Ad Hoc Networks 9 (2011) 767–787

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Provably secure hybrid key agreement protocols in cluster-based wireless ad hoc networks

Ratna Dutta *, Tom Dowling

Computer Science Department, National University of Ireland, Maynooth, Co. Kildare, Ireland

ARTICLE INFO

Article history:
Received 13 March 2009
Received in revised form 22 April 2010
Accepted 30 August 2010
Available online 1 October 2010

Keywords:
Clustering
Provable security
Wireless ad hoc networks
Group key agreement
Key distribution

ABSTRACT

Wireless ad hoc networks support rapid on-demand and adaptive communication among the nodes due to their self-configurable and autonomous nature and lack of fixed infrastructure. Security is a crucial factor for such systems. Since ad hoc networks rely on the collaboration principle, the issue of key distribution and efficient group key management in such networks represents two of the most important problems. We describe hybrid solutions to the problem of key distribution and key management by reflecting ad hoc networks in a topology composed of a set of clusters. To date no security proofs exist for these types of protocols. We present two dynamically efficient schemes. We show that both our hybrid schemes are provably secure in the standard model under Decision Diffie–Hellman (DDH) assumption. The proposed protocols avoid the use of a trusted third party (TTP) or a central authority, eliminating a single point of attack. We analyse the complexity of the schemes and differentiate between the two approaches based on performance in a wireless setting. In comparison with the existing cluster-based hybrid key agreement protocols, our proposed approaches individually provide better performance in terms of both communication and computation, handle dynamic events efficiently, and are supported by sound security analysis in formal security models under standard cryptographic assumptions.

1. Introduction

Ad hoc wireless networks offer anytime-anywhere networking services for infrastructure-free communication over the shared wireless medium. In this setting, secure group key agreement and efficient group key management are considered challenging tasks from both an algorithmic and computational point of view due to resource constraint in wireless networks. There are a wide variety of applications of wireless ad hoc networks in many areas ranging from military applications, emergency, law enforcement, rescue missions and other collaborative applications for commercial uses. Security is one of the most crucial factors for such systems. Designing communication efficient secure group key agreement protocols in wireless ad hoc networks has attracted significant attention due to popularity and increased security concerns of wireless ad hoc networks. There are quite a number of group key agreement protocols described in the literature [6,7,10,13,14,16,17,24,28,34]. However, traditional group key agreement protocols are not applicable in ad hoc networks. The major challenges in providing secure authenticated communication for wireless ad hoc networks come from the following unique features of such networks:

(i) lack of a fixed reliable public key infrastructure,
(ii) dynamic network topology due to high mobility and joining/leaving devices,
(iii) energy and resource constrained nodes with limited storage, communication and computation power,

* Corresponding author. Tel.: +353 1 708 4595; fax: +353 1 708 3848.
E-mail addresses: ratna.dutta@gmail.com, rdutta@cs.nuim.ie (R. Dutta), tdowling@cs.nuim.ie (T. Dowling).

1570-8705/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2010.08.021

(iv) lack pre-distributed symmetric keys shared between nodes,
(v) high-level of self-organization,
(*vi*) vulnerable multi-hop wireless links, etc.

One possible solution to achieve ubiquitous computing in such wireless networks is to enable wireless nodes to operate in an ad hoc mode and self-organize themselves into a cluster-based network architecture. An application of such an architecture is the creation of Virtual Research Clusters in a multidisciplinary geographical setting. Researchers from different backgrounds (*e.g.* Mathematics, Computer Science, Engineering, etc.) can come together and cluster around an idea such as wireless ad hoc network security. Each cluster is lead by a designated professor, called sponsor, who is an expert in the research area associated with that cluster and is able to communicate with sponsors of other clusters. Researcher within a cluster can communicate with each other and also with the sponsor of that cluster. As the cluster develop, new researchers may enter or existing ones leave. At some stage, the idea will become sensitive, so security will be implemented and future enter/leave will be subject to our protocols. This concept has obvious applications in the business world where projects are created and cancelled at great speed. Privacy, authenticity, integrity and availability are four fundamental security issues which must be addressed depending on application specific requirements.

Clustering is a method that enables nodes to be organized based on their relative proximity to one another. Mobile nodes that are within the communication range of each other can communicate directly, whereas the nodes that are far apart have to rely on intermediate nodes to relay messages. This dynamic and multi-hop nature of ad hoc network makes the security one of the most important implementation issue apart from efficiency. A general approach to build up a cluster-based network architecture is to design efficient algorithms to organize wireless nodes into set of clusters. Many clustering algorithms have been proposed to minimize the cluster maintenance overhead, thereby reduce the waste of the precious bandwidth and also saves the consumption of the limited battery power. A detail survey of the clustering algorithms can be found in [40]. We assume ad hoc networks consist of nodes which have no prior contact, trust or authority relation and which may move freely and communicate with other nodes via wireless links. While all nodes are identical in their capabilities, certain nodes are elected to form the sponsors which are vested with the responsibility for the resource assignments, cluster maintenance, and of routing messages for all the nodes within their clusters. Sponsors typically communicate with sponsors of other clusters. The election of sponsors has been a topic of many papers as documented in [3,4,18]. The general idea among the related literature is to select sponsors based on some attributes of the networks, such as node degree, link delay, transmission power, and mobility. There are several mission critical applications (such as in military, emergency, rescue missions), scientific explorations (such as in environmental monitoring and disaster response), civilians (such as in law enforcement, building automation) and other collaborative applications for commercial uses where sponsors have a powerful radio which other nodes in the cluster do not have so that sponsors can communicate among themselves. For example, military networks consist of mobile devices carried by soldiers, automatic weapons, sensing devices, etc. In this setup, a platoon commander may play the role of sponsor and may be able to communicate with platoon commanders of (all/some) other clusters. On the other hand, soldiers are cluster mobile nodes which are able to communicate with the soldiers (and platoon commanders) within their own clusters and may move from one cluster to another.

Group key agreement and key management are easier to handle inside each cluster compared to the entire ad hoc network. This is because of the fact that clusters have more stable internal connections due to the larger amount of links between nodes within the same cluster. Besides, inter-cluster key agreement is more sensible as clusters are assumed to stay together longer than the nodes do in average for wireless ad hoc networks. Clustering may thus bring the necessary scalability into key establishment in very large networks.

1.1. Our contribution

The main contribution in this paper is to obtain two provably secure dynamically efficient authenticated hybrid key agreement protocols, namely AHP-1 and AHP-2 for cluster based wireless ad hoc networks, which are proven to be secure under Decision Diffie–Hellman (DDH) assumption. In a mobile ad hoc environment, the number of nodes could be very large. We divide all the nodes into clusters based on their relative proximity to one another. We differentiate between two types of keys. By cluster key we mean the key generated among all the nodes within a cluster and by session key we mean a common network key among all the nodes in the system. The aim is to generate a session key common to all the nodes in the network. This shared session key can later be used by all the nodes in the network to perform efficient symmetric encryption such as DES [31] and AES [30] for secure and faster communication among themselves.

The basic idea of our constructions are the followings: All nodes within a cluster dynamically generate their cluster keys *using a scalable constant round dynamic group key agreement protocol*. We choose the constant round multiparty dynamic key agreement protocol of Dutta–Barua DB [15] for this purpose, which is a variant of Burmester–Desmedt protocol BD-I [9] and handles dynamic operations very efficiently. Each cluster selects a sponsor using a sponsor selection mechanism. In our first protocol AHP-1, the sponsors form nodes in a spanning tree. Adjacent nodes in this tree will generate a secret key among themselves using 2-party Diffie–Hellman (DH) key agreement. This key used with an appropriate symmetric encryption scheme will be used to distribute the root session key (generated by the root sponsor in the tree) between them. In this way, each sponsor node of each cluster will obtain the root session key. On obtaining the root session key each sponsor will distribute it to the other cluster nodes in its own cluster using the cluster key and an appropriate

symmetric encryption scheme. On the contrary, in our second protocol AHP-2, all the sponsors will generate a common group session key using multi-party DB protocol. Each sponsor will then distribute this group session key to the other cluster members in its own cluster using the cluster key and an appropriate symmetric encryption scheme.

We call our protocols hybrid because these protocols are combinations of key agreement and key distribution via symmetric encryption. We authenticate our protocols using digital signature schemes with appropriate modifications in Katz–Yung [23] compiler as described in [15]. In keeping with the dynamic nature of mobile ad hoc networks, these protocols facilitate efficient handling of elements leaving and joining clusters. Our proposed protocols are designed without using computation-intensive pairings and are highly efficient as compared to the existing cluster-based hybrid key agreement protocols [2,20,25,26,36,39]. In contrast of invoking the DB protocol from scratch among all the nodes in the system (which may be infeasible in ad hoc environments), we obtain efficiency gain in terms of both communication and computation for most of the nodes in our proposed cluster-based protocols (which are more amenable to wireless ad hoc networks).

To the best of our knowledge, our proposals are the *first* attempt in the context of cluster-based hybrid key agreement to provide security analysis in formal security model under standard cryptographic assumptions and also handling dynamic membership change efficiently to make these protocols suitable for ad hoc wireless network with resource constrained mobile nodes. We emphasize our first hybrid key agreement protocol AHP-1 from a performance point of view, especially in handling dynamic operations as compared to our second hybrid key agreement protocol AHP-2. On a more positive note, the network for protocol AHP-1 is more flexible in the context of ad hoc network as compared to protocol AHP-2, as it uses a tree-based network of sponsors and reconstruction of this tree is done rarely as sponsors do not leave the group very frequently.

1.2. Previous work

Li et al. [26] first propose a communication efficient hybrid key agreement protocol using the concept of connected dominating set. However, the protocol is inefficient in handling dynamic events. Based on this work, Yao et al. [39] propose a hierarchical key agreement protocol which is communication efficient in handling dynamic events. Both the protocols of [26,39] employ some existed group key agreement protocols and are unauthenticated with no security analysis against active adversary. There are a few proposals for key agreement protocols [2,25,36] in wireless ad hoc networks using pairing which are not computationally very efficient. The pairing-based key agreement protocols proposed in [25,36] employ Joux's [22] tripartite protocol. While [36] uses a ternary tree structure with leaf nodes as users, [25] uses the concept of binary cluster trees with each cluster having 3 (or 2) nodes where internal nodes in the tree are also users. Protocols in [25] are more efficient than protocol in [36]. However, handling dynamic events for [36] is easier whereas it is difficult for [25], especially to handle leave operations. Abdel-Hafez et al. [2] provide another pairing-based key agreement protocol that uses clusters of arbitrary size and requires a trusted authority. However the protocol is not efficient in terms of both communication and computation. We should mention here another work by Hietalahti [20] who presents a solution that uses BD protocol [9] within each cluster and then invokes AT-GDH protocol [19] by employing a spanning tree of sponsors. Both BD and AT-GDH protocols are static and consequently handling dynamic events are not easy for this scheme. The security against active adversary for all the above protocols are argued only heuristically.

1.3. Organization

In Section 2 we present the background for the main ideas in the paper. Section 3 introduces the hybrid protocols and details their components. For the ease of understanding, we first describe the unauthenticated versions of our protocols. We then show how to transform these unauthenticated protocols into authenticated protocols. Section 4 is concerned with the security analysis following standard security framework under standard cryptographic assumptions. Proofs of security for unauthenticated and authenticated versions are dealt with here. This section is the main contribution of the paper. Section 5 considers efficiency issues and Section 6 compares our protocols with the existing cluster-based similar protocols. We present conclusions in Section 7.

2. Preliminaries and background protocols

We first define DDH problem. We use two-party DH [11] protocol and multi-party DB [15] protocol in designing our hybrid key agreement protocols. We describe these protocols in the following two subsections. These protocols are secure under the assumption that DDH problem is hard. We adapt a security model for key agreement protocols based on Bresson et al.'s [8] formal security model. We also use a symmetric encryption scheme and adapt the find-and-guess security notion of symmetric encryption following [1]. These security notions are also presented in this section. Let G be a finite multiplicative group of some large prime order q where the well-known Discrete Logarithm (DL) problem is believed to be intractable and g be a generator of G. Also consider a hash function $\mathcal{H} : \{0,1\}^* \rightarrow Z_q^*$.

2.1. Decision Diffie–Hellman (DDH) problem

The Decision Diffie–Hellman (DDH) problem on G is defined as follows (The notation $a \leftarrow S$ denotes that a is chosen randomly from S):

Instance: (g^a, g^b, g^c) for some $a, b, c \in Z_q^*$.
Output: **yes** if $c = ab \bmod q$ and output **no** otherwise.

We consider two distributions as:

$$\Delta_{\mathsf{Real}} = \left\{a, b \leftarrow Z_q^*, A = g^a, B = g^b, C = g^{ab} : (A, B, C)\right\}$$
$$\Delta_{\mathsf{Rand}} = \left\{a, b, c \leftarrow Z_q^*, A = g^a, B = g^b, C = g^c : (A, B, C)\right\}.$$

The advantage of any probabilistic, polynomial-time, 0/1-valued distinguisher $\mathcal{D}$ in solving DDH problem on G is defined to be :

$$\mathsf{Adv}_{\mathcal{D},G}^{\mathsf{DDH}} = |\mathsf{Prob}[(A, B, C) \leftarrow \Delta_{\mathsf{Real}} : \mathcal{D}(A, B, C) = 1] - \mathsf{Prob}[(A, B, C) \leftarrow \Delta_{\mathsf{Rand}} : \mathcal{D}(A, B, C) = 1]|.$$

The probability is taken over the choice of $\log_g A$, $\log_g B$, $\log_g C$ and $\mathcal{D}$'s coin tosses. $\mathcal{D}$ is said to be a (t, ϵ)-DDH distinguisher for G if $\mathcal{D}$ runs in time at most t such that $\mathsf{Adv}_{\mathcal{D},G}^{\mathsf{DDH}}(t) \geqslant \epsilon$. We define $\mathsf{Adv}_G^{\mathsf{DDH}}(t) = \max_{\mathcal{D}}\{\mathsf{Adv}_{\mathcal{D},G}^{\mathsf{DDH}}(t)\}$ where the maximum is over all $\mathcal{D}$ with time complexity t.

DDH assumption: There exists no (t, ϵ)-DDH distinguisher for G. In other words, for every probabilistic, polynomial-time, 0/1-valued distinguisher $\mathcal{D}$, $\mathsf{Adv}_{\mathcal{D},G}^{\mathsf{DDH}} \leqslant \epsilon$ for any sufficiently small $\epsilon > 0$.

2.2. Protocol DH [11]

Assume that two entities A and B want to decide upon a common key. They perform the following steps.

1. User A chooses a random private ephemeral key $a \in Z_q^*$, computes $T_A = g^a$ and sends T_A to B.
2. User B chooses a random private ephemeral key $b \in Z_q^*$, computes $T_B = g^b$ and sends T_B to A.
3. User A computes $K_A = T_B^a$ and similarly user B computes $K_B = T_A^b$.

If A and B execute the above steps honestly, they will agree upon a common key $K_{AB} = K_A = K_B = g^{ab}$ (Fig. 1).

The protocol is unauthenticated in the sense that it is secure against passive adversary under DDH problem. An active adversary can mount man-in-the-middle attack and impersonation attacks. A large number of variations of Diffie–Hellman (DH) protocol has been proposed to improve its security degree.

