Naresh *et al. Hum. Cent. Comput. Inf. Sci.* (2019) 9:26
https://doi.org/10.1186/s13673-019-0186-5

Human-centric Computing and Information Sciences

RESEARCH Open Access

A provably secure cluster-based hybrid hierarchical group key agreement for large wireless ad hoc networks

Vankamamidi S. Naresh[1*], Sivaranjani Reddi[2] and Nistala V. E. S. Murthy[3]

*Correspondence:
res.naresh@gmail.com
[1] Department of Computer Science and Engineering, Sri Vasavi Engineering College, Tadepalligudem, Andhra Pradesh 534101, India
Full list of author information is available at the end of the article

Abstract

Group key agreement protocol permits a set of users to create a common key to make sure security of information exchange among members of the group. It is extensively used in secure multiparty computation, resource security sharing, and distributed collaborative computing etc. For large wireless ad-hoc network, there is no authentication center, the computing power and communication distance of terminals are constrained, and nodes frequently join and exit the network. For these reasons, Group Key Management for securing multicast communications in an energy-constrained large wireless ad-hoc network environment is still remains a critical and challenging issue. In this direction, we propose a cluster-based hybrid hierarchical-group key agreement (CHH-GKA) framework to provide a scalable solution for Secure Group Communication (SGC) in large wireless ad hoc networks (WANETs). This technique is based on splitting a large group into a certain number of clusters in which the last member of each of the clusters is designated as a cluster head (CH) and the last member of the group is designated as the group controller (GC). First we apply on hand Naresh–Murthy-group key agreement (NM-GKA) protocol locally in every cluster in parallel in level-I to generate CKs and then in level-II, the CHs' use these CKs and implement NM-GKA protocol again among them to form the complete group key. Finally each CH distributes the group key to all its members through their respective CK encrypted links. In this process, first we survey several cluster-based hierarchical GKA protocols and compare the proposed one with them and show that it provides optimal performance with regard to computation and communication expenses. Further, it also handles dynamic events and is provably secure in formal security model under the cryptographic suppositions.

Keywords: Secure Group Communication, Elliptic curve Diffie–Hellman (ECDH), Hierarchical, Clustering, Hybrid, Group key agreement, Wireless ad hoc networks

Introduction

WANETS provide whenever–wherever networking amenities for communication establishment through the public wireless medium. In this environment, Secure-GKA and proficient group key management are known to be complicated tasks with respect to both computational and algorithmic points of view because of resource constraints in WANET [1]. There is an extensive range of applications for WANET which includes emergency medical services deployed in various environments which can considerably improve the quality of medical care; military applications, rescue missions, collaborative

commercial applications; law enforcement etc. Security is the decisive factor for designing an efficient Wireless Sensor Network (WSN) protocol. Consequently, secure GKA protocols have gained extensive attention. We presented considerable number of GKA schemes in the literature [2–11]. However, traditional GKA protocols are *not* appropriate for ad hoc networks. The principal challenge here is provision of secure authenticated communication which comes from their distinctive features which include (i) need for a fixed trustworthy Public Key Infrastructure (PKI); (ii) need to support dynamic network topology as a result of high mobility like joining/leaving; (iii) nodes with less amount of storage, computation and communication power; (iv) be deficient in predisseminated symmetric keys among the peers; (v) higher level of self-network arrangement; (vi) susceptible multi-hop wireless connections, etc.

In large WANETs, establishment of group key [1] is a tricky job due to its dynamism. A usual solution suggested to address this issue is to split up the large network into a certain number of constituent network clusters [12]. Categorization of the clustering algorithms can be done by the type of clusters they are forming. Several clustering algorithms pick special nodes as CHs, responsible for cluster creation and afterward-maintenance of the cluster [13], at times routing also. The CHs are not always mandatory. A few protocols used in clustering algorithms do not use them at all. Instead, they prefer gateways to communicate messages from one cluster to another. A gateway generally fits to more than one cluster if there is an overlap in the clusters. In depth description relating to some of these clustering algorithms can be found, for example, in [14].

