 1 to handle session key reveal and H queries both of which involve $ID_i (or ID_j)$. But, problem for C occurs when arises A raise Session Key Reveal queries for other sessions including both ID_i and ID_j than the test session $\prod^T_{i,j}$, or makes H queries involving both ID_I and ID_j. In all these conditions C is unable to compute $M^1_{i,j}$ and $M^2_{i,j}$, as C do not have the knowledge of static private key of ID_i nor of ID_j. Without loss of generality,assume that adversary A can activates a session $\prod^W_{j,i}$ with the outgoing ephemeral public key $\hat{P}_j = \hat{r_j} P$ generated by C and the incoming ephemeral public key $\hat{P}_i = \hat{r_i} P$ that may be generated by A. We claim that A has negligible probability for correctly generating the values $M^1_{i,j} = e(\hat{P}_i + H_1(ID_i), \hat{r_j} aP + aH_1(ID_j))$ and $M^2_{i,j} == e(\hat{P}_i + H_2(ID_i), \hat{r_j} aP + aH_2(ID_j))$. To proof our claim, we will show how to construct a CBDH problem solver if the adversary A queries these values correctly as follows:firstly compute:

$$\check{M}^1_{i,j} = \frac{M^1_{i,j}}{e(\hat{P}_i + H_1(ID_i), \hat{r_j} aP)}$$

$$= e(\hat{P}_i + H_1(ID_i), aH_1(ID_j)) = e(\hat{P}_i + bP, acP)$$

and and

$$\check{M}^2_{i,j} = \left(\frac{M^2_{i,j}}{e(\hat{P}_i + H_2(ID_i), \hat{r_j} aP) e(\hat{P}_i + y_{trap_1} P zbP, y_{trap_2} aP) e(y_{trap_1} aP, z\, cP)}\right)^{\frac{1}{z}}$$

$$= \left(\frac{e(\hat{P}_i + H_2(ID_j), aH_2(ID_j)}{e(\hat{P}_i + y_{trap_1} P zbP, y_{trap_2} aP) e(y_{trap_1} aP, z\, cP)}\right)^{\frac{1}{z}}$$

$$= \left(\frac{e(\hat{P}_i + y_{trap_1} P zbP, z\, acP)}{e(y_{trap_1} aP, z\, cP)}\right)^{\frac{1}{z}}$$

$$= e(\hat{P}_i - zbP, -z\, acP)^{\frac{1}{z}}$$
$$= e(\hat{P}_i - zbP, acP);$$

then BDH (aP, bP, cP) can be obtained by computing:

$$\left(\frac{\tilde{M}^1_{i,j}}{\tilde{M}^2_{i,j}}\right)^{\frac{1}{z+1}} = \left(\frac{e(\hat{P}_i + bP, acP)}{e(\hat{P}_i - zbP, acP)}\right)^{\frac{1}{z+1}}$$
$$= e(bP + zbP, acP)^{\frac{1}{z+1}}$$
$$= e(bP, acP)$$
$$= e(P, P)^{abc}.$$

This contradicts the CBDH assumption. Thus A cannot generate a valid H query for other sessions including both ID_i and ID_j than the test session (except with negligible probability). Now, having this conclusion, C can answer the adversarys queries easily: it handles such new queries involving both ID_i and ID_j in a usual way as described above. As the attack that the adversary A mounts is Forging attack, if A succeeds, it must have queried oracle H with inputs of the correct form $M^1_{i,j} = e(P_j + H_1(ID_j), r_i aP + aH_1(ID_i))$ and $M^2_{i,j} = e(P_j + H_2(ID_j), r_i aP + aH_2(ID_i))$, where P_i is the outgoing ephemeral public key of the test session $\prod^T_{i,j}$ from C and $r_j P = cP$ is the incoming ephemeral public key of the test session$\prod^T_{i,j}$. To solve the CBDH problem for all entries in H^{list}, C randomly chooses one entry of $M^1_{i,j}, M^2_{i,j}$ and proceeds as follows: C first computes:

$$M^{*1}_{i,j} = \frac{M^1_{i,j}}{e(P_j + H_1(ID_j), r_i aP)}$$
$$= e(P_j + H_1(ID_j), aH_1(ID_i))$$
$$= e(P_j + cP, abP)$$