2.3. Protocol DB [15]

Suppose a set of n users $\{U_1, \ldots, U_n\}$ wish to establish a common session key among themselves. Consider users $U_1, \ldots, U_n$ participating in the protocol are on a *virtual* ring and U_{i-1}, U_{i+1} are respectively left and right neighbours of U_i for $1 \leqslant i \leqslant n$. Here index i stands for i *mod* n and we have $U_0 = U_n$ and $U_{n+1} = U_1$. The protocol consists of three algorithms DB.Setup, DB.Join and DB.Leave for initial setup, user join and user leave respectively. The protocol is executed as follows among n users $U_1, \ldots, U_n$ (Figs. 2–4).

2.3.1. Protocol *DB.Setup*

1. In round 1, each user U_i chooses randomly a private ephemeral key $x_i \in Z_q^*$ and broadcasts $y_i = g^{x_i}$.
2. In round 2, user U_i on receiving y_{i-1} and y_{i+1} computes its left key $K_i^L = y_{i-1}^{x_i}$, right key $K_i^R = y_{i+1}^{x_i}$, $Y_i = K_i^R / K_i^L$ and broadcasts Y_i. Notice that $K_i^L = K_{i-1}^R$ is a common DH key between users U_i, U_{i-1} and $K_i^R = K_{i+1}^L$ is a common DH key between users U_i, U_{i+1}.
3. Finally, in the key computation phase, U_i computes $\overline{K}_{i+1}^R, \overline{K}_{i+2}^R, \ldots, \overline{K}_{i+(n-1)}^R$ as follows making use of its own right key $K_i^R : \overline{K}_{i+1}^R = Y_{i+1} K_i^R, \overline{K}_{i+2}^R = Y_{i+2}\overline{K}_{i+1}^R, \ldots, \overline{K}_{i+(n-1)}^R = Y_{i+(n-1)}\overline{K}_{i+(n-2)}^R$. Then U_i verifies if $\overline{K}_{i+(n-1)}^R$ is same as that of its own left key $K_i^L (= K_{i+(n-1)}^R)$. If verification fails, then U_i aborts. Otherwise, U_i has correct right keys of all the users. U_i computes the session key $\mathsf{sk} = \overline{K}_1^R \overline{K}_2^R \ldots \overline{K}_n^R \bmod q$ which is equal to $g^{x_1 x_2 + x_2 x_3 + \cdots + x_n x_1}$. U_i also computes and stores $\hat{h} = \mathcal{H}(\mathsf{sk})$ for a possible subsequent join operation and stores its left key and right key K_i^L, K_i^R respectively for a possible subsequent leave operation.

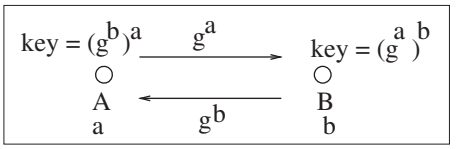

Fig. 1. Diffie–Hellman key agreement.

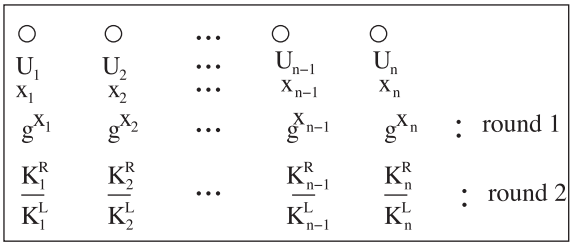

Fig. 2. Initial group key agreement (DB.Setup).

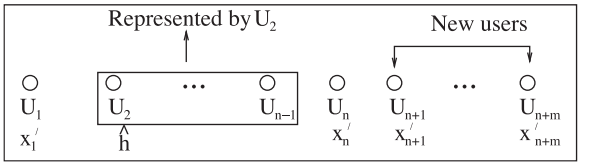

Fig. 3. Join operation (DB.Join).

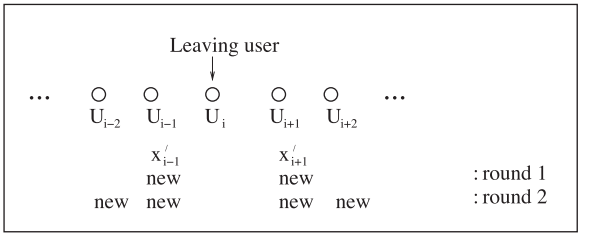

Fig. 4. Leave operation (DB.Leave).

2.3.2. Protocol DB.Join

The key agreement in Join algorithm is done in such a way that new users are unable to know previous session keys. When a set of new users $U[n+1,\ldots,n+m]$ with respective private ephemeral keys $x'[n+1\ldots,n+m]$ wants to join a group of users $U[1,\ldots,n]$ with respective new randomly chosen private ephemeral keys $x'[1,\ldots,n]$, DB.Setup is invoked among users $U_1, U_2, U[n,n+1,\ldots, n+m]$ with respective private ephemeral keys $x'_1, \hat{h}, x'[n,n+1,\ldots,n+m]$. We consider U_2 as a representative of the set $U[2,3,\ldots,n-1]$ and execute DB.Setup considering a virtual ring of $m+3$ users, instead of a virtual ring of $n+m$ users. Here $\hat{h}$ is the common seed value that users $U[1,2,\ldots,n]$ agree upon in previous session before users $U[n+1,\ldots,n+m]$ have joined. In addition, from transmitted messages, users $U[3,\ldots,n-1]$ are able to compute the current session key in the same way as U_2 does, as they know the common seed value $\hat{h}$. However, they do not participate in the communication during protocol execution, although they receive all the messages which they are supposed to receive. Here it is assumed that new members have the knowledge of who are in the system, whom to communicate with and who their immediate neighbours are on the virtual ring, *i.e.* they have the knowledge of old partner identity in the previous session before executing the Join algorithm for the current session. To handle join operations efficiently (without restarting the initial protocol), participants use the saved value $\hat{h}$ in the previous session. Thus the proposed join algorithm takes advantage of reusability of users' precomputed values in previous sessions in order to save most users' computations for updating session keys in subsequent session in which users join the group.

2.3.3. Protocol DB.Leave

The key update in Leave algorithm is performed to make the set of revoked users in a session unable to know the current and subsequent session keys. The leaving users are dropped from the virtual ring. Left and right neighbours of each leaving user choose new private ephemeral keys. Then the key agreement is done as usual in the resulting ring of users with respective private ephemeral keys by invoking DB.Setup. However, in round 1, only the first closest non-leaving left and right neighbours of a leaving participant choose new private ephemeral keys and broadcast new values, the second closest non-leaving left and right neighbours of a leaving participant broadcast precomputed values, and other users broadcast nothing. On the other hand, in round 2, only the following non-leaving neighbours of a leaving user broadcast the modified new values, while the other users broadcast the precomputed values saved in previous session:

- the *first* closest non-leaving *left* neighbour of the leaving user computes its new left key and new right key and broadcasts the modified new quotient of new right key and new left key,
- the *first* closest non-leaving *right* neighbour of the leaving user computes its new left key and new right key and broadcasts the modified new quotient of new right key and new left key,
- the *second* closest non-leaving *left* neighbour of the leaving user computes its new right key and broadcasts the modified new quotient of new right key and precomputed left key,
- the *second* closest non-leaving *right* neighbour of the leaving user computes its new left key and broadcasts the modified new quotient of precomputed right key and new left key

Here also, it is assumed that each user knows the positions of its partners on the virtual ring in the current session associated with this Leave operation. To handle leave operations efficiently (without restarting the initial protocol), most of the participants use the saved values of their left keys and right keys precomputed in the previous session, only a limited number of participants compute their new left and right keys. Thus the proposed leave algorithm takes advantage of reusability of user's precomputed values in previous sessions in order to save most user's computations for updating session keys in subsequent session in which users leave the group.

The unauthenticated DB protocol can be viewed as a variant of unauthenticated protocol of Burmester and Desmedt BD-I [9] . However, the session key computation is done differently, although with same complexity of BD-I protocol. This unauthenticated protocol is proven to be secure against passive adversary assuming the intractability of decision Diffie–Hellman (DDH) problem. Authors incorporate authentication in the unauthenticated DB protocol using digital signature with appropriate modifications in Katz–Yung [23] compiler. The authenticated DB protocol is a simplification of the protocol of [23] in which authors have done away with random nonces and have been able to reduce number of rounds by one. The security proof is actually tighter than that in Katz–Yung [23]. This is due to the fact that authors use a setting of unique instance numbers instead of random nonces and are able to avoid the event Repeat that comes in Katz–Yung security proof, thus reducing complexity of Katz–Yung compiler. They further extend this static authenticated protocol to dynamic setting by introducing algorithms for join and leave as described above. To handle these operations efficiently (without restarting the initial protocol), participants use the saved values $\hat{h} = \mathcal{H}\left(\mathsf{sk}_{U_i}^{d_i}\right), K_i^L$ and K_i^R to take advantage of reusability of user's precomputed values in previous sessions in order to save most user's computations for updating session keys in subsequent session in which users join and/or leave the group. Thus they gain in computation complexity in their dynamic protocol in contrast to executing the BD-I protocol (which is static) among the new group of users. Besides, the protocol has the ability to detect the presence of a corrupted group member, although it cannot detect who among the group members are behaving improperly. If an invalid message is sent by a corrupted member, then this can be detected by all legitimate members of the group and the protocol execution may be stopped instantly. This feature makes the DB protocol interesting when the adversarial model no longer assumes that the group members are honest.

2.4. Security model for symmetric encryption

Let $\mathcal{M} \cup \{\Phi\}$ be the message space, $\mathcal{K}$ be the key space and $\mathcal{C}$ be the ciphertext space. A symmetric encryption scheme is a pair of algorithms $\mathsf{Symm} = (\mathcal{E}, \mathcal{D})$. The encryption algorithm $\mathcal{E}$ is a probabilistic algorithm which takes as input a key $k \in \mathcal{K}$, a plaintext $x \in \mathcal{M}$ and returns a ciphertext $y = \mathcal{E}(k,x) \in \mathcal{C}$. The decryption algorithm $\mathcal{D}$ is deterministic and takes as input a key $k \in \mathcal{K}$, a purported ciphertext $y \in \mathcal{C}$ and returns a unique value $\mathcal{D}(k,y) \in \mathcal{M}$. We have $\mathcal{D}(k, \mathcal{E}(k,x)) = x$ for all $x \in \mathcal{M}$ and $k \in \mathcal{K}$. A return value of Φ from $\mathcal{D}$ is intended to indicate that the ciphertext was not the encryption of any plaintext and thus regarded as "invalid". CBC encryption and Vernam cipher encryption are examples of symmetric encryption algorithms. The following notion of security for symmetric encryption is defined as in [1].

Security (find-and-guess notion): Let $\mathcal{A}$ be an adversary that defeats the security of symmetric encryption scheme $\mathsf{Symm} = (\mathcal{E}, \mathcal{D})$. Consider the following game between $\mathcal{A}$ and a challenger.

1. At the start of the game, the challenger chooses a random key $k \in \mathcal{K}$ and gives $\mathcal{A}$ the access to the encryption oracle $\mathcal{E}(k,\cdot)$.
2. $\mathcal{A}$ runs in two stages – find and guess. $\mathcal{A}$ may query the encryption oracle $\mathcal{E}(k,\cdot)$ during either stage.
3. In the find stage, $\mathcal{A}$ outputs a pair of equal length plaintexts $x_o, x_1 \in \mathcal{M}$.
4. The challenger chooses a bit $b \in \{0,1\}$ uniformly at random, computes $y_b = \mathcal{E}(k, x_b)$ and outputs y_b.
5. In the guess stage, $\mathcal{A}$ is given y_b which is a random ciphertext for one of the plaintexts x_0, x_1 and outputs a guess bit b'.

The adversary $\mathcal{A}$ wins the game if $b = b'$. Let Succ be the event that $\mathcal{A}$ wins in the above game. We define

$$\mathsf{Adv}^{\mathsf{ind-cpa-fg}}_{\mathsf{Symm},\mathcal{A}} = |2\mathsf{Prob}[\mathsf{Succ}] - 1|$$

to be the *ind-cpa-advantage* of $\mathcal{A}$ in this find-and-guess notion of security and

$$\mathsf{Adv}^{\mathsf{ind-cpa-fg}}_{\mathsf{Symm}}(t, q_{\mathcal{E}}, \mu) = \max_{\mathcal{A}}\{\mathsf{Adv}^{\mathsf{ind-cpa-fg}}_{\mathsf{Symm},\mathcal{A}}\}$$

where the maximum is over all $\mathcal{A}$ with time complexity t, making at most $q_{\mathcal{E}}$ queries to the encryption oracle and sum of lenghts of all the encryption queries is at most μ bits.

2.5. Security model for key agreement

The following security model for key agreement is based on Bresson et al.'s [8] formal security model for group key agreement.

Let $\mathcal{P} = \{U_1, \ldots, U_n\}$ be a set of n (fixed) users or participants. At any point of time, any subset of $\mathcal{P}$ may decide to establish a session key. We identify the execution of protocols for key agreement, member(s) join and member(s) leave as different sessions. The adversarial model allows concurrent execution of the protocol and consists of allowing each user an unlimited number of instances with each instance may be used only once, and with which it executes the protocol for key agreement or inclusion or exclusion of a user or a set of users. *We assume adversary has complete control over all communication, but never participates as a user in the protocol.* We will require the following notations.

Π^i_U	ith instance of user U.
sk^i_U	session key after execution of the protocol by Π^i_U.
sid^i_U	session identity for instance Π^i_U. We set $\mathsf{sid}^i_U = S = \{(U_1,i_1),\ldots,(U_k,i_k)\}$ such that $(U,i) \in S$ and users $U_1,\ldots,U_k$ wish to agree upon a common key in a session using the unused instances $\Pi^{i_1}_{U_1},\ldots,\Pi^{i_k}_{U_k}$.
pid^i_U	partner identity for instance Π^i_U, defined by $\mathsf{pid}^i_U = \{U_1,\ldots,U_k\}$, such that $(U_j,i_j) \in \mathsf{sid}^i_U$ for all $1 \leqslant j \leqslant k$, where i_j comes from sid^i_U as defined above.
acc^i_U	0/1-valued variable which is set to be 1 by Π^i_U upon normal termination of the session and 0 otherwise.