The implementation of GKA and key management are easier within the cluster in contrast to the complete ad hoc network. Since the clusters have further stable internal links because of the huge quantity of connections among peers within the similar cluster. Further, inter-cluster GKA is meaningful as clusters are put on to stick jointly more than the hops do on average for WANETs. Clustering may thus fetch the essential scalability and failure in one cluster does *not* affect the whole group for establishing the group key in large networks. Thus clustering was adopted in the proposed work.

A public key cryptography is used in majority of distributed GKA protocols because there are *no* alternative approaches available for distributing a common key through a public channel. The Public Key Computations (PKC) methods, as well as D-H's exponentiation are both costly and very difficult for WANET. While distributing extra common keys to nodes that have embarrassed capabilities or bandwidth of storage and computation. The management techniques for computational overheads must be considered into account. As ECDH is lightweight and efficient when compared to regular DH, the ECC-base [15] is used to secure dynamic authenticated GKAs: Consequently, in this paper an ECC-based NM-GKA [16] is used as a pre-requisite for the proposed protocol.

In hierarchical framework, a network is formulated from a nested grouping (clustering) of nodes, connected in the form of a tree structure. Hierarchical frameworks are often utilized in routing as in [12, 17, 18], where best clustering frameworks are derived so as to minimize routing table's size. Numerous protocols require the information of the entire topology of network, whereas others carry out the computations with the knowledge of the nearby nodes and their likely cluster-memberships [13, 19]. After having a thorough study of these existing hierarchical and cluster-based protocols, we derived some notable merits which include (i) a hierarchical structure is adopted to handle the

dynamic events efficiently, (ii) a hybrid encryption is employed as this approach can reduce the computation overhead, and therefore, it is quite suitable for WSN. Some common drawbacks in the existing hierarchical protocols which include (i) the clustering method is not easy to handle certain member events, such as a CH node leaving the network. More precisely, it is rather costly to use the cluster method to deal with the situation that several CH nodes leave the network at the same time. (ii) Distinct complex algorithms should be carefully designed for handling different kinds of dynamic events. On the other hand, as it was stated in [20, 21], when every cluster are having the same amount of nodes and sizes, the hierarchical framework becomes fully balanced and also achieves the best performance. Besides, the authors of [21] asserted that the competence of the entire scheme is enriched if the amount of levels is little (let it be 3). In this work, a fully balanced hierarchical framework of level 2 was adopted with all the clusters with equal size except for one.

The proposed work has adopted hybrid based symmetric encryption where it combines the key distribution and key agreement. A digital signature scheme as in [16] can be used to authenticate our protocols. In view of the MANET's (Mobile Adhoc Network) dynamic, the proposed protocols adeptly address the dynamic events. It is designed exclusive of utilizing calculation-exhaustive pairings [22] and is extremely efficient relative to the existing hybrid cluster-based GKA protocols [20, 23–29].

In contrast, usage and implementation of NM-GKA [16] protocol among all the nodes in the system may *not* be feasible for large WANETs. Consequently, we plan to use the same for each cluster and then for all the CHs in two levels hierarchically.

Notice that this paper assumes that the cluster structure has already been established (includes the amount of levels in the cluster hierarchy, formation of clusters [20, 21, 25, 26, 30–33] and the selection of CHs) and thus does not consider overhead computation during the cluster-setup phase.

Related work

Two-party DH-key agreement [34] is the origin for enormous amount of consequent GKA schemes. The majority of distributed/contributory-GKA protocols rely on generalizations of 2-party DH or its extensions [3, 7, 16, 35–39]. Key management in distributed/contributory-GKA are less difficult to deal with in each subgroup/cluster compared to the whole ad hoc network. So most recent works [18, 20, 21, 25, 26, 30, 31, 35, 39, 40–42] adopted subgroup/cluster based approach, in which the whole group is divided into clusters. Distinct controllers are utilized to control every cluster which minimizes the issue of imposing the work on a single point.