and

$$\frac{M^2_{i,j}}{} = \left(\frac{M^2_{i,j}}{e(P_j + H_2(ID_j), r_i aP)e(P_j + y_{trap_2}{}^{P - z\, cP, y_{trap_1}} aP)e(y_{trap_2}{}^{aP, -zbP})}\right)^{\frac{1}{z}}$$
$$= \left(\frac{e(P_j + H_2(ID_j), aH_2(ID_i))}{e(P_j + y_{trap_2}{}^{P - z\, cP, y_{trap_1}} aP)e(y_{trap_2}{}^{aP, -zbP})}\right)^{\frac{1}{z}}$$
$$= \left(\frac{e(P_j + y_{trap_2}{}^{P - z\, cP, a(y_{trap_1}{}^{P - zbP}))}}{e(P_j + y_{trap_2}{}^{P - z\, cP, y_{trap_1}} aP)e(y_{trap_2}{}^{aP, zbP})}\right)^{\frac{1}{z}}$$
$$= \left(\frac{e(P_j + y_{trap_2}{}^{P - z\, cP, -zabP})}{e(y_{trap_2}{}^{aP, -zbP})}\right)^{\frac{1}{z}}$$
$$= e(P_j - z\, cP, -zabP)^{\frac{1}{z}}$$
$$= e(P_j - z\, cP, abP)$$

then BDH (aP, bP, cP) can be obtained by computing:

$$M^* = \left(\frac{M^{*1}_{i,j}}{M^{*2}_{i,j}}\right)^{\frac{1}{z'+1}} = \left(\frac{e(P_j + cP, abP)}{e(P_j - z'cP, abP)}\right)^{\frac{1}{z'+1}}$$
$$= e(cP + z'cP, abP)^{\frac{1}{z'+1}}$$
$$= e(cP, abP)$$
$$= e(P, P)^{abc},$$

then C gives M^* as the solution of the CBDH problem instance. The success probability of C is lower bounded by

$$Adv^{CBDH}_C(k) \geq \frac{Adv^{\phi}_A}{4n_s(k)[n_p(k)]^2 n_h(k)},$$

where $n_h(k)$ is the polynomial bound on the number of distinct H queries made by the adversary A.

Strategy 4: A neither have the knowledge of the ephemeral secret of ID_i nor of ID_j. But, can corrupt the static private key of both ID_i and ID_j. To use the adversary A to solve the CBDH problem instance (aP, bP, cP), C sets $r_i P = bP, r_j P = cP$ and $P_{pub} = aP$ in the test session $\prod^T_{i,j}$ and patches the H_1 and H_2 oracle as $sH_1(ID_i) = l_i aP, sH_2(ID_i) = y_i aP$. Then simulates the queries in the same way as in the previous strategies. As the attack that the adversary A mounts is Forging attack, if A succeeds, it must have queried oracle H with inputs of the correct form $M^1_{i,j} = e(cP + H_1(ID_j), baP + aH_1(ID_i))$ and $M^2_{i,j} = e(cP + H_2(ID_j), baP + aH_2(ID_i))$, where P_i is the outgoing ephemeral public key of the test session $\prod^T_{i,j}$ from C and $r_j P = cP$ is the incoming ephemeral public key of the test session$\prod^T_{i,j}$. To solve the CBDH problem for all entries in H^{list}, C randomly chooses one entry of $M^1_{i,j}, M^2_{i,j}$ and proceeds as follows: C first computes $\tilde{M}^1_{i,j} = M^1_{i,j}/(e(cP + l_j P, l_i ap)e(H_1(bP, l_j aP))y_j aP))$; then C gives $\tilde{M}^1_{i,j}$ as the solution of the CBDH problem instance. The success probability of C is lower bounded by

$$Adv^{CBDH}_C(k) \geq \frac{Adv^{\phi}_A}{4n_s(k)[n_p(k)]^2 n_h(k)},$$

where $n_h(k)$ is the polynomial bound on the number of distinct H queries made by the adversary A.
Thus all the above strategies establishes the theorem. □