The following oracles model an adversary's interaction with the users in the network, where S, S_1, S_2 are three sets such that $S \cap S_1 = \emptyset$ and $S_2 \subseteq S$. More precisely, let

$$S = \{(U_1,i_1),\ldots,(U_l,i_l)\}, S_1 = \{(U_{l+1},i_{l+1}),\ldots,(U_{l+k},i_{l+k})\}, S_2 = \{(U_{j_1},i_{j_1}),\ldots,(U_{j_k},i_{j_k})\}$$

where $\{U_1,\ldots,U_l\}$ is any non-empty subset of $\mathcal{P}$.:

- $\mathsf{Send}(U,i,m)$: The output of the query is the reply (if any) generated by the instance Π^i_U upon receipt of message m. The adversary is allowed to prompt the unused instance Π^i_U to initiate the protocol with partners $U_2,\ldots,U_l$, $l \leqslant n$, by invoking $\mathsf{Send}(U,i,\langle U_2,\ldots,U_l\rangle)$. This query models an active attack, in which the adversary may intercept a message and then either modify it, create a new one or simply forward it to the intended participant.
- $\mathsf{Execute}(S)$: The output of this query is the transcript of an honest execution of the key agreement protocol among unused instances $\Pi^{i_1}_{U_1},\ldots,\Pi^{i_l}_{U_l}$. A transcript consists of the messages that were exchanged during the execution of the protocol.
- $\mathsf{Join}(S,S_1)$: The output of this query is the transcript generated by the invocation of algorithm Join for the insertion of user instances $\Pi^{i_{l+1}}_{U_{l+1}},\ldots,\Pi^{i_{l+k}}_{U_{l+k}}$ in the group $\left\{\Pi^{i_1}_{U_1},\ldots,\Pi^{i_l}_{U_l}\right\}$. If $\mathsf{Execute}(S)$ has not taken place, then the adversary is given no output. This query is initiated by a send query.
- $\mathsf{Leave}(S,S_2)$: The adversary is given the transcript generated by the honest execution of procedure Leave for the removal of user instances $\Pi^{i_{j_1}}_{U_{j_1}},\ldots,\Pi^{i_{j_k}}_{U_{j_k}}$ from the group $\left\{\Pi^{i_1}_{U_1},\ldots\Pi^{i_l}_{U_l}\right\}$. If $\mathsf{Execute}(S)$ has not taken place, then the

adversary is given no output. This query is also initiated by a send query.
- Reveal(U,i) : This unconditionally outputs session key sk_U^i.
- Corrupt(U) : This outputs the long-term secret key (if any) of player U.
- Test(U,i) : A bit $b \in \{0,1\}$ is chosen uniformly at random. The adversary is given sk_U^i if $b = 1$, and a random session key if $b = 0$.

A *passive* adversary has access to Execute, Join, Leave, Reveal, Corrupt and Test oracles while an *active* adversary is additionally given access to Send oracle. For static case, there are no Join or Leave queries as a group of fixed size is considered. The adversary can ask Send, Execute, Join, Leave, Reveal and Corrupt queries several times, but Test query is asked only once and on a fresh instance.

We say that an instance Π_U^i is *fresh* unless either the adversary, at some point, queried Reveal(U,i) or Reveal(U',j) with $U' \in \mathsf{pid}_U^i$ or the adversary queried Corrupt(V) (with $V \in \mathsf{pid}_U^i$) before a query of the form Send$(U,i,*)$ or Send$(U',j,*)$ where $U' \in \mathsf{pid}_U^i$.

Finally, the adversary outputs a guess bit b'. Such an adversary is said to win the game if $b = b'$, where b is the hidden bit used by Test oracle. Let Succ denote the event that the adversary $\mathcal{A}$ wins the game for a key agreement protocol XP. We define

$$\mathsf{Adv}_{\mathcal{A},\mathsf{XP}} := |2\ \mathsf{Prob}[\mathsf{Succ}] - 1|$$

to be the advantage of the adversary $\mathcal{A}$ in attacking the protocol XP.

The protocol XP is said to be a *secure unauthenticated group key agreement* (KA) protocol if there is no polynomial-time *passive* adversary with non-negligible advantage. We say that protocol XP is a *secure authenticated group key agreement* (AKA) protocol if there is no polynomial-time *active* adversary with non-negligible advantage. In other words, for every probabilistic, polynomial-time, 0/1 valued algorithm $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A},\mathsf{XP}} < \frac{1}{M^L}$ for every fixed $L > 0$ and sufficiently large integer M For concrete security analysis, we define

$\mathsf{Adv}_{\mathsf{XP}}^{\mathsf{KA}}(t,q_E)$	maximum advantage of any passive adversary attacking protocol XP, running in time t and making q_E calls to Execute oracle.
$\mathsf{Adv}_{\mathsf{XP}}^{\mathsf{AKA}}(t,q_E,q_S)$	maximum advantage of any active adversary attacking protocol XP, running in time t and making q_E calls to Execute oracle and q_S calls to Send oracle.
$\mathsf{Adv}_{\mathsf{XP}}^{\mathsf{AKA}}(t,q_E,q_J,q_L,q_S)$	maximum advantage of any active adversary attacking protocol XP, running in time t and making q_E calls to Execute oracle, q_J calls to Join oracle, q_L calls to Leave oracle and q_S calls to Send oracle.

3. Hybrid key agreement protocols

In this section, we propose two cluster-based hybrid key agreement protocols. We first address the clustering technique for organization of the network nodes into clusters and then detail the unauthenticated versions of our protocols HP-1 and HP-2. A high-level description of these two hybrid protocols is provided in Fig. 5. Finally, we will show how to extend these protocols into authenticated protocols AHP-1 and AHP-2 respectively.

3.1. Cluster formation

Suppose we have a large multi-hop wireless ad hoc network with no fixed infrastructure such as switching centres or base stations. Each mobile node is assumed to have some computational power and an omni-directional antenna. A message sent by a node can be received by all nodes within its transmission range. On the other hand, the nodes that are far apart have to rely on intermediate nodes to relay messages. Clustering is an efficient technique of organizing the nodes in a wireless ad hoc environment. There are large variety of clustering protocols [5,32,35]. The overhead involved in a clustering technique comprises of two phases: *cluster initialization phase* and *cluster maintenance phase*. Sucec and Marsic [35] provides a theoretical upper bound $O(n\log n)$ (n is the node count) on the communication overhead incurred by a particular clustering algorithm in ad hoc networks. Clustering is also used in some routing protocols for ad hoc networks. Routing is then divided into two parts: routing within a cluster (intra-cluster) and routing between different clusters (inter-cluster). Most of the clustering protocols perform hierarchical routing among the clusters to (i) increase the robustness of routes by providing multiple possibilities for routing among the clusters, (ii) reduce the size of routing tables [5], and (iii) incurs less communication overhead for tracking mobile nodes in large multi-hop mobile wireless network [5,32].

In our proposed protocols, we use a clustering method based on [26] to (hierarchically) organize the mobile nodes based on their relative proximity to one another. Our clustering algorithm divides nodes into small groups, called clusters. We assume that every node can find a cluster to join, *i.e.* all nodes are reachable by at least one other node. Each cluster elects a cluster head which acts as a backbone node or gateway node or sponsor for cluster interaction and is responsible for establishing and organizing the cluster. This cluster head election within a cluster can be done according to node connectivity, computational power and/or extra radio facilities. For instance, the node with the highest connectivity in a given area may become the cluster head. We assume that cluster heads have more computation and communication power than the ordinary nodes. More precisely, cluster heads have an additional powerful radio to establish wireless links among themselves. The election of cluster heads has been a topic of many papers as documented in [3,4,18]. All nodes in a cluster are within direct transmission range of the cluster head and each node pair in the same cluster can also communicate in

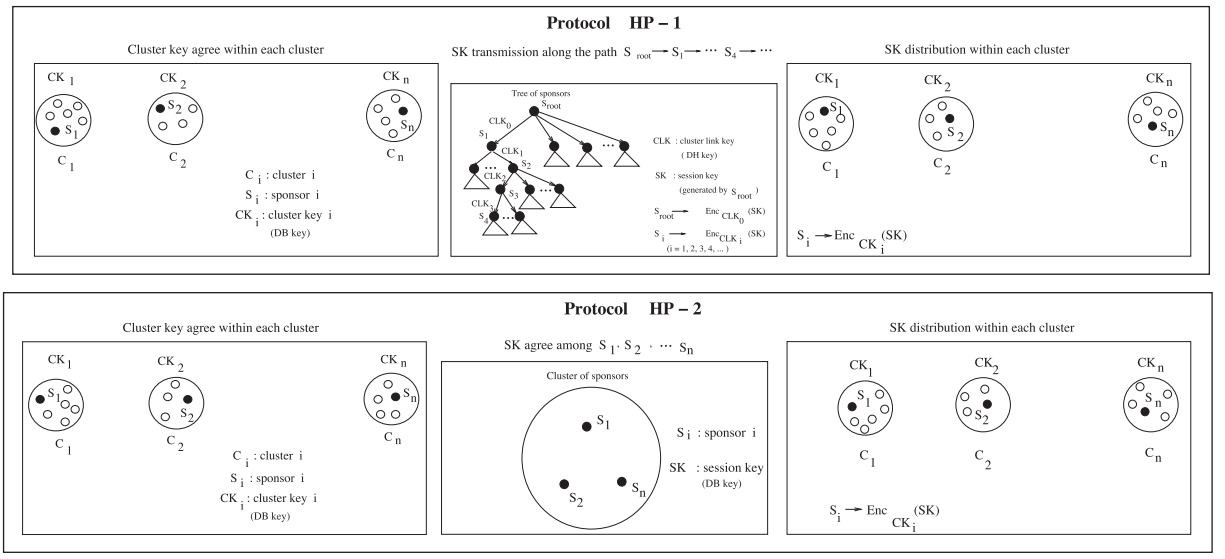

Fig. 5. Hybrid key agreement protocols ($S_i \rightarrow X$ means S_i broadcasts X).

one-hop. We briefly review below a clustering method based on [26], which consists of two phases: *cluster initialization phase* and *cluster maintenance phase*.

- *Cluster initialization phase:* This partitions an ad hoc network into a number of clusters and each node gains the knowledge of who are within its own cluster.
  (a) Each node makes its active neighbours aware of its presence by broadcasting an initial IamAlive message and its identity to its one-hop away neighbours.
  (b) Once the nodes have gathered information about their neighbours, the cluster head within a certain region should broadcast a message IamSponsor to its one-hop away neighbours to confirm its leadership within that cluster.
  (c) A node receiving IamSponsor message marks itself as an ordinary node and broadcasts IamOrdinary to all its one-hop away neighbours for confirming its inclusion in that cluster.
  (d) Each node in a cluster knows the identities of all its one-hop away neighbours. The cluster head computes a cluster-identifier, which is concatenation of identities of all the cluster nodes within that cluster.

We assume that a sponsor has the knowledge that it is a designated sponsor, election of which is based on some attributes of the network. In case a node gets several messages like IamSponsor, it processes only the message that it receives first. Thus each cluster elects exactly one sponsor and each node belongs to a unique cluster. Consequently, the clusters are disjoint in the resulting clusterization. So far, each node only has to broadcast twice to its one-hop neighbours – one for telling its identity and one for notifying its sponsor/ordinary status. Wireless nodes are not static and can move around. If the node's movement does not cause the change of the network topology, it is needless to say that no maintenance is necessary. Otherwise, the following steps are executed.

- *Cluster maintenance phase:* This enables all the nodes within a cluster to know which nodes want to join or which nodes want to leave the cluster. Thus each node is updated with the knowledge of who are within its own cluster after any membership change has occurred.

  Adding node.
  (a) Suppose a new node falls within the direct transmission range of a cluster, *i.e.* it is able to communicate with all the nodes in the cluster in one-hop. If this new node is willing to join the network, it broadcasts its identity together with a message IamAlive – thereby making aware of its presence to its one-hop away neighbours.
  (b) The cluster head broadcasts the cluster-identity and a message IamSponsor to its one-hop away neighbours to confirm its leadership within that cluster.
  (c) The new node receiving IamSponsor message marks itself as an ordinary node and broadcasts to its one-hop away neighbours IamOrdinary to confirm its inclusion in that cluster. On receiving the cluster-identity from the cluster head, the new node becomes aware of its neighbours in that cluster.
  (d) The cluster head modifies the cluster-identifier by concatenating the identity of the new user with the old cluster-identifier.

  Removing node
  (*a*) Suppose a node has moved out of the range of a cluster. The cluster head would be aware of this variation. Otherwise, when a node wants to get revoked from a cluster, it broadcasts its identity together with a message IamLeaving to all its one-hop away neighbours. The cluster head modifies the cluster-identifier by dropping the identity of the revoked node

from the concatenation of the identities (old cluster-identifier).

(b) If the cluster head is revoked, then the cluster should be reconstructed. A new cluster head need to be elected among the nodes of the cluster such that the new cluster head has the highest connectivity with additional power of computation and communication to establish wireless links among the other cluster heads in the network.

3.2. Notations

We now describe the notations for our algorithms that are used in the subsequent subsections. Algorithms 1, 2, 4, 5 and 6 use the following notations for $i = 1,\ldots,n$.

C_i	ith cluster,
S_i	sponsor node in C_i,
CK_i	cluster key agreed among all nodes of C_i,
$\mathcal{E}_K(\cdot)$	an encryption function under symmetric key K,
$\mathcal{D}_K(\cdot)$	corresponding decryption function under symmetric key K,
SK	session key,
T	a spanning tree with vertices $S[1,\ldots,n]$,
h	height of the tree T,
l_i	number of children of S_i in T,
PS_i	parent of sponsor S_i in T,
C_jS_i	jth child of S_i in T,
$\mathsf{CLK}[PS_i]$	cluster link key of S_i with its parent,
$\mathsf{CLK}[C_jS_i]$	cluster link key of S_i with its jth child,
$\mathsf{CLK}_i[0,\ldots,l_i]$	cluster link keys of S_i with its direct neighbours in T

More precisely, let $\mathsf{CLK}_i[0] = \mathsf{CLK}[PS_i], \mathsf{CLK}_i[j] = \mathsf{CLK}[C_jS_i]$ for $j = 1,\ldots,l_i$.

For Algorithm 3, we need to use a level for each node in the spanning tree T. The following notations are used to describe this algorithm for $j = 1,\ldots,h$:

$S_i^{(j)}$	ith node at the jth level in T,
$S_1^{(h)}$	root node in T at the highest level h,
n_j	number of nodes at the jth level in T,
$l_i^{(j)}$	number of children of $S_i^{(j)}$ in T, $i = 1,\ldots,n_j$,
$P^{(j+1)}S_i^{(j)}$	parent of sponsor $S_i^{(j)}$ in T at the $j+1$th level,
$C_k^{(j-1)}S_i^{(j)}$	kth child of $S_i^{(j)}$ in T at the $(j-1)$th level,
$\mathsf{CK}_i^{(j)}$	cluster key of the sponsor $S_i^{(j)}$,
$\mathsf{CLK}[P^{(j+1)}S_i^{(j)}]$	cluster link key of $S_i^{(j)}$ with its parent at the $(j+1)$th level in T,
$\mathsf{CLK}[C_k^{(j-1)}S_i^{(j)}]$	cluster link key of $S_i^{(j)}$ with its kth child at the $(j-1)$th level in T, and
$\mathsf{CLK}_i^{(j)}[0,\ldots,l_i^{(j)}]$	cluster link keys of $S_i^{(j)}$ with its direct neighbours in T

More precisely, let $\mathsf{CLK}_i^{(j)}[0] = \mathsf{CLK}[P^{(j+1)}S_i^{(j)}]$, $\mathsf{CLK}_i^{(j)}[k] = \mathsf{CLK}\left[C_k^{(j-1)}S_i^{(j)}\right]$ for $k = 1,\ldots,l_i^{(j)}$.

3.3. Unauthenticated hybrid protocol HP-1

In this protocol, we do not need to assume that all sponsors can communicate among themselves in single-hop. Rather we use a more flexible setting where the sponsors constitute a connected multi-hop hierarchical network. In other words, all sponsors in this setup need not to communicate among themselves, only neighbouring sponsors communicate and perform DH key exchange to form cluster link keys. This is a more reasonable assumption in the context of wireless ad hoc network because the sponsors' transmission ranges might be limited as well and it might be difficult for a sponsor to reach the other sponsors in a single-hop. We describe HP-1 in two phases: (*a*) *Initial Key Agreement (IKA)* phase that establishes the initial group session key, and (*b*) *Group Key Maintenance (GKM)* phase that handles all dynamic events such as a member join/leave and then refreshes the group session key.