The majority of CK-GKAs' [18, 20, 21, 25, 26, 30, 31, 35, 39, 40–42] presume a hierarchical framework of the clusters or hierarchical structure, then execute a natural key agreement schemes such as, D-H [34] or the Burmester and Desmedt (BD) [3] GKA scheme, or a variety GKA schemes [3, 7, 16, 35–39] is at first implemented locally in each cluster, after that utilize these CKs in the next level with equivalent or an alternate key agreement scheme among CHs' to generate the whole group key. For further information on a comparison of the existing protocols [18, 21, 25, 26, 28, 30, 31, 33, 43] in this direction, one can refer to Table 2, summary of the key characteristics of cluster based protocols.

In the existing cluster-based GKA protocols, only [20, 26, 28, 31] offer authentication. Authentication confirms that only legitimate group members are allowed to derive the key in the key setup phase and accordingly facilitate the group members to secure against MITM attacks in the course of the key agreement phase. In the schemes [18, 21, 29], the authors suggest an approach of making their scheme into an authenticated approach, but doesn't analyse the additional communication and computation cost in order to authenticate each and every message which is shared among the group members. Lastly, some protocols [25, 44] did not even consider the authentication mechanism at all in key agreement phase. On the other hand, these schemes can be altered in order to accomplish authentication by means of either a special kind of compiler or an authenticated GKA (AGKA) [45].

Further, most of the traditional GKA schemes stated in the literature are unable to handle the dynamic nature (joining and leaving of nodes from the clusters) in WANETs. In precise, the renowned protocols in [3, 11, 36, 38] competent for wired networks, may not be applicable to the WANETs due to their enormous dynamism. On the other hand, clustering strategy empowers hubs to be sorted out in a various levelled ad hoc network dependent on their relative nearness to each other, along these lines debilitating the one hop presumption in natural GKA protocols.

After a thorough study in examined research area, in this work we adopted cluster based hybrid hierarchical approach: dynamic cluster-based hybrid hierarchical group key agreement for large wireless ad hoc networks.

Our contribution

The key objective of this work is to achieve "a provably secure CHH-GKA for large WANETs". The base behind the proposed creation is to divide and conquer. This protocol works by dividing larger group into a certain number of clusters created on their relative closeness to each another. For this we employ two types of keys namely group key (GK) and cluster key (CK). A CK is nothing but the key produced among every member inside a cluster and the GK is the complete network key among every node in the group.

In this work we choose dynamic authenticated NM-GKA protocol [16] for establishment of the CKs in level-I and then for GK in level-II. Further, the last member of each cluster will act as its CH and generates the CK among the cluster members in level-I. The last member of the group will act as the GC for the entire group and combines all the CKs to create the GK. Key for the entire group in level-II. This scheme reduces the computational complexity $O(lr)$ to $O(l+r)$ where $l=Max$ ($|C1|$, $|C2|$,...,$|Cr|$) and "r" is the number of clusters.

For building provably secure model for the proposed protocol we adopted Bresson et al.'s [46] because it is the first formal provably secure model for authenticated GKA. The concept of provable security is utilized over the contemporary literature to demonstrate in a mathematical means, and under sensible suppositions, that a cryptographic technique accomplishes the essential goals of security. Such proofs are generally build by means of a formal setting that indicates: (1) the computing environment (involving cryptographic parameters, users, their trust association, communication etc.), (2) the adversarial environment and (3) the definitions of a few solid goals of security.

Overall contribution

i. The key contribution of this work is authenticated cluster-based hybrid hierarchical GKA: NM-CHH-GKA for large wireless ad hoc networks.
ii. Extended NM-CHH-GKA to dynamic NM-CHH-GKA by proposing join and leave of single or multiple group members for membership changes.
iii. Established recognized proofs of provable security for to dynamic NM-CHH-GKA.
iv. Our comparative analysis assessed and measured the effectiveness of proposed protocol and compared with identified protocols in terms of energy cost for computation and communication and shown that the proposed protocol is optimal.

Some salient features of the proposed scheme

i. Different CH are used to control each cluster and it minimizes the total load on a single point (GC).