Now, we will show that the proposed protocol fulfills all the security property discussed in the section 4: (1)Unknown key share (UK-S) resilience: The key derivation function H resist the UK-S attack since every time it takes as input the identity of a user in the group it would output different key hence prevents UK-S attack. (2) Weak perfect forward secrecy (WPFS): The analysis of strategy assures WPFS. (3) Basic impersonation resilience: An attacker can not mount this attack as it must know the private key of participants to compute $M^1_{i,j}, M^2_{i,j}, (1 \leq i, j \leq n)$ which are inputs to the key derivation function H. (4) Key-compromise impersonation (K-CI) resilience: Except strategy 3, rest of the strategy are resilient to (K-CI) attack.(5) Partial ephemeral secrets reveal resistance (ESRR): Strategy 1 and 2 provides partial security to ephemeral secrets leakage, and strategy 3 is fully secure to ephemeral secrets leakage.(6) Known-key security (K-KS): The definition of Secure AKA Protocol ensure the security against Known-key attack.(7) No key control: Since in the security analysis the key derivation function H is modeled as a random oracle, the proposed protocol is secure against key control attack. In addition to this protocol is secure against key control attacks launched by an outside adversary.Otherwise, if manipulating message can control the session key bits, the outside adversary A must have a non-negligible ability to distinguish between the session key held by the test session and a randomly chosen value from the session key space.

7. EFFICIENCY COMPARISON

This section present the efficiency comparison of the proposed scheme with the existing schemes [21] and [23]. The following notations will be used:
From the above comparison table, we note that protocols [21] and [23] are signature based, so for comparison we assume that theses protocol adopted the signature scheme give in [11]. The signature

Comparison with other protocols	
Notations	Description
Round	The total number of rounds
Com	The total number of Computation
M	Modular multiplication in Z_q^*
E	The cost of modular exponentiation
P	The cost of pairing computation.
S	The cost of signing a message.
V	The cost of verification.
n	The number of group members.

Comparison with other protocols			
Protocol	Round	Sign	Com
[23]	2	yes	$(4n-1)E+2nS+(2n^2-2n)V=(2n^2+4n-1)E+2nM+(2n^2-2n)P$
[21]	3	yes	$3nS+(2n^2-n)V=(2n^2+n)E+3nM+(2n^2-2n)P$
Proposed	2	no	$(n^2+n)M+(2n^2-2n)P$

scheme needs one scalar multiplication and one paring computation in the signing process; and one modular exponentiation and one pairing computation in the verification process. Thus, we can see that the cost of [21] and [23] equals $(2n^2+n)E+3nM+(2n^2-2n)Pand(2n^2+4n-1)E+2nM+(2n^2-2n)P$, respectively. The proposed protocol do-not involve signature scheme, additionally protocol [23] adopts NAXOS technique which does not resist side channel attack. Therefore, the proposed protocol is more secure than the protocols [21] and [23] . The proposed protocol achieves the desired security requirements in the eCK model without involving NAXOS technique, and it achieves mutual authentication without adopting signatures.

8. CONCLUSION

Key escrow is an essential in situations where confidentiality and audit trail are legal requirement. In this paper we present a provable secure and escrow-able authenticated group key agreement protocol without NAXOS trick in eCK model. With the help of the trapdoor test technique, we have reduced the security of proposed protocol to the standard CBDH assumption in the random oracle model. Under the assumption that master private key is secure, we have shown that our protocol is secure as long as each party has at least one un compromised secret. To the best of our knowledge, our scheme is the first escrow able authenticated group key agreement protocol without NAXOS trick in eCK model.

9. ACKNOWLEDGEMENT

The authors would like to express their sincere thanks to the referees for their valuable comments on this article.

10. REFERENCES

[1] M. Bellare, P. Rogaway, Entity authentication and key distribution, in: CRYPTO 1993, in: LNCS, vol. 773, Springer-Verlag, 1994, pp. 232249.

[2] M. Bellare, P. Rogaway, Provably secure session key distribution: the three party case, in: Proceedings of STOC 1995, ACM, 1995, pp. 5766.