3.3.1. Initial Key Agreement

Initially, suppose the entire set of nodes in the network are divided into n clusters $C_1,\ldots,C_n$ following our clustering algorithm as described in the Section 3.1. Let S_i be the sponsor node in C_i for $i = 1,\ldots,n$. Our protocol consists of the following steps.

Step 1: (*Cluster Key Agreement*) All clusters execute DB.Setup in parallel and compute their respective cluster keys. Let CK_i be the common cluster key agreed among all the nodes in C_i for $i = 1,\ldots,n$. We call this procedure ClusterKeyAgree which is described as Algorithm 1.

Algorithm 1. (**procedure** ClusterKeyAgree) Computation of cluster key CK_i among all the nodes in the cluster C_i for $i = 1,\ldots,n$

1: **for** $i \leftarrow 1$ **to** n **in parallel**
2: **call** DB.Setup among all nodes in the cluster C_i
3: Let CK_i be the common cluster key agreed among all nodes of C_i
4: **end for**

Step 2: (*Cluster Link Key Agreement*) Consider a graph with sponsors $S_1,S_2,\ldots,S_n$ as vertices of the graph and two vertices S_i,S_j are connected by an edge if they can communicate in one-hop. Find a spanning tree T in the graph by performing a Breadth-First Search (BFS) or a Depth-First Search (DFS) on the graph, time complexity of which is $O(e)$, where e is the number of edges in the graph [21,38] . This means that T has all the sponsors $S_1,S_2,\ldots,S_n$ as its vertices and each vertex in T can communicate with its parent and children in one-hop. Thus all the sponsors in the ad hoc network construct a virtual hierarchical link network. Let h be the height of the tree T. All sponsors in T invoke DH key

agreement simultaneously with their one-hop away neighbours (parent and children) in T to construct all their cluster link keys. While doing this, each sponsor uses the same private ephemeral key for all its cluster link keys in a session. The root sponsor in T has no parent, so it computes cluster link keys only with its children. As long as the sponsors do not leave the network, this spanning tree T does not need to be reconstructed and can be used for several sessions to update/refresh session key due to member join or member leave or any change in the network topology. We call this procedure ClusterLinkKeyAgree. See Algorithm 2.

Algorithm 2. (**procedure** ClusterLinkKeyAgree) Computation of cluster link keys CLK_i along the edges (or between the adjacent sponsors) of the spanning tree T of sponsors $S_1,\ldots,S_n$

1: **for** $i \leftarrow 1$ **to** n **in parallel**
2: **perform** Steps 3–8 **in parallel**
3: **call** DH between sponsor S_i and its child C_jS_i for each j, $j = 1,\ldots,l_i$
4: Let $\mathsf{CLK}[C_jS_i]$ be the cluster link keys agreed between S_i and its respective child CjS_i in T for each j, $j = 1,\ldots,l_i$
5: **end for**

Step 3: (*Session Key Distribution among sponsors using cluster link keys*) We call this procedure SessionKeyDist-Sponsors which is given by Algorithm 3.

(a) The root node (sponsor) in the tree T selects a random session key SK, drawn uniformly from the key space, encrypts SK using the cluster link keys of its children and sends the respective encrypted values to the corresponding children.
(b) The one-hop away children (sponsors) of the root sponsor decrypts the corresponding broadcast value with their respective cluster link keys and recovers the session key SK.
(c) Each child re-encrypts the recovered session key SK with the cluster link keys of its children and broadcasts the encrypted values.
(d) Continue stages (*b*) and (*c*) in each subtree of the root sponsor until the leaf nodes are reached.

Algorithm 3. (**procedure** SessionKeyDist-Sponsors) Session Key Distribution among the sponsors from the root level to the leaf level in the spanning tree T of sponsors

1: The root sponsor $S_1^{(h)}$ chooses a random session key SK and then computes and sends $\mathcal{E}_{\mathsf{CLK}[C_k^{(h-1)}S_1^{(h)}]}(\mathsf{SK})$ to its kth child $C_k^{(h-1)}S_1^{(h)}$ in T for $k = 1,\ldots,l_1^{(h)}$
2: **for** $j \leftarrow h-1$ **downto** 1 **do**
3: **for** $i \leftarrow 1$ **to** n_j **do in parallel**
4: $S_i^{(j)}$ decrypts $\mathcal{E}_{\mathsf{CLK}\left[P^{(j+1)}S_i^{(j)}\right]}(\mathsf{SK})$ using the cluster link key $\mathsf{CLK}\left[P^{(j+1)}S_i^{(j)}\right] \in \mathsf{CLK}_i^{(j)}\left[0,\ldots,l_i^{(j)}\right]$ and recovers SK
5: $S_i^{(j)}$ then computes and sends $\mathcal{E}_{\mathsf{CLK}[C_k^{(j-1)}S_i^{(j)}]}(\mathsf{SK})$ to its kth child $C_k^{(j-1)}S_i^{(j)}$ in T for $k = 0,\ldots,l_i^{(j)}$ using the cluster link keys $\mathsf{CLK}\left[C_k^{(j-1)}S_i^{(j)}\right] \in \mathsf{CLK}_i^{(j)}\left[0,\ldots,l_i^{(j)}\right]$
6: **end for**
7: **end for**

Step 4: (*Session Key Distribution among the nodes in a cluster using cluster key*) The sponsor S_i in the cluster C_i encrypts the session key SK using the cluster key CK_i and broadcasts this encrypted value within its cluster. Each cluster member in C_i decrypts the broadcast value using the common cluster key CK_i and recovers the session key SK. We call this procedure SessionKeyDist-ClusterNodes and describe as Algorithm 4.

Algorithm 4. (**procedure** SessionKeyDist-ClusterNodes) Session Key Distribution among all the cluster nodes within the cluster C_i

1: The sponsor S_i in the cluster C_i computes and broadcasts $\mathcal{E}_{\mathsf{CK}_i}(\mathsf{SK})$
2: Each cluster node in C_i decrypts the broadcast value using the common cluster key CK_i and recovers SK

3.3.2. Group Key Maintenance

Wireless nodes are highly mobile in nature and the nodes' movement may cause the change of the network topology frequently. It is therefore important and necessary to update the group session key to ensure security. No maintenance is necessary in case the node's movement do not cause the change of the cluster structure. Otherwise, the following steps are performed.

Step 1: Suppose a set of nodes U wants to join/leave the cluster C_i. Update the cluster key CK_i in C_i by invoking DB.Join among $C_i \cup U$ if U wants to join C_i or DB.Leave among $C_i \backslash U$ if U wants to leave C_i. This procedure is called ClusterKeyUpdate and can be found as Algorithm 5.

We update the cluster keys in the clusters where membership changes have occurred by invoking ClusterKeyUpdate.

Algorithm 5. (**procedure** ClusterKeyUpdate) Cluster key update in the cluster C_i when a set of users U joins/leaves C_i

1: **if** U wants to join C_i **then**
2: **call** DB.Join among the nodes $C_i \cup U$
3: **els if** U wants to leave C_i **then**

4: **call** DB.Leave among the nodes $C_i \backslash U$
5: **end if**
6: Let CK_i be the updated cluster key in the new cluster

Step 2: The cluster link keys of sponsor S_i are updated by invoking DH key agreement simultaneously between S_i and its immediate neighbours (parent and children) in the spanning tree T of sponsors. Thus if number of children of S_i in T is l_i, then S_i needs to perform $(l_i + 1)$ DH key agreement to update the cluster link keys and the parent and children of S_i has to perform only 1 DH key agreement each to update the corresponding cluster link keys. All these cluster link key updates are done among the sponsor S_i of the modified cluster C_i (where membership changes have occurred) and its one-hop away neighbours in the tree T in parallel by simultaneous execution of DH key agreement. Other sponsors in the tree T do not need to update their cluster link keys. (*cf.* Fig. 6).

Step 3: A new session key is chosen by the root sponsor in T. This new session key is distributed among the other sponsors by invoking SessionKeyDist-Sponsors (see Algorithm 3) using the cluster link keys.

Step 4: Finally, each sponsor distributes the new session key among all the nodes in its own cluster by invoking SessionKeyDist-ClusterNodes (see Algorithm 3) using the cluster key.

3.4. Unauthenticated hybrid protocol HP-2

In this protocol, we assume that all the sponsors are able to communicate among themselves in single-hop and thus is a more restrictive network for mobile nodes as compared to HP-1. As in HP-1, protocol HP-2 also consists of two phases: (*a*) *Initial Key Agreement (IKA)* phase, and (*b*) *Group Key Maintenance (GKM)* phase.

3.4.1. Initial Key Agreement

Initially, all the nodes are divided into n clusters $C_1,\ldots,C_n$ with sponsors $S_1,\ldots,S_n$ respectively following our clustering algorithm as described in the Section 3.1. Our protocol consists of the following Steps.

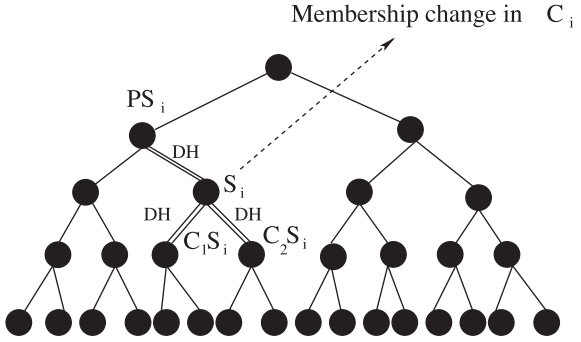

Fig. 6. Cluster link key updates in case T is a balanced binary tree of sponsors, where membership changes have occurred within the cluster C_i; sponsor S_i performs 3 DH and its one-hop away neighbours (PS_i, C_1S_i and C_2S_i) in T perform 1 DH each.

Step 1: (*Cluster Key Agreement*) We invoke procedure ClusterKeyAgree, where all clusters $C_1,\ldots,C_n$ execute DB.Setup in parallel and compute their respective cluster keys $CK_1\ldots,CK_n$ following Algorithm 1.

Step 2: (*Session key agreement among sponsors*) Consider the sponsors $S_1,S_2,\ldots,S_n$ on a virtual ring and execute DB.Setup among them to agree upon a common session key SK. While doing this, the sponsors select their respective private ephemeral keys at random. We call this procedure SessionKeyAgree-Sponsors and present as Algorithm 6.

Algorithm 6. (**procedure** SessionKeyAgree-Sponsors) Agreement of session key SK among the sponsors $S_1,\ldots,S_n$

1: **call** DB.Setup among the sponsors $S_1,S_2,\ldots,S_n$
2: Let SK be the common session key agreed among $S_1,S_2,\ldots,S_n$

Step 3: (*Session Key Distribution among the nodes in a cluster using cluster key*) We invoke procedure SessionKeyDist-ClusterNodes described by Algorithm 4 in each cluster C_i. The sponsor S_i in the cluster C_i encrypts the session key SK using the cluster key CK_i and broadcasts this encrypted value within its cluster. Each cluster member in C_i decrypts the broadcast value using the common cluster key CK_i and recovers the session key SK.

3.4.2. Group Key Maintenance

The following steps are performed to update group session key to handle dynamic membership change.

Step 1: Suppose a set of nodes U wants to join/leave the cluster C_i. Update the cluster keys in the clusters where membership changes have occurred by invoking ClusterKeyUpdate as given by Algorithm 5.

Step 2: A new session key is agreed by invoking DB.Setup among the sponsors $S_1,S_2,\ldots,S_n$. This is done in an efficient manner to save the computation costs for most of the sponsors retaining the security of the protocol. While executing DB.Setup, the sponsors corresponding to the clusters where membership changes have occurred select new private

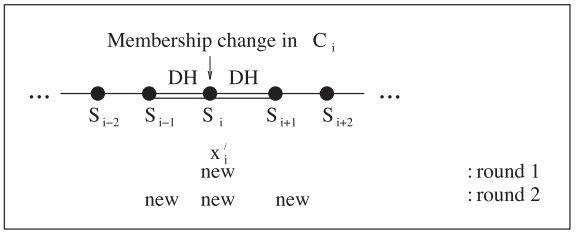

Fig. 7. DB.Setup among sponsors where membership changes have occurred within the cluster C_i; the sponsor S_i performs 2 DH and its immediate neighbours (S_{i-1} and S_{i+1}) in the ring performs 1 DH each.

ephemeral keys. Other sponsors make use of the precomputed exponentiation in the first round communication and precomputed left key- right key pairs in the second round communication. (*cf.* Fig. 7).

Step 3: Finally, each sponsor distributes the new session key among all the nodes in its own cluster by invoking SessionKeyDist-ClusterNodes (see Algorithm 4) using the cluster key.

3.5. Authenticated hybrid protocols AHP-1 and AHP-2

We now describe the idea of transforming our unauthenticated protocols HP-1 and HP-2 to secure authenticated protocols AHP-1 and AHP-2 respectively to mount attacks against an active adversary. We adapt the authentication mechanism used by Dutta and Barua [15] which is an efficient and simplified variant of Katz–Yung [23] generic technique for converting any group key agreement protocol secure against a passive adversary to a group key agreement protocol secure against an active adversary. We use a digital signature scheme $\mathsf{DSig} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ which is strongly unforgeable under adaptive chosen plaintext attack. Here $\mathcal{K}$ is key generation algorithm, $\mathcal{S}$ is signature generation algorithm and $\mathcal{V}$ is signature verification algorithm. Let $\mathsf{Adv}_{\mathsf{DSig}}(t)$ denote the maximum advantage of any adversary running in time t in forging a new message-signature pair. For simplicity, we assume that the signature length is independent of the length of the message signed, which can be achieve in practical by using collision-resistance hash function on the message. We describe briefly the construction of AHP-1 from HP-1. A similar construction can be given for AHP-2 from HP-2. From henceforth, $A|B$ stands for concatenation of elements A and B.

1. As part of the signature scheme, each user U_i chooses a signing and a verification key sk_{U_i} and pk_{U_i} respectively by running the key generation algorithm.
2. The users in the network now execute the protocol $\mathsf{HP}-1$ with the following modifications:
   (a) Quite often, we identify a user U_i with its instance $\Pi_{U_i}^{d_i}$ for some integer d_i that is session specific during a protocol execution. Let m be the tth message broadcast by an instance $\Pi_{U_i}^{d_i}$ with identity U_i as part of the protocol $\mathsf{HP}-1$. Then for the protocol $\mathsf{AHP}-1$, instance $\Pi_{U_i}^{d_i}$ replaces m by $M_i = U_i|t|m|d_i$, computes the signature $\sigma_i = \mathcal{S}(\mathsf{sk}_{U_i}, M_i)$ and broadcasts $M_i{-}\sigma_i$.
   (b) Let instance $\Pi_{U_i}^{d_i}$ receives a message of the form $U_j|t|m|d_j|\sigma_j$ as part of the protocol AHP-1. Then $\Pi_{U_i}^{d_i}$ checks that:
   - $U_j \in \mathsf{pid}_{U_i}^{d_i}$, where $\mathsf{pid}_{U_i}^{d_i}$ is the partner identity for instance $\Pi_{U_i}^{d_i}$ as defined in Section 2.5;
   - t is the next expected message number for messages from instance $\Pi_{U_j}^{d_j}$; and
   - the validity of the signature σ_j using the verification algorithm $\mathcal{V}$ and respective verification key pk_{U_j}. If verification fails, then $\Pi_{U_i}^{d_i}$ sets $\mathsf{acc}_{U_i}^{d_i} = 0$, $\mathsf{sk}_{U_i}^{d_i} = \mathsf{NULL}$ and aborts. Otherwise, $\Pi_{U_i}^{d_i}$ continues as it would in $\mathsf{HP}-1$ upon receiving tth message m from instance $\Pi_{U_j}^{d_j}$.
3. Each non-aborted instance computes the session key as in HP-1 and builds up the session identity as the protocol proceeds by extracting the identities and instance numbers of the participants from the publicly transmitted messages. The detail construction of session identity will be described shortly.