   For instance consider one of the most promising applications [24] of cluster-based hierarchical GKA over WSNs in the healthcare sector.

   NM-CHH-GKA over infrastructure-based WSN situation is appropriate for medical environments in which one can have numerous powerful nodes those can take CH role, such as intra-hospital environments. We can then suppose that CHs are predetermined and that consumption of energy is not a principal concern for them. The hospital sensor network can be split into various clusters by considering their geographical location.

   NM-CHH-GKA over infrastructure-less WSN situation is appropriate for medical environments in which there is no fixed infrastructure at all or no full coverage, as in the case of a medical emergency. In this situation, dynamically sensors can be clustered into non-overlapping or overlapping groups. Whenever a node wants to send out data, the node closer to the gateway (best path) is selected as the CH. For further information please refer [24].
ii. The failure of one CH or node doesn't affect the entire group.
iii. Parallel computation of CKs provides reduced computational load from $O(l{\cdot}r)$ to $O(l+r)$.
iv. Both membership changes and subgroup dynamics can be optimally achieved.
v. Local rekey: membership change in a cluster are treated locally, so that rekey of a cluster will not disturb the entire GK.
vi. The two level cluster based hierarchical GKA scheme allows distributed key management scheme to implement at the cluster level to realize dynamism without losing efficiency.
vii. The two level GKA reduces load on the GC by distributing or arranging the group members in the form of hierarchy, which enhances scalability and security.
viii. Every cluster member requires a minimum storage space to preserve the CKs.

Table 1 Notations

Symbol	Comment
F_p	The finite field with cardinality p
E	An EC equation by weierstrass
$E(F_p)$	An EC-group over F_p
P, Q	EC-points $\in E(F_p)$
$P+Q$	The sum of P and Q
$[k]P$	The k-th multiple of a point P
x_P (or) y_P	The x and y coordinates of point P respectively
P	The base point of $E(F_p)$
N	The order of P. Usually, N is a large prime number of bit length ≥ 224
M_i	The ith member of the group, $1 \leq i \leq n$, where n the total group members
M_n	The last member of the group is the group controller (GC)
C_i	The ith cluster, $1 \leq i \leq r$, where $r =$ number of clusters
x_{s_i}	The CKs of C_i, $1 \leq i \leq r$
CK_i	The ith updated CK for new cluster, $1 \leq i \leq r$
M_{ij}	The jth member of ith cluster, $1 \leq i \leq r$, $1 \leq j \leq n$
M_{i_l}	The CH of ith cluster and the last member of that cluster
x_i	The M_i's private key, an integer belongs to $[1, N-1]$
$x \leftarrow [N-1]$	Pick an integer $x \in [N-1]$
X_i	The M_i's public key
$x_{K_{ij}}$	ECDH common key between ith member M_i and jth member M_j

Organization/structure of the paper "Background protocols" section talks about the protocol's prerequisites. The proposed protocol is exhibited in "Proposed protocol" section. "Security analysis" section speaks about analysis of security. "Comparative analysis" section delivers a relative analysis with the existing prominent protocols. Finally, "Conclusion and future scope" section concludes with several observations and future scope.

Background protocols

Here first we introduce several notations presented during the course of the paper and then we present the backbone on hand NM-GKA protocol.

Notations

The several notations utilized in this paper are presented in Table 1.

Naresh–Murthy group key agreement protocol (NM-GKA)

Let M_1, M_2,...,M_i,...,M_n be the members of group and let M_n the last member be the *GC*. As shown in Fig. 1, in round-1, the GC M_n establishes $(n-1)$, 2-party ECDH common keys with every residual members. During round-2 the GC generates $(n-1)$ public keys L_i by means of 2-party keys generated in round-1 after that it sends these public keys to the corresponding members and on getting, every member products it with their own common key in order to calculate the GK. Further the GC combines all the 2-party keys generated in round-1 into a GK and it turn into a part of the group. Authentication is provided with a digital signature (DSig) as in [16]. The NM.Initial group key agreement (NM.IGKA) protocol is presented in Fig. 1. Further