[3] M. Bellare, D. Pointcheval, P. Rogaway, Authenticated key exchange secure against dictionary attacks, in: EUROCRYPT 2000, in: LNCS, vol. 1807, Springer-Verlag, 2000, pp. 139155.

[4] D. Boneh, M. Franklin, Identity-based encryption from the Weil pairing, in: CRYPTO 2001, in: LNCS, vol. 2139, Springer-Verlag, 2001, pp. 213229.

[5] D. Boneh, B. Lynn, H. Shacham, Short signatures from the Weil pairing, in: Proc. ASIACRYPT 2001, in: LNCS, vol. 2248, Springer-Verlag, 2001, pp. 514532.

[6] P. Barreto, H. Kim, B. Lynn, M. Scott, Efficient algorithms for pairing-based cryptosystems, in: Proc. CRYPTO 2002, in: LNCS, vol. 2442, Springer-Verlag, 2002, pp. 354368.

[7] D. Boneh and M. Franklin, Identity-based encryption from the weil pairing, SIAM Journal on Computing, vol. 32, no. 3, pp. 586615, 2003.

[8] R. Canetti, H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels, in: EUROCRYPT 2001, in: LNCS, vol. 2045, Springer-Verlag, 2001, pp. 453474.

[9] L. Chen, Z. Cheng, and N. P. Smart, Identity-based key agreement protocols from pairings, International Journal of Information Security, vol. 6, no. 4, pp. 213241, 2007.

[10] Cash D, Kiltz E, Shoup V. The twin DiffieHellman problem and applications. In: Proceedings of the EUROCRYPT 2008. LNCS, vol. 4965. Springer-Verlag; 2008. p. 12745.

[11] Florian H. Efficient identity-based signature schemes based on pairings. In: Proceedings of the ACM Symposium on Applied Computing, Newfoundland, 2002. 310324

[12] Gorantla M C, Boyd C, Nieto J M G. Modeling key compromise impersonation attacks on group key exchange protocols. In: Proceedings of 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, 2009. 105123.

[13] Huang H, Cao Z. An ID-based authenticated key exchange protocol based on bilinear DiffieHellman problem. In: Proceedings of the ACM ASIACCS 2009. ACM; 2009. p. 33342.

[14] H. Krawczyk, HMQV: a high performance secure DiffieHellman protocol, in: CRYPTO 2005, in: LNCS, vol. 3621, Springer-Verlag, 2005, pp. 546566.

[15] Ingemarsson I, Tang D T, Wong C K. A conference key distribution system. IEEE Trans Inf Theory, 1982, 28: 714720

[16] B.A. LaMacchia, K. Lauter, A. Mityagin, Stronger security of authenticated key exchange, in: ProvSec 2007, in: LNCS, vol. 4784, Springer-Verlag, 2007,pp. 116.

[17] G. Lippold, C. Boyd, J.G. Nieto, Strongly secure certificateless key agreement, in: Pairing 2009, in: LNCS, vol. 5671, Springer-Verlag, 2009, pp. 206230.

[18] N. McCullagh, P.S.L.M. Barreto, A new two-party identity-based authenticated key agreement, in: CT-RSA 2005, in: LNCS, vol. 3376, Springer-Verlag, 2005, pp. 262274.

[19] Moriyama D, Okamoto T. An eCK-secure authenticated key exchange protocol without random oracles. In: Proceedings of the Provable Security Conference, Guangzhou, 2009. 154167.

[20] T. Okamoto, D. Pointcheval, The gap-problems: a new class of problems for the security of cryptographic schemes, in: PKC 2001, in: LNCS, vol. 1992, Springer-Verlag, 2001, pp. 104118.

[21] Ruxandra F O. Provable secure constant-round group key agreement protocol based on secret sharing. In: Proceedings of International Joint Conference SOCO13-CISIS13-ICEUTE13, Salamanca, 2013. 489498.

[22] Ustaoglu B. Comparing session state reveal and ephemeral key reveal for Diffie- Hellman protocol. In: Proceedings of Provable Security Conference, Guangzhou, 2009. 183197

[23] Zhao, Jianjie, Dawu Gu, and M. Choudary Gorantla. "Stronger security model of group key agreement." Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security. ACM, 2011.