We point out the following issues related to the above authentication mechanism.

- *Partner identity:* We assume that any instance $\Pi_{U_j}^{d_j}$ knows its partner identity $\mathsf{pid}_{U_j}^{d_j}$, which is essentially the set of users with which it is partnered in the particular session.
- *Numbering of the messages:* Any instance $\Pi_{U_j}^{d_j}$ sends out a finite (two) number of messages, which can be uniquely numbered by $\Pi_{U_j}^{d_j}$ based on their order of occurrence.
- *Session identity:* This is defined in a different way than Katz–Yung and plays an important role in authenticating our protocols. Session identity is required to identify a session uniquely and all participants executing a session should hold the same session identity. Conventionally, session identity sid_U^i for an instance Π_U^i is set to be concatenation of all (broadcast) messages sent and received by Π_U^i during its course of execution. This essentially assumes that all the partners of Π_U^i hold the same concatenation value of sent and received messages which may not be the case in general. Our definition of session identity is different which is similar to that documented in [15] and can be applied for more general protocols.

Suppose users $U_{i_1}, \ldots, U_{i_k}$ wish to agree upon a common key in a session using unused instances $\Pi_{U_{i_1}}^{d_1}, \ldots, \Pi_{U_{i_k}}^{d_k}$. According to our definition in Section 2.5, $\mathsf{sid}_{U_{i_j}}^{d_j} = \{(U_{i_1}, d_1), \ldots, (U_{i_k}, d_k)\}$. At the start of the session, $\Pi_{U_{i_j}}^{d_j}$ need not to know the entire set $\mathsf{sid}_{U_{i_j}}^{d_j}$. This set is built up as the protocol proceeds. Of course, we assume that $\Pi_{U_{i_j}}^{d_j}$ knows the pair (U_{i_j}, d_j). Clearly, $\Pi_{U_{i_j}}^{d_j}$ knows U_{i_j}. Knowledge of d_j can be maintained by U_{i_j} by keeping a counter which is incremented when a new instance is created. Each instance keeps the partial information about the session identity in a variable psid_U^i. Before the start of a session an instance $\Pi_{U_{i_j}}^{d_j}$ sets $\mathsf{psid}_{U_{i_j}}^{d_j} = \{(U_{i_j}, d_j)\}$. As the protocol proceeds, $\Pi_{U_{i_j}}^{d_j}$ keeps on extracting identity-instance number pairs of other instances engaged in that session. Notice that identity-instance number pairs are "patched" with appropriate signatures in the publicly transmitted messages of AHP-1. After completion of the session, $\mathsf{psid}_{U_{i_j}}^{d_j} = \mathsf{sid}_{U_{i_j}}^{d_j} = \{(U_{i_1}, d_1), \ldots, (U_{i_k}, d_k)\}$. We refer to [12,15] for join and leave algorithms where the detail construction of session identity sid from partial session identity psid is provided.

We will make the assumption that in each session at most one instance of each user participates. Further, an instance of a particular user participates in exactly one session. This is not a very restrictive assumption, since a user can spawn an instance for each session it participates in. On the other hand, there is an important consequence of this assumption. Suppose there are several sessions which

are being concurrently executed. Let the session identities be $\mathsf{sid}_1, \ldots, \mathsf{sid}_k$. Then for any instance Π_U^i, there is at most one j such that $(U,i) \in \mathsf{sid}_j$ and for any $j_1 \neq j_2$, we have $\mathsf{sid}_{j_1} \cap \mathsf{sid}_{j_2} = \emptyset$. Thus at any particular point of time, if we consider the collection of all instances of all users, then the relation of being in the same session is an equivalence relation whose equivalence classes are the session identities. Moreover, an instance Π_U^i not only knows U, but also the instance number i – this being achieved by maintaining a counter.

We bind the session identity with the message transmitted during our protocol execution. Since session identities uniquely identifies a session and all users in a particular session hold the same session identity, such an inclusion of session identity in transmitted messages prevents replay attack. In replay attack, adversary uses messages transmitted in a previous session in current session to obtain some information. The use of previously transmitted message in the current session is not valid as session identities are different.

The above variant of Katz–Yung compiler [23] reduces communication rounds by one by avoiding random nonces. Unlike Katz–Yung, we do not need any extra round for communication of random nonces during the initialization phase. A setting of unique instance number is used instead, which additionally avoids the event Repeat that comes in Katz–Yung security proof, thus reducing complexity of Katz–Yung compiler with a tighter security proof.

Remark. The clusters need to be stable enough so that the nodes do not leave or join while the protocol is still being executed. We may allow periodic group re-keying, *i.e.* periodic join/leave of the nodes.

4. Security analysis

We consider the security of the Initial Key Agreement (*IKA*) phase and the group key management (*GKM*) phase of our proposed protocols separately. Let HP-1.IKA and HP-1.GKM respectively denote the *IKA* and *GKM* phase of the protocol HP-1. Similarly, we define HP-2.IKA and HP-2.GKM for HP-2, AHP-1.IKA and AHP-1.GKM for AHP-1, and AHP-2.IKA and AHP-2.GKM for AHP-2. We first state the security results of HP-1.IKA and HP-2.IKA in Theorems 4.3 and 4.4 respectively and present the proof of Theorem 4.3 which makes use of Lemmas 4.1 and 4.2. We will show in the proof of Theorem 4.3 that HP-1.IKA is secure against passive adversary assuming that DDH problem is hard and the symmetric encryption scheme Symm is secure. A similar proof holds for Theorem 4.4. Lemma 4.1 is the security result of the DB protocol against passive adversary and we refer to [15] for its proof. Lemma 4.2 is the security result of DH protocol against passive adversary and its proof is provided in this section.

Lemma 4.1. [15] *The unauthenticated protocol* DB *described in Appendix 2.3 is secure against passive adversary under DDH assumption, achieves forward secrecy and satisfies the following:* $\mathsf{Adv}_{\mathsf{DB}}^{\mathsf{KA}}(t,q_E) \leqslant 4\,\mathsf{Adv}_G^{\mathsf{DDH}}(t') + \frac{4q_E}{|G|}$ *where* $t' = t + O(|\mathcal{P}|\ q_E\ t_{exp})$, t_{exp} *is the time required to perform exponentiation in* G, $|\mathcal{P}|$ *is number of participants in the network (which is a fixed number) and* q_E *is the number of* Execute *queries that an adversary may ask.*

Proof. The proof of this Lemma is given in [15] as a Theorem. □

Lemma 4.2. *The unauthenticated protocol* DH *described in Appendix 2.2 is secure against passive adversary under DDH assumption, achieves forward secrecy and satisfies the following:* $\mathsf{Adv}_{\mathsf{DH}}^{\mathsf{KA}}(t,q_E) \leqslant 2\mathsf{Adv}_G^{\mathsf{DDH}}(t') + \frac{2q_E}{|G|}$ *where* $t' = t + O(2q_E t_{exp})$, t_{exp} *is the time required to perform exponentiation in* G *and* q_E *is the number of* Execute *queries that an adversary may ask.*

Proof. Let $\mathcal{A}$ be a passive adversary for the unauthenticated Diffie–Hellman protocol DH. Given $\mathcal{A}$, we construct an algorithm $\mathcal{D}$ which solves DDH problem with non-negligible advantage. The adversary $\mathcal{A}$ has access to Execute, Reveal and Test queries. The protocol trivially achieves forward secrecy as there is no long-term secret key in the unauthenticated DH protocol and thus Corrupt query may simply be ignored. We first consider $\mathcal{A}$ makes a single Execute query. Let us define two distributions Real and Fake for transcript/session key pairs (T,sk) as follows:

$$\mathsf{Real} := \left\{ \begin{array}{l} x_1, x_2 \leftarrow Z_q^*; \\ y_1 = g^{x_1}, y_2 = g^{x_2}; \\ T = (y_1, y_2); \mathsf{sk} = g^{x_1 x_2} \end{array} : (T,\mathsf{sk}) \right\}$$

$$\mathsf{Fake} := \left\{ \begin{array}{l} x_1, x_2 \leftarrow Z_q^*; \\ y_1 = g^{x_1}, y_2 = g^{x_2}; \\ T = (y_1, y_2); \mathsf{sk} \leftarrow G \end{array} : (T,\mathsf{sk}) \right\}$$

We first have the following claim that deals with the Execute and Reveal queries. □

Claim 1. *For any adversary* $\mathcal{A}$ *running in time* t, *we have the following where* $t'' \leqslant t + 2\ t_{exp}$:

$$|\mathsf{Prob}[(T,\mathsf{sk}) \leftarrow \mathsf{Real} : \mathcal{A}(T,\mathsf{sk}) = 1] - \mathsf{Prob}[(T,\mathsf{sk}) \leftarrow \mathsf{Fake} : \mathcal{A}(T,\mathsf{sk}) = 1]| \leqslant \mathsf{Adv}_G^{\mathsf{DDH}}(t'') + \frac{1}{|G|}.$$

Proof. Let $(A,B,C) \in G^3$ be any instance for DDH problem. Using $\mathcal{A}$, we construct a distinguisher $\mathcal{D}$ for DDH problem that takes (A,B,C) as input and outputs whatever $\mathcal{A}$ outputs. Let Dist be defined as follows using (A,B,C):

$$\mathsf{Dist} := \left\{ \begin{array}{l} x_1, x_2 \leftarrow Z_q^*; \\ y_1 = A^{x_1}, y_2 = B^{x_2}; \\ T = (y_1, y_2); \mathsf{sk} = C^{x_1 x_2} \end{array} : (T,\mathsf{sk}) \right\}$$

We now analyze the output of $\mathcal{D}$. The distribution Real and the distribution

$$\{a, b \leftarrow Z_q^*, A = g^a, B = g^b, C = g^{ab}; (T,\mathsf{sk}) \leftarrow \mathsf{Dist} : (T,\mathsf{sk})\}$$

are statistically equivalent as long as the exponents x_1, x_2 used in Dist are random. On the other hand, the distribution Fake and the distribution

$\{a, b \leftarrow Z_q^*, c \leftarrow Z_q^* \setminus \{ab\}, A = g^a, B = g^b,$
$C = g^c; (T, \mathsf{sk}) \leftarrow \mathsf{Dist} : (T, \mathsf{sk})\}$

are statistically close to within a factor of $\frac{1}{|G|}$. The only difference is in Fake the value of sk is chosen uniformly at random from G, whereas in Dist this value is chosen uniformly from $G\backslash\{g^{ab}\}$, otherwise, these two distributions are statistically equivalent by the random self-reducibility property of DDH problem. Consequently, we have the following:

$$\begin{aligned}|\mathsf{Prob}[(T,\mathsf{sk}) \leftarrow \mathsf{Real} : \mathcal{A}(T,\mathsf{sk}) = 1] - \mathsf{Prob}[(T,\mathsf{sk}) \\ \leftarrow \mathsf{Fake} : \mathcal{A}(T,\mathsf{sk}) = 1]| \leqslant |\mathsf{Prob}[a,b \\ \leftarrow Z_q^* : \mathcal{D}(g^a, g^b, g^{ab}) = 1] - \mathsf{Prob}[a,b \\ \leftarrow Z_q^*, c \leftarrow Z_q^* \setminus \{ab\} : \mathcal{D}(g^a, g^b, g^c) = 1]| + \frac{1}{|G|} \\ \leqslant \mathsf{Adv}_G^{\mathsf{DDH}}(t'') + \frac{1}{|G|}\end{aligned}$$

as the time of $\mathcal{D}$ is dominated by the time t'' of $\mathcal{A}$. Here t'' is basically equal to $t + 2t_{exp}$, where t_{exp} is the time required to perform an exponentiation in G. □

Next we have the following claim to deal with the Test query, the proof of which follows from the fact that in Fake, the session key sk is completely independent of the transcript T generated there.

Claim 2. *For any computationally unbounded adversary $\mathcal{A}$, we have*

$$\mathsf{Prob}[(T,\mathsf{sk}_0) \leftarrow \mathsf{Fake}; \mathsf{sk}_1 \leftarrow G; b \leftarrow \{0,1\} : \mathcal{A}(T,\mathsf{sk}_b) = b] = \frac{1}{2}.$$

Then we have

$$\begin{aligned}\mathsf{Adv}_{\mathsf{DH},\mathcal{A}}^{\mathsf{KA}}(t,1) := |2\mathsf{Prob}[\mathsf{Succ}] - 1| = 2|\mathsf{Prob}[(T,\mathsf{sk}_0) \\ \leftarrow \mathsf{Real}, \mathsf{sk}_1 \leftarrow G, b \leftarrow \{0,1\} : \mathcal{A}(T,\mathsf{sk}_b) \\ = b] - \frac{1}{2}\Big| = 2|\mathsf{Prob}[(T,\mathsf{sk}_0) \leftarrow \mathsf{Real}, \mathsf{sk}_1 \leftarrow G, b \\ \leftarrow \{0,1\} : \mathcal{A}(T,\mathsf{sk}_b) = b] - \mathsf{Prob}[(T,\mathsf{sk}_0) \\ \leftarrow \mathsf{Fake}, \mathsf{sk}_1 \leftarrow G, b \leftarrow \{0,1\} : \mathcal{A}(T,\mathsf{sk}_b) = b]|\end{aligned}$$

by Claim 2 *and*

$$\mathsf{Adv}_{\mathsf{DH}}^{\mathsf{KA}}(t,1) \leqslant 2\mathsf{Adv}_G^{\mathsf{DDH}}(t'') + \frac{2}{|G|}$$

by Claim 1.

Now consider the case for q_E(>1) Execute query. The distinguisher $\mathcal{D}$ first generates q_E tuples (A_i, B_i, C_i), $1 \leqslant i \leqslant q_E$ with the following properties from the tuple $(A,B,C) \in G^3$ given to it.

1. If $(A,B,C) \leftarrow \Delta_{\mathsf{Real}}$, then $(A_i,B_i,C_i) \leftarrow \Delta_{\mathsf{Real}}$ for all i, $1 \leqslant i \leqslant q_E$ with (A_i,B_i) randomly distributed in G^2 (independently of anything else).
2. If $(A,B,C) \leftarrow \Delta_{\mathsf{Rand}}$, then $(A_i,B_i,C_i) \leftarrow \Delta_{\mathsf{Rand}}$ for all i, $1 \leqslant i \leqslant q_E$ with (A_i,B_i,C_i) randomly distributed in G^3 (independently of anything else).