Round 1:

Group Members : M_1 M_2 ... M_i ... M_n (GC)

• • ... • ... •

Private Keys: x_1 x_2 ... x_i ... x_n

Public Keys: x_1P x_2P ... x_iP ... x_nP

Communication:

$M_n \xrightarrow{x_nP} M_i,\ 1 \le i \le n$

$M_i \xrightarrow{x_iP} M_n, 1 \le i \le n$

Computation:

M_n:$K_{n,i}$=$x_nX_i = x_n(x_i\,P) = x_nx_i\,P = \left(x_{K_{n,i}}, x_{K_{n,i}}\right)$,

$1 \le i \le n-1$

M_i:$K_{n,i}$=$x_iX_n = x_i(x_n\,P) = x_ix_n\,P = \left(x_{K_{n,i}}, x_{K_{n,i}}\right)$,

$1 \le i \le n-1$

Therefore $x_{K_{n,i}}$ are the "*n-1*" common keys amid of the GC M_n and the residual member $M_i, 1 \le i \le n-1$of the group, respectively.

Round 2:

Computation:

M_n: $L_i = \left[\prod_{j=1,j\neq i}^{n-1} x_{K_{n,j}}\right]P,\ 1 \le i \le n-1$

Communication:

$M_n \xrightarrow{L_i} M_i, \quad 1 \le i \le n-1$

GK Computation:

M_i: $K = \left[x_{K_{n,i}}\right]L_i$

$= \left[x_{K_{n,i}}\right]\left[\prod_{j=1,j\neq i}^{n-1} x_{K_{n,j}}\right]P$

$=\left[\prod_{i=1}^{n-1} x_{K_{n,i}}\right]P$

$=(x_K, y_K)$

M_n: As the GC be acquainted with every common key, it also creates the *GK* as under:

$K = \left[\prod_{i=1}^{n-1} x_{K_{n,i}}\right]P$

$=(x_K, y_K).$

Therefore x_K is now a *GK* among the group participants.

Fig. 1 NM.IGKA protocol

we presented NM-GKA dynamic protocols [16], NM.Join and NM.Leave in Figs. 2 and 3 respectively.

Proposed protocol

Here we presented an outline of the proposed protocol and then the detailed proposed protocol.

Outline of the proposed scheme: NM-CHH-GKA

The proposed scheme consists of 4 steps as follows:

i) Once a fresh participant M_{n+1} wishes to join the group, it informs the M_n (GC) and produces an ECDH key $x_{K_{n,n+1}}$ with GC utilizing ECDH.

ii) The GC produces R'_{n+1} a random number and broadcasts $[x_{K_{n,n+1}}.\ R'_{n+1}]\ P$ to every earlier participants M_i of the group.

On getting, they establish the New Join Group Key (NJGK) as under:

$$NJGK = (PGK)\ [x_{K_{n,n+1}}.\ R'_{n+1}]$$
$$= \left[\prod_{i=1}^{n-1} x_{K_{n,i}}\right] P\ .\ [x_{K_{n,n+1}}.\ R'_{n+1}]$$
$$= \left[\prod_{i=1,i\neq n}^{n+1} x_{K_{n,i}}.\ R'_{n+1}\right] P,$$

Where PGK is the previous GK

iii) The GC sends $[PGK.\ R'_{n+1}]$ to M_{n+1}.

Now, M_{n+1} compute the fresh key as follows:

$$NJGK = [PGK.\ R'_{n+1}]\ .\ x_{K_{n,n+1}}$$
$$= \left[\left[\prod_{i=1}^{n-1} x_{K_{n,i}}\right] P\ .\ R'_{n+1}\right]\ .\ x_{K_{n,n+1}}$$
$$= \left[\prod_{i=1,i\neq n}^{n+1} x_{K_{n,i}}.R'_{n+1}\right] P$$

Fig. 2 NM.Join protocol

i) Once M_j desires to withdraw from the group, it informs the GC. Then the GC, M_n produces R'_j a random number.