Then proceeding in the similar way as above of defining distributions Real, Fake, Dist we may define distributions Real_{q_E}, Fake_{q_E} and Dist_{q_E} which simply consist of q_E independent copies of each of the corresponding distributions. In case of Dist_{q_E}, we use the corresponding tuple (A_i,B_i,C_i) for the ith copy. We use notation $(\vec{T}, \vec{\mathsf{sk}})$ to denote the transcript/session key pair generated by these distributions. Then similar to the claims 1 and 2 , we can prove the following claims:

Claim 3. *For any algorithm $\mathcal{A}$ running in time t, we have*

$$\begin{aligned}|\mathsf{Prob}[(\vec{T},\vec{\mathsf{sk}}) \leftarrow \mathsf{Real}_{q_E} : \mathcal{A}(\vec{T},\vec{\mathsf{sk}}) = 1] - \mathsf{Prob}[(\vec{T},\vec{\mathsf{sk}}) \leftarrow \mathsf{Fake}_{q_E} \\ : \mathcal{A}(\vec{T},\vec{\mathsf{sk}}) = 1]| \leqslant \mathsf{Adv}_G^{\mathsf{DDH}}(t') + \frac{q_E}{|G|}\end{aligned}$$

where t' is as in the statement of the Theorem.

Claim 4. *For any computationally unbounded adversary $\mathcal{A}$, we have*

$$\begin{aligned}\mathsf{Prob}[(\vec{T},\vec{\mathsf{sk}}_0) \leftarrow \mathsf{Fake}; \vec{\mathsf{sk}}_1 \leftarrow G^{q_E}; b \leftarrow \{0,1\} : \mathcal{A}(\vec{T},\vec{\mathsf{sk}}_b) = b] \\ = \frac{1}{2}.\end{aligned}$$

Claims 3 and 4 *yield the result stated in the Lemma.*

Theorem 4.3. *The unauthenticated static hybrid key agreement protocol* HP − 1.IKA *described in Section 3.3 is secure against passive adversary under DDH assumption, achieves forward secrecy and satisfies the following:* $\mathsf{Adv}_{\mathsf{HP-1.IKA}}^{\mathsf{KA}}(t, q_E) \leqslant \frac{(3n-2)}{n(n-1)q_E}\mathsf{Adv}_G^{\mathsf{DDH}}(t') + \frac{3}{|G|} + \mathsf{Adv}_{\mathsf{Symm}}^{\mathsf{ind-cpa-fg}}(t,0,0) + \frac{(2n-1)}{2n(n-1)q_E}$ *where $t' = t + O(|\mathcal{P}_{\max}|\ n\ q_E\ t_{exp}), t_{exp}$ is the time required to perform exponentiation in G, $|\mathcal{P}_{\max}|$ is maximum number of nodes in a cluster (which is a fixed number), n is the number of clusters in the network,* Symm *is the symmetric encryption scheme used in the protocol* HP-1.IKA *and q_E is the maximum number of* Execute *queries that an adversary may ask.*

Proof. The proof considers an adversary $\mathcal{A}$ who defeats the security of our unauthenticated hybrid key distribution protocol HP-1.IKA. Given $\mathcal{A}$, we construct an adversary $\mathcal{B}$ attacking the symmetric encryption scheme Symm; relating the success probability of $\mathcal{A}$ and $\mathcal{B}$ gives the stated result of the Theorem. Before describing $\mathcal{B}$, we first define event Bad and bound its probability. Let Bad be the event that $\mathcal{A}$ is able to distinguish a cluster link key (which is a key agreed by DH protocol) or a cluster key (which is a key agreed by the DB protocol) from a random value at any point during its execution. Let Prob[Bad] stands for $\mathsf{Prob}_{\mathcal{A},\mathsf{HP-1.IKA}}[\mathsf{Bad}]$. Let Succ_1 and Succ_2 respectively denote the events that $\mathcal{A}$ wins the game as described in the security model in Section 2.4 for key agreement protocols DH and DB respectively.

Notice that with n clusters in the network, the spanning tree with n sponsor nodes have $(n-1)$ edges and hence $(n-1)$ execution of DH protocol is performed in each execution of our protocol HP-1.IKA to form the cluster link

keys, whereas each execution of HP-1.IKA invokes the DB protocol n times to form the cluster keys. The adversary $\mathcal{A}$ makes q_E Execute queries and consequently performs nq_E execution of the DB protocol and $(n-1)q_E$ execution of DH protocol. Consequently, $\text{Prob}[\text{Bad}] \leqslant \frac{\text{Prob}[\text{Succ}_1]}{(n-1)q_E} + \frac{\text{Prob}[\text{Succ}_2]}{nq_E}$.

Now by definition in Section 2.4, $\text{Adv}^{\text{KA}}_{\text{DH},\mathcal{A}} = |2\ \text{Prob}[\text{Succ}_1] - 1|$ and $\text{Adv}^{\text{KA}}_{\text{DB},\mathcal{A}} = |2\ \text{Prob}[\text{Succ}_2] - 1|$, which imply $\text{Prob}[\text{Succ}_1] \leqslant \frac{\text{Adv}^{\text{KA}}_{\text{DH},\mathcal{A}}+1}{2}$ and $\text{Prob}[\text{Succ}_1] \leqslant \frac{\text{Adv}^{\text{KA}}_{\text{DB},\mathcal{A}}+1}{2}$. Hence we have

$$\text{Prob}[\text{Bad}] \leqslant \frac{\text{Adv}^{\text{KA}}_{\text{DH},\mathcal{A}}}{2(n-1)q_E} + \frac{\text{Adv}^{\text{KA}}_{\text{DB},\mathcal{A}}}{2nq_E} + \frac{(2n-1)}{2n(n-1)q_E}.$$

$\mathcal{B}$ simulates all oracle queries of $\mathcal{A}$ by executing the protocol HP − 1.IKA on its own. Consequently, $\mathcal{B}$ can detect the occurrence of the event Bad. $\mathcal{B}$ provides a perfect simulation for $\mathcal{A}$ as long as the event Bad does not occur. If ever the event Bad occurs, $\mathcal{B}$ aborts and outputs a random bit. Otherwise, $\mathcal{B}$ outputs whatever bit is eventually output by $\mathcal{A}$. So $\text{Prob}_{\mathcal{A},\text{HP}-1.\text{IKA}}[\text{Succ}|\text{Bad}] = \frac{1}{2}$. Now

$$\begin{aligned}
\text{Adv}_{\mathcal{B},\text{Symm}} &:= 2\ |\text{Prob}_{\mathcal{B},\text{Symm}}[\text{Succ}] - 1/2| \\
&= 2\ |\text{Prob}_{\mathcal{A},\text{HP}-1.\text{IKA}}[\text{Succ} \wedge \overline{\text{Bad}}] \\
&\quad + \text{Prob}_{\mathcal{A},\text{HP}-1.\text{IKA}}[\text{Succ} \wedge \text{Bad}] - 1/2| \\
&= 2\ |\text{Prob}_{\mathcal{A},\text{HP}-1.\text{IKA}}[\text{Succ} \wedge \overline{\text{Bad}}] \\
&\quad + \text{Prob}_{\mathcal{A},\text{HP}-1.\text{IKA}}[\text{Succ}|\text{Bad}]\text{Prob}_{\mathcal{A},\text{HP}-1.\text{IKA}}[\text{Bad}] - 1/2| \\
&= 2\ |\text{Prob}_{\mathcal{A},\text{HP}-1.\text{IKA}}[\text{Succ} \wedge \overline{\text{Bad}}] \\
&\quad + (1/2)\text{Prob}_{\mathcal{A},\text{HP}-1.\text{IKA}}[\text{Bad}] - 1/2| \\
&= 2\ |\text{Prob}_{\mathcal{A},\text{HP}-1.\text{IKA}}[\text{Succ}] - \text{Prob}_{\mathcal{A},\text{HP}-1.\text{IKA}}[\text{Succ} \wedge \text{Bad}] \\
&\quad + (1/2)\text{Prob}_{\mathcal{A},\text{HP}-1.\text{IKA}}[\text{Bad}] - 1/2| \\
&\geqslant |2\ \text{Prob}_{\mathcal{A},\text{HP}-1.\text{IKA}}[\text{Succ}] - 1| - |\text{Prob}_{\mathcal{A},\text{HP}-1.\text{IKA}}[\text{Bad}] \\
&\quad - 2\ \text{Prob}_{\mathcal{A},\text{HP}-1.\text{IKA}}[\text{Succ} \wedge \text{Bad}]| \\
&\geqslant \text{Adv}_{\mathcal{A},\text{HP}-1.\text{IKA}} - \text{Prob}[\text{Bad}]
\end{aligned}$$

Note that $\mathcal{B}$ never invokes its encryption oracle $\mathcal{E}$. Moreover, the running time of $\mathcal{B}$ is at most t. Also since $\text{Adv}_{\mathcal{B},\text{Symm}} \leqslant \text{Adv}_{\text{Symm}}(t,0,0)$ by assumption,

$$\begin{aligned}
\text{Adv}_{\text{HP}-1.\text{IKA}}(t,q_E) &\leqslant \text{Adv}_{\text{Symm}}(t,0,0) + \text{Prob}[\text{Bad}] \\
&\leqslant \text{Adv}_{\text{Symm}}(t,0,0) + \frac{\text{Adv}^{\text{KA}}_{\text{DH},\mathcal{A}}(t,(n-1)q_E)}{2(n-1)q_E} \\
&\quad + \frac{\text{Adv}^{\text{KA}}_{\text{DB},\mathcal{A}}(t,nq_E)}{2nq_E} + \frac{(2n-1)}{2n(n-1)q_E} \\
&\leqslant \text{Adv}_{\text{Symm}}(t,0,0) + \frac{\text{Adv}^{\text{DDH}}_{G}(t'')}{(n-1)q_E} + \frac{1}{|G|} \\
&\quad + \frac{2\text{Adv}^{\text{DDH}}_{G}(t')}{nq_E} + \frac{2}{|G|} + \frac{(2n-1)}{2n(n-1)q_E}
\end{aligned}$$

by Lemmas 4.1 and 4.2 where t' is as in the Theorem and $t'' = t + O(2\ (n-1)q_E t_{exp})$. Hence we obtain the statement of the theorem. □

Theorem 4.4. *The unauthenticated static hybrid key agreement protocol* HP − 2.IKA *described in Section 3.4 is secure against passive adversary under DDH assumption, achieves forward secrecy and satisfies the following:* $\text{Adv}^{\text{KA}}_{\text{HP}-2.\text{IKA}}(t, q_E) \leqslant \frac{2}{(n+1)q_E}\text{Adv}^{\text{DDH}}_{G}(t') + \frac{2}{|G|} + \text{Adv}^{\text{ind}-\text{cpa}-\text{fg}}_{\text{Symm}}(t,0,0) + \frac{1}{2(n+1)q_E}$ *where* $t' = t + O(|\mathcal{P}_{\max}|\ n\ q_E t_{exp}), t_{exp}$ *is the time required to perform exponentiation in* G, $|\mathcal{P}_{\max}|$ *is maximum number of nodes in a cluster (which is a fixed number),* n *is the number of clusters in the network,* Symm *is the symmetric encryption scheme used in the protocol* HP-2.IKA *and* q_E *is the maximum number of* Execute *queries that an adversary may ask.*

Next we consider the security of the static authenticated protocols AHP-1.IKA and AHP-2.IKA and security of dynamic authenticated protocols AHP-1.GKM and AHP-2.GKM and state the respective results in Theorems 4.5, Theorems 4.6, 4.7 and 4.8. The security of all these authenticated protocols rely on that of unauthenticated HP-1.IKA and HP-2.IKA protocols assuming that the signature scheme DSig is secure. Since we use the authentication mechanism of [15], the proofs of these theorems are exactly similar to the reduction proof technique used by [15]

Theorem 4.5. *The authenticated protocol* AHP-*1*.IKA *described in section 3.5 is secure against active adversary under DDH assumption, achieves forward secrecy and satisfies the following:* $\text{Adv}^{\text{AKA}}_{\text{AHP}-1.\text{IKA}}(t, q_E, q_S) \leqslant \text{Adv}^{\text{KA}}_{\text{HP}-1.\text{IKA}}(t', q_E + \frac{q_S}{2}) + |\mathcal{P}|\text{Adv}_{\text{DSig}}(t')$ *where* $t' \leqslant t + (|\mathcal{P}|q_E + q_S)t_{\text{AHP}-1.\text{IKA}}$, *with* $t_{\text{AHP}-1.\text{IKA}}$ *is the time required for execution of* AHP − *1*.IKA *by any party,* q_E *and* q_S *are respectively the maximum number of* Execute *and* Send *query an adversary may ask.*

Theorem 4.6. *The authenticated protocol* AHP − *2*.IKA *described in Section 3.5 is secure against active adversary under DDH assumption, achieves forward secrecy and satisfies the following:* $\text{Adv}^{\text{AKA}}_{\text{AHP}-2.\text{IKA}}(t, q_E, q_S) \leqslant \text{Adv}^{\text{KA}}_{\text{HP}-2.\text{IKA}}(t', q_E + \frac{q_S}{2}) + |\mathcal{P}|\text{Adv}_{\text{DSig}}(t')$ *where* $t' \leqslant t + (|\mathcal{P}|q_E + q_S)t_{\text{AHP}-2.\text{IKA}}$, *with* $t_{\text{AHP}-2.\text{IKA}}$ *is the time required for execution of* AHP − *2*.IKA *by any party,* q_E *and* q_S *are respectively the maximum number of* Execute *and* Send *query an adversary may ask.*

Theorem 4.7. *The dynamic authenticated key agreement protocol* AHP − *1*.GKM *described in Section 3.5 satisfies the following:* $\text{Adv}^{\text{AKA}}_{\text{AHP}-1.\text{GKM}}(t, q_E, q_J, q_L, q_S) \leqslant \text{Adv}^{\text{KA}}_{\text{HP}-1.\text{IKA}}(t', q_E + (q_J + q_L + q_S)/2) + |\mathcal{P}|\ \text{Adv}_{\text{DSig}}(t')$ *where* $t' \leqslant t + (|\mathcal{P}|q_E + q_J + q_L + q_S)t_{\text{AHP}-1.\text{GKM}}$, *with* $t_{\text{AHP}-1.\text{GKM}}$ *is the time required for execution of* AHP-*1*.GKM *by any party,* q_E, q_S, q_J *and* q_L *are respectively the maximum number of* Execute, Send, Join *and* Leave *query an adversary may ask.*

Theorem 4.8. *The dynamic authenticated key agreement protocol* AHP-*2*.GKM *described in Section 3.5 satisfies the following:* $\text{Adv}^{\text{AKA}}_{\text{AHP}-2.\text{GKM}}(t, q_E, q_J, q_L, q_S) \leqslant \text{Adv}^{\text{KA}}_{\text{HP}-2.\text{IKA}}(t', q_E + (q_J + q_L + q_S)/2) + |\mathcal{P}|\ \text{Adv}_{\text{DSig}}(t')$ *where* $t' \leqslant t + (|\mathcal{P}|q_E + q_J + q_L + q_S)t_{\text{AHP}-2.\text{GKM}}$, *with* $t_{\text{AHP}-2.\text{GKM}}$ *is the time required for execution of* AHP − *2*.GKM *by any party,* q_E, q_S, q_J *and* q_L *are respectively the maximum number of* Execute, Send, Join *and* Leave *query an adversary may ask.*

A brief sketch of how the proofs are achieved is the following: We transform an active adversary $\mathcal{A}'$ attacking authenticated protocol P' into a passive adversary $\mathcal{A}$ attacking the underlying unauthenticated protocol P. Adversary $\mathcal{A}$ generates the verification/signing keys pk_U, sk_U for each user $U \in \mathcal{P}$ and gives the verification keys to $\mathcal{A}'$. We define event Forge to be the event that a signature of DSig is forged by $\mathcal{A}'$ and bound its probability following [23] by $\mathsf{Prob}[\mathsf{Forge}] \leqslant |\mathcal{P}|\mathsf{Adv}_{\mathsf{DSig}}(t')$. If ever the event Forge occurs, adversary $\mathcal{A}$ aborts and outputs a random bit. Otherwise, $\mathcal{A}$ outputs whatever bit is eventually output by $\mathcal{A}'$. Note that since the signing and verification keys are generated by $\mathcal{A}$, it can detect occurrence of the event Forge.

$\mathcal{A}$ simulates the oracle queries of $\mathcal{A}'$ using its own queries to the Execute *oracle*. The idea is that the adversary $\mathcal{A}$ queried its Execute oracle to obtain a transcript T of P for each Execute query of $\mathcal{A}'$ and also for each initial send query $\mathsf{Send}_0(U, i, *)$ of $\mathcal{A}'$. For dynamic case also, $\mathcal{A}$ itself simulates all the oracle queries including Join and Leave oracles of $\mathcal{A}'$ using its own Execute and Reveal oracles of the static variant of P and then modifying the resulting transcript according to the dynamic variant of P as in [15]. $\mathcal{A}$ then patches appropriate signatures with the messages in T to obtain a transcript T' of P' and uses T' to answer queries of $\mathcal{A}'$. Since $\mathcal{A}'$ cannot forge signatures with respect to any of the users, $\mathcal{A}'$ is 'limited' to send messages already contained in T'. This technique provides a good simulation. Finally, we relate the advantages of $\mathcal{A}'$ and $\mathcal{A}$ as $\mathsf{Adv}^{\mathsf{AKA}}_{\mathcal{A}',P'} \leqslant \mathsf{Adv}^{\mathsf{KA}}_{\mathcal{A},P}(t', q_E + q_S/2) + \mathsf{Prob}[\mathsf{Forge}]$ which gives the results stated in the above theorems. Due to space constraints we omit the proofs here and refer to [15] for more specific details of oracle simulations.

The way we bind session identity with each message in our protocol run, enables to handle replay attacks without using random nonces as in Katz–Yung [23]. In our unauthenticated protocols, there are no long term secret keys. Thus we can avoid Corrupt oracle queries and the unauthenticated protocols trivially achieve forward secrecy. Then following Katz–Yung [23], we can avoid Corrupt query for our authenticated protocols (both static and dynamic) also, because this query for the authenticated protocols outputs long-term secret key (if any), defined as part of the unauthenticated protocol. Thus we can trivially achieve forward secrecy for the authenticated protocols.

5. Efficiency

Concerning the efficiency, notice that both our proposed hybrid protocols are designed without using pairings and make use of multi-party dynamic key agreement protocol DB. Additionally, our first hybrid protocol uses 2-party DH key exchange. In this section, we analyse the performance in key management problem for our hybrid solutions. Sucec and Marsic [35] provides a theoretical upper bound $O(n\log n)$ (n is the node count) on the communication overhead incurred by a particular clustering algorithm in ad hoc networks. The optimization of a cluster-based ad hoc network has been a topic of many papers. We consider it as a separate issue and will not address it

Table 1
Complexity of 2-party DH and n-party DB.Setup

Protocol	Communication		Computation					
	R	B	Exp	Mul	Div	Sig	Ver	Hash
DH	1	1	2	0	0	1	1	0
DB.Setup	2	2	3	$2n-2$	1	2	$n+1$	1

Table 2
DB.Join – a set of users $U[n+1,\ldots,n+m]$ joins the set of users $U[1,\ldots,n]$, resulting a user set of size $n+m$.

DB.Join	Communication		Computation					
	R	B	Exp	Mul	Div	Sig	Ver	Hash
$U_1, U_2, U[n, \ldots, n+m]$	2	2	3	$2m+4$	1	2	$m+4$	1
$U[3,\ldots,n-1]$		0	2	$2m+4$	0	0	$m+5$	1

Table 3
DB.Leave – users $U_{l_1}, \ldots, U_{l_m}$ leave the set of users $U[1,\ldots,n]$, resulting a new user set of size $n-m$. Users U_{l_i-L}, U_{l_i+R} are respectively the closest non-leaving left and right neighbours of the leaving user U_{l_i} for $1 \leqslant i \leqslant m$. 'Rest of the users' in the table means users in $U[1,\ldots,n] \setminus (\{U_{l_1}, \ldots, U_{l_m}\} \cup \{U_{l_i-L-1}, U_{l_i-L}, U_{l_i+R}, U_{l_i+R+1}, 1 \leqslant i \leqslant m\})$.

DB.Leave	Communication		Computation					
	R	B	Exp	Mul	Div	Sig	Ver	Hash
$U_{l_i-L}, U_{l_i+R}, 1 \leqslant i \leqslant m$	2	2	3	$2(n-m)-2$	1	2	$n-m+1$	1
$U_{l_i-L-1}, U_{l_i+R+1}, 1 \leqslant i \leqslant m$		2	1	$2(n-m)-2$	1	2	$n-m+1$	1
Rest of the users		1	0	$2(n-m)-2$	1	1	$n-m+1$	1

Table 4
Initial Key Agreement: Complexity of sponsor S_i in cluster C_i, where $\hat{n}_i$ denotes the total cluster nodes in C_i, l_i denotes the total number of children at height $h-j$ of the sponsor S_i in the tree T of height h for $0 \leqslant j \leqslant h$ and $1 \leqslant i \leqslant n$ ($l_i < n$).

Protocol	Communication		Computation							
(*IKA*)	*R*	*B*	Exp	Mul	Div	Sig	Ver	Hash	Enc	Dec
AHP − 1	$h+4$	5	l_i+5	$2\hat{n}_i-2$	1	5	$\hat{n}_i+l_i+3$	1	l_i+2	1
AHP − 2	5	5	6	$2\hat{n}_i+2n-4$	2	5	$\hat{n}_i+n+2$	1	1	0

Table 5
Comparing complexity of Step 2 for sponsors in Dynamic Case when the dynamic membership changes have taken place only within the cluster C_i with sponsor S_i, l_i denotes the total number of children at height $h-j$ of the sponsor S_i in the tree T of height h for $0 \leqslant j \leqslant h$ and $1 \leqslant i \leqslant n$ ($l_i < n$).

Protocol	Sponsor	Communication		Computation				
(*GKM*)		*R*	*B*	Exp	Mul	Div	Sig	Ver
AHP − 1	S_i	1	1	l_i+2	0	0	1	l_i+1
Step 2	parent and children of S_i		1	2	0	0	1	1
	other sponsors		0	0	0	0	0	0
AHP − 2	S_i	2	2	3	$2n-2$	1	2	$n+1$
Step 2	S_{i-1}, S_{i+1}		2	1	$2n-2$	1	2	$n+1$
	other sponsors		2	0	$2n-2$	1	2	$n+1$

Table 6
Comparing complexity of Step 3 and/or Step 4 for sponsors in Dynamic Case when the dynamic membership changes have taken place only within the cluster C_i with sponsor S_i, l_k denotes the total number of children at height $h-j$ of the sponsor S_k in the tree T of height h for $0 \leqslant j \leqslant h$ and $1 \leqslant k \leqslant n$ ($l_k < n$).

Protocol	Sponsor	Communication		Computation							
(*GKM*)		*R*	*B*	Exp	Mul	Div	Sig	Ver	Hash	Enc	Dec
AHP − 1 Step 3 & 4	S_k in T at height $h-j$	$j+1$	2	0	0	0	2	1	0	l_k+2	1
AHP − 2 Step 3	S_k in the Virtual ring	1	1	0	0	0	1	0	0	1	0

in the analysis below. For more details on the complexity of cluster formation and cluster maintenance, we refer to [40].

We first provide the complexity of DH, DB.Setup, DB.Join, and DB.Leave in Tables 1–3. Then we provide a comparative summary of performance analysis of our two protocols AHP − 1 and AHP − 2 in Tables 4–6. In the tables, *R* stands for total number of rounds, *B* is the maximum number of broadcast communication per user, Exp is the maximum number of modular exponentiation computed per user, Mul is the maximum number of modular multiplication computed per user, Hash denotes the maximum number of hash function computed per user, Div is the maximum number of division computed per user, Sig is the maximum number of signature generated per user, Ver is the maximum number of signature verification per user, Enc is the maximum number of symmetric encryption computed per user, and Dec is the maximum number of symmetric decryption computed per user.

Complexity of sponsors for Initial Key Agreement (*IKA*) phase is compared in Table 4 for our hybrid authenticated protocols AHP − 1 and AHP − 2. In this phase complexity of cluster nodes other than the sponsors are same for both the protocols and is equal to the complexity of DB.Setup in Table 1, together with 1 additional signature verification and 1 symmetric decryption.

Complexity of nodes for Group Key Maintenance (*GKM*) phase is analyzed below in different steps: In step 1, complexity of cluster nodes within a cluster where membership changes have occurred are same as in Table 2 and/or Table 3 as the case may be. Complexity of all the cluster nodes (including the sponsor of that cluster) in other clusters with no membership change are zero in this step. Step 1 is same for both the protocols AHP − 1 and AHP − 2. Complexity of sponsors in step 2 is compared in Table 5 for both AHP − 1 and AHP − 2. Complexity of cluster nodes other than the sponsors are zero at this step. In step 3 and step 4, complexity of sponsors is compared in Table 6 for both the protocols where AHP − 2 does not have step 4. Complexity of cluster nodes other than sponsors are same for both the protocols in these steps. Consequently, in dynamic case complexity of cluster nodes other than the sponsors are the same for both AHP − 1 and AHP − 2. If they are members of the cluster where a join or leave has occurred, then their complexities are same as that of DB.Join in Table 2 or DB.Leave in Table 3 as the case may be, together with 1 additional signature verification and 1 symmetric

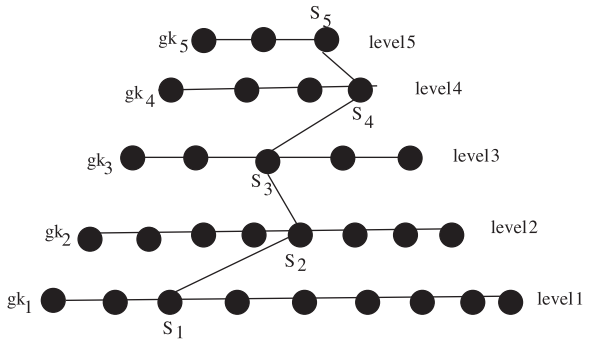

Fig. 8. Example of an hierarchical network of sponsors, HP − 2 is applied in each level to compute the group key gk of that level and HP − 1 is applied for the connection path from sponsor S_1 at level 1 to sponsor S_5 at level 5.

decryption. Otherwise, they perform only one signature verification and one symmetric decryption.

AHP − 1 and AHP − 2 both have advantages over each other. Sponsors in AHP − 1 need more communication rounds and perform more exponentiation and symmetric encryption (for $l_i \geqslant 2$) and an additional symmetric decryption as compared to AHP − 2 for both the *IKA* phase and *GKM* phase. On the other hand, sponsors in AHP − 2 need to compute additionally more multiplications, divisions, and signature verifications as compared to AHP − 1 for both *IKA* and *GKM* phase. Cluster nodes other than the sponsors have the same complexity for both the schemes in *IKA* phase as well as in *GKM* phase. However, symmetric encryption/decryption operations are faster and cheaper, and if $l_i = 1$ for all $i, 1 \leqslant i \leqslant n$ (*i.e.* the connection tree T is unary), then AHP − 1 is more economic than AHP − 2 in terms of computation cost.

Moreover, depending on the application, a more realistic scenario may be the hybrid of AHP − 1 and AHP − 2 as described below. Suppose we have a hierarchical network of sponsors with height h as in Fig. 8. All sponsors at level i, $1 \leqslant i \leqslant h$, can communicate among themselves and make use of AHP − 2 to agree upon a common group key gk_i. We assume that there exists at least one sponsor in each level i who can communicate with a sponsor in the lower level $i - 1$ and a sponsor in the upper level $i + 1$. Then we can apply AHP − 1 along this connection path from the bottom to the top level of the hierarchy among these connecting sponsors. Thus the connection tree in this case is a unary tree. After computing the session key SK by AHP − 1, each sponsor S_i on this connection tree at level i distributes the session key to other sponsors in that level by encrypting it with the group key gk_i and broadcasting the encrypted value. The other sponsors in that level can decrypt it and recover the session key by using their group key gk_i which is common with S_i agreed upon by AHP − 2. Once the sponsors recover the session key, they can distribute it among all the cluster nodes in their respective clusters by using the respective cluster keys CK_i or group keys gk_i as in AHP − 1 or AHP − 2. Note that nodes can communicate securely within a cluster using their common cluster key. Nodes at level i in the hierarchical network can communicate securely among themselves using their common group key gk_i . Nodes in the whole hierarchical network can communicate securely among themselves using their common session key SK in the multi-hop communication channel.

6. Comparison

There are quite a number of group key agreement protocols in literature [16,?], but all are not applicable in ad hoc networks because of dynamic and multi-hop nature of the mobile nodes. Clustering method enables nodes to be organised in an hierarchical ad hoc network based on their relative proximity to one another, thereby weakening the one-hop assumption in common group key agreement protocols. In contrast of invoking the DB protocol from scratch among all the nodes in the system (which may be infeasible in ad hoc environments), we obtain efficiency gain in terms of both communication and computation for most of the nodes in our proposed cluster-based protocols (which are more amenable to wireless ad hoc networks). We now compare our protocols with the existing cluster-based group key agreement protocols [26,39,36,2,25,20] for wireless ad hoc networks.

Li et al. [26] propose a communication efficient hybrid key agreement using the concept of connected dominating set. However, the protocol is inefficient in handling dynamic events. Based on this work, Yao et al. [39] propose a hierarchical key agreement protocol which is communication efficient in handling dynamic events. Both the protocols of [26,39] employ some existed group key agreement protocol such as GDH [37] and are unauthenticated with no security analysis against active adversary. In comparison with these cluster-based hybrid key agreement protocols, our proposed approaches individually provide better performance in terms of both communication and computation, handles dynamic events efficiently, and are supported by sound security analysis in formal security models under standard cryptographic assumptions.

Now according to the results in [27,33], the timing of pairings is more than that of modular exponentiation in RSA [29]. So pairings are not suitable for some time-intensive cases. Our proposed hybrid protocols are designed without using costly operations like pairings and are thus efficient as compared to the existing similar protocols. The existing group key agreement protocols that use clusters and pairings are [36,2,25]. We detail below the comparison of our protocols with the protocols [2,25,36].

In [36], Shi et al. propose a hierarchical key agreement protocol suitable for wireless ad hoc network as it can handle the dynamic events efficiently. They use a ternary tree structure with leaf nodes as users and employ Joux's [22] tripartite protocol and a generalized DH protocol as the basic building blocks. However, this protocol is computationally costly as each user needs to compute h pairings, where h is the height of the cluster key tree. Also the communication round is linear to the number of the nodes for this protocol. The authentication is achieved using an ID-based signature scheme and heuristic arguments are made in support of the security of the protocol. In contrast, our

proposed hybrid protocols are designed without using time-intensive pairings and are thus efficient as compared to the protocol in [36].

Abdel-Hafez et al. [2] provide a partial solution to the key management problem in ad hoc wireless network and proposed two authenticated protocols. The first protocol uses clusters of arbitrary size and requires a trusted authority. However, the protocol is not efficient as it requires n rounds and $n-1$ pairing computations per user, where n denotes the node count in the whole network. The second protocol is essentially an authenticated variant of Burmester–Desmedt [9] protocol. Both these protocols do not handle dynamic membership change and the security analysis is completely heuristic. Moreover, the second protocol does not achieve perfect forward secrecy as claimed by the authors. On the other hand, our hybrid protocols are designed without using pairings and handle dynamic operations efficiently. Additionally, our protocols are proven to be secure in a formal security model.

Recently, Konstantinou [25] propose two protocols using pairings, one is contributory and another is non-contributory. They use the concept of binary cluster trees with each cluster having 3 (or 2) nodes and incorporate Joux's [22] tripartite protocol to construct the group key. The contributory scheme is similar to [12] with the exception that no mechanism is provided to make the tree structure most balanced. The round complexity of these protocols are $O(h)$, where h is the height of the bigger branch of the tree structure. Each participant broadcasts at most three messages, computes at most two scalar multiplications, one symmetric encryption and one symmetric decryption. Additionally, each user computes at most two pairings. However, each internal node in the binary tree structure is a user. Consequently, handling dynamic membership change, especially, the leave operation is very difficult. More precisely, there is no clear description on how to manage a leave event in case the tree structure becomes disconnected due to the leave of internal nodes. Also the protocols are not supported by proper security analysis. The protocols in [25] do not handle group key maintenance phase efficiently and complexity of these protocols may become huge (may be linear) as there is no mechanism in these protocols to make the binary tree most balanced. On the contrary, as mentioned above, our hybrid protocols are highly efficient in handling dynamic events, do not use costly operations like pairings and have concrete security analysis in a formal security model.

Another work by Hietalahti [20] presents a solution that uses BD-I protocol [9] within each cluster and then invokes AT-GDH protocol [19] by employing a spanning tree of sponsors. Both BD-I and AT-GDH protocols are static and consequently handling dynamic events are not easy for this scheme. Our hybrid approaches use protocol DB within each cluster instead of protocol BD-I. The following two advantages of the DB protocol make our proposed protocols highly efficient.

(a) The DB protocol is a provably secure scalable constant round dynamic authenticated group key agreement protocol that uses a ring structure of participants instead of a tree structure. The overhead in reconstructing a tree is more than reconstructing a ring in dynamic setting.
(b) The DB protocol provides efficient join and leave algorithms for dynamic membership change which take advantage of reusability of user's precomputed values in previous sessions in order to save most user's computations for updating session keys in subsequent session in which users join and/or leave the group, retaining the desirable security attributes.

Similar to [20], our protocol first protocol AHP − 1 considers a spanning tree T of sponsors and invokes two-party DH protocol between pairwise adjacent sponsors along the edges of T and distributes a randomly generated session key by the root sponsor in T among all the sponsors from the root level to the leaf level in T using these DH keys. On the other hand, our second protocol AHP − 2 invokes multi-party DB protocol among the sponsors to agree upon a common session key. Moreover, Hietalahti [20]'s work does not provide any security analysis whereas our protocols are supported by proper security analysis in formal security model instead of heuristic arguments.

In summary, we obtain efficiency gain in AHP − 1 as compared to AHP − 2 for dynamic case. For any dynamic membership change, a few sponsor nodes in the spanning tree T in AHP − 1 perform a few two-party DH protocol, whereas in AHP − 2, all the sponsors need to perform a multi-party DB protocol among themselves to agree upon a new common session key. Since in wireless ad hoc network, sponsor's transmission ranges might be limited, it might be difficult for a sponsor to reach the other sponsors in a single-hop. Hence, the network for AHP − 1 is more realistic in the context of wireless ad hoc network. AHP − 1 considers a tree-network of sponsors which is multi-hop in nature, and where only adjacent sponsors in the tree are assumed to be able to communicate among themselves in single-hop. On the contrary, AHP − 2 considers more restrictive network for mobile nodes as compared to AHP − 1 as it assumes a network where all sponsors need to be able to communicate among themselves in single-hop. Moreover, in the light of the discussion above, our proposed protocols are highly efficient as compared to the existing cluster-based key agreement protocols [2,20,25,26,36,39].

7. Conclusion

Ad hoc networks may be logically represented as a set of clusters. The main aim of this work has been to address the problem of providing formal security proofs for protocols handling key agreement and distribution in cluster-based wireless ad hoc networks. This work is largely based on works previously done by us and existing techniques. We have presented two hybrid schemes each handling dynamic operations efficiently as compared to the existing cluster-based hybrid protocols. It may be infeasible in an ad hoc environment with large number of nodes to invoke a group key agreement protocol from scratch among all the nodes in the system. In contrast, our approaches are more

amenable to wireless ad hoc networks with huge number of nodes and enable efficiency gain in both communication and computation for most of the users. Our designs are computationally more efficient as compared to the existing cluster-based key agreement protocols. We have proved both schemes to be provably secure in the standard model under DDH assumption. We have distinguished between the two approaches from a performance point of view and shown that the first scheme is the better scheme in the context of wireless ad hoc networks.

We are currently looking for some realistic simulations to compute the communication overhead and time for cluster management and to evaluate the degree of mobility of nodes under which our hybrid schemes still work reliably. As a future work, we plan to implement our key agreement schemes to analyse performance from a practical point of view in the context of Elliptic Curve Cryptography (ECC) as this is most suitable for resource constrained mobile devices that generates the smaller key sizes and provides more efficient computation compared to other public key cryptosystems. The National Security Agency (NSA) have already selected ECC as the 'recommended' asymmetric crypto-system and the National Institute of Standards and Technology (NIST) is expected to recommend a change in 2010 to Eliptic-curve Cryptograpy for all US government agencies.

Acknowledgment

This material is based upon works supported by the Science Foundation Ireland under Grant No. 06/MI/006.

References

[1] M. Abdalla, M. Bellare, P. Rogaway, DHIES: an encryption scheme based on the Diffie–Hellman Problem, in: Proceedings of CT-RSA, 2001, LNCS 2020, pp. 143–158, Springer-Verlag, 2001.
[2] A. Abdel-Hafez, A. Miri, L. Oronzo-Barbosa, Authenticated group key agreement protocols for ad hoc wireless networks, International Journal of Network Security 4 (1) (2007) 90–98.
[3] D.J. Baker, A. Ephremides, The architectural organization of a mobile radio network via a distributed algorithm, IEEE Transactions on Communications COM-29 (11) (1981) 1694–1701.
[4] D.J. Baker, A. Ephremides, J.A. Flynn, The design and simulation of a mobile radio network with distributed control, IEEE Journal on Selected Areas in Communications (1984) 226–237.
[5] E.M. Belding-Royer, Hierarchical routing in ad hoc mobile networks, Wireless Communication and Mobile Computing 2 (5) (2002) 515–532.
[6] P.S.L.M. Barreto, H.Y. Kim, M. Scott, Efficient algorithms for pairing based cryptosystems, in: Proceedings of Crypto 2002, LNCS 2442, Springer-Verlag, 2002, pp. 354–368.
[7] D. Boneh, M. Franklin, Identity-based encryption from weil pairing, in: Proceedings of Crypto 2001, LNCS 2139, Springer-Verlag, 2001, pp. 213–229.
[8] E. Bresson, O. Chevassut, D. Pointcheval, Dynamic group Diffie–Hellman key exchange under standard assumptions, in: Proceedings of Eurocrypt 2002, LNCS 2332, Springer-Verlag, 2002, pp. 321–336.
[9] M. Burmester, Y. Desmedt, A secure and efficient conference key distribution system, in: Proceedings of Eurocrypt 1994, LNCS 950, Springer-Verlag, 1995, pp. 275–286.
[10] M. Burmester, Y. Desmedt, A secure and scalable group key exchange system, Information Processing Letters 94 (3) (2005) 137–143.
[11] W. Diffie, M. Hellman, New directions in cryptography, IEEE Transaction on Information Theory IT-22 (6) (1976) 644–654.
[12] R. Dutta, R. Barua, P. Sarkar, Provably secure authenticated tree based group key agreement. in: Proceedings of ICICS'04, LNCS 3269, Springer-Verlag, 2004, pp. 92–104.
[13] R. Dutta, R. Barua, Dynamic group key agreement in tree-based setting. in: Proceedings of ACISP 2005, LNCS 3574, Springer-Verlag, 2005, pp. 101–112.
[14] R. Dutta, R. Barua, Constant round dynamic group key agreement. in: Proceedings of ISC 2005, LNCS 3650, Springer-Verlag, 2005, pp. 74–88. <http://eprint.iacr.org/2005/221>.
[15] R. Dutta, R. Barua, Provably secure constant round contributory group key agreement in dynamic setting, IEEE Transactions on Information Theory 54 (5) (2008) 2007–2025.
[16] R. Dutta, R. Barua, Overview of key agreement protocols. <http://eprint.iacr.org/2005/289>.
[17] S. Galbraith, K. Harrison, D. Soldera, Implementing the tate pairing. in: Proceedings of Algorithm Number Theory Symposium – ANTS V, LNCS 2369, Springer-Verlag, 2002, pp. 324–337.
[18] M. Gerla, J.T.-C. Tsai, Multicluster, mobile, multimedia radio network, ACM Baltzer Journal of Wireless Networks 1 (3) (1995) 255–265.
[19] M. Hietalahti, Efficient key agreement for ad hoc networks. Master's Thesis, Helsinki University of Technology, Department of Computer Science and Engineering, Espoo, Finland, 2001.
[20] M. Hietalahti, A clustering-based group key agreement protocol for ad-hoc networks, Electronic Notes in Theoretical Computer Science, vol. 192, Elsevier, 2008. pp. 43–53.
[21] R. Jayakumar, Analysis and study of a spanning tree enumeration algorithm. in: Combinatorics and Graph Theory, Lecture Notes in Mathematics, vol. 885, Springer-Verlag, 1981, pp. 284–289.
[22] A. Joux, A One round protocol for tripartite Diffie–Hellman. in: Proceedings of ANTS 4, LNCS 1838, Springer-Verlag, 2000, pp. 385–394.
[23] J. Katz, M. Yung, Scalable protocols for authenticated group key exchange. in: Proceedings of Crypto 2003, LNCS 2729, Springer-Verlag, 2003, pp. 110–125.
[24] Y. Kim, A. Perrig, G. Tsudik, Tree based group key agreement, ACM Transactions on Information and System Security 7 (1) (2004) 60–96.
[25] E. Konstantinou, Cluster-based group key agreement for wireless ad hoc networks, in: Proceedings of IEEE ARES 2008, 2008, pp. 550–557.
[26] X. Li, Y. Wang, O. Frieder, Efficient hybrid key agreement protocol for wireless ad hoc networks, in: Proceedings of IEEE International Conference on Computer Communications and Networks, 2002, pp. 404–409.
[27] B. Lynn, M. Scott, P.S.L.M. Berreto, H.Y. Lynn. Efficient algorithms for pairing-based cryptosystems, in: Proceedings of Crypto 2002, LNCS 2442, 2002, pp. 354–369.
[28] M. Manulis. Security-Focused Survey on Group Key Exchange Protocols. <http://eprint.iacr.org/2006/395>.
[29] A. Menezes, P.C. Van Oorschot, S. Vanstone, Handbook of Applied Cryptography. CRC Press, Boca Raton, 1997. <http://cacr.math.uwaterloo.ca/hac>.
[30] NIST, AES, December 2000. <http://www.nist.gov/aes>.
[31] National Bureau of Standards, Data Encryption Standard, US Department of Com- merce, FIPS pub. 46, 1977.
[32] G. Pei, M. Gerla, X. Hong, C.C. Chiang, A wireless hierarchical routing protocol with group mobility, in Proceedings of IEEE WCNC 1999, 1999, pp. 1538–1542.
[33] M. Scott, Computing the tate pairing, in: Proceedings of CT-RSA 2005, LNCS 3376, Springer-Verlag, 2005, pp. 293–304.
[34] M. Scott, N. Costigan, W. Abdulwahab, Implementing cryptographic pairings on smart cards. <http://www.iacr.org/2006/144>.
[35] J. Sucec, I. Marsic, Clustering overhead for hierarchical routing in mobile adhoc networks, in Proceedings of IEEE Infocomm 2002, 2002, pp. 1698–1706.
[36] H. Shi, M. He, Z. Qin. Authenticated and communication efficient group key agreement for clustered ah hoc networks, in: Proceedings of CANS 2006, LNCS 4301, Springer-Verlag, 2006, pp. 73–89.
[37] M. Steiner, G. Tsudik, M. Waidner, Diffie–Hellman key distribution extended to group communication, in: Proceedings of ACM CCS 1996, ACM Press, 1996, pp. 31–37.
[38] M.N.S. Swamy, K. Thulasiraman, Graphs, Networks, and Algorithms, Wiley-Interscience, New York, 1981.
[39] G. Yao, K. Ren, F. Bao, R.H. Deng, D. Feng, Making the key agreement protocol in mobile ad hoc network more efficient, in: Proceedings of ACNS'03, LNCS 2846, Springer-Verlag, 2003, pp. 343–356.
[40] J.Y. Yu, H.J.P. Chong, A survey of clustering schemes for mobile ad hoc networks, IEEE Communications – Surveys and Tutorials 17 (1) (2005) 32–48 (first quater).

Ratna Dutta is currently an assistant professor in the Department of Mathematics, Indian Institute of Technology, Kharagpur, India. She received her B.Sc (in Mathematics Hons.) and M.Sc (in Applied Mathematics) from University of Calcutta, India in 1996 and 1998 respectively. She received her PhD degree in the area of Cryptology (Computer Science) from Indian Statistical Institute, India in 2006. She visited ENSTA, Paris in 2006 and worked as an associate scientist at the Institute for Infocomm Research, Singapore during 2006–2008. She was also a post-doctoral research fellow in the Department of Computer Science, National University of Ireland, Maynooth during 2008–2010. Her research interests include public key cryptography, elliptic curve cryptosystem and pairings, dynamic group key agreement, password-based protocols, word-based public key cryptography and wireless ad hoc networks.

Tom Dowling is a lecturer and researcher with the computer security and cryptography group at the National University of Ireland Maynooth. He received a degree in Applied Physics and Mathematics from Dublin Institute of Technology, Ireland in 1991. His M.Sc. and Ph.D. are in pure Mathematics from the National University of Ireland. His research interests include cryptography, network security, smart cards, and numerical computing.