ii) The GC M_n sends $[x_{K_{n,j}}{}^{-1}\ \ R'_j]$ by encrypting with $x_{K_{n,i}}$ to the consequent group participant M_i, $i\neq j$, (i.e) excluding leaving participant. On getting each participant by M_i decrypts with $x_{K_{n,i}}$ and calculates the New Leave Group Key ($NLGK$) as:

$$NJGK = (PGK)\ [x_{K_{n,j}}{}^{-1}\ .\ R'_j]$$
$$= \left[\prod_{i=1}^{n-1} x_{K_{n,i}}\right] P\ [x_{K_{n,j}}{}^{-1}\ .\ R'_j]$$
$$= \left[\prod_{i=1,i\neq j}^{n-1} x_{K_{n,i}}.R'_j\right] P$$

iii) Also M_n calculates the fresh key as under:

$$NJGK = (PGK)\ [x_{K_{n,j}}{}^{-1}\ .\ R'_j]$$
$$= \left[\prod_{i=1}^{n-1} x_{K_{n,i}}\right] P\ [x_{K_{n,j}}{}^{-1}\ .\ R'_j]$$
$$= \left[\prod_{i=1,i\neq j}^{n-1} x_{K_{n,i}}.R'_j\right] P$$

Fig. 3 NM.Leave protocol

Step 1: (Cluster key agreement) In this step parallel execution of NM-GKA protocol in all the clusters for computing their respective CKs as in Algorithm 1.

Algorithm 1. (Procedure for Cluster Key Agreement): Calculation of CK x_{s_i} among every node of the cluster C_i, $1 \leq i \leq r$.

Cluster_key_Agreement ()
Input: C_i, 1 ≤ i ≤ r Cluster nodes
Output: x_{s_i} cluster key
1: for i =1 to r do in parallel
2: call NM-Setup (C_i)
3: Choose x_{s_i} share it to cluster Ci nodes
4: end for
end Cluster_key_Agreement

Step 2: (Group key agreement) In this step execution of NM-GKA protocol among all the CHs for computing their complete GK as in Algorithm 2.

Algorithm 2. (Procedure for Group Key Agreement): Computation of GK among the CHs M_{i_l}, $1 \leq i \leq r$ and the last CH being GC $M_{r_l} = M_n$

Group_key_Agreement ()
Input: M_{i_l}, 1 ≤ i ≤ r Cluster Head nodes
Output: x_k group key
1: initialize GC to M_{r_l}
2: for i =1 to r-1 do in parallel
2: call NM-GKA(M_{i_l}, GC)
3: Choose x_K share it to M_{i_l} nodes
4: end for
end Group_key_Agreement

Step 3: (Group key distribution among the cluster nodes) In this step each of the CH distributes the established GK in step-2 to their members through their respective CK encrypted links.

Algorithm 3: (Procedure group key distribution to cluster nodes): Each CH M_{i_l} encrypts the x_K with their respective CK x_{s_i} and broadcasts this within their cluster. Now every member of the cluster decrypts the message with its CK and gets the GK, x_K.

Group_Key _Cluster_Distribution()
Input: M_{i_l}, 1 ≤ i ≤ r Cluster nodes, r: no of clusters, l: no of nodes in cluster
1: for i =1 to r do in parallel
2: Initialise M_{i_l} to CH
3: for j =1 to l-1 do in parallel
4: Encrypt (x_k) $using\ x_{s_j}$ and broadcast to C_j
5: end for
6: for j =1 to l-1 do in parallel
7: Decrypt(Encrypt (x_k)) $using\ x_{s_j}$ to recover x_k
8: end for
9:end for
end Group_Key _Cluster_Distribution()

Step 4: (Group key maintenance) As per the dynamic nature of wireless nodes, the nodes' movement may vary the topology of network often. It is consequently significant and essential to update session key of the group to guarantee security. For establishing new GK, in level-1 we renew the CKs where changes in membership arise by call upon CK update as in Algorithm 4 and then in level-2 by invoking GK update as in Algorithm 5.

Algorithm 4. (Procedure for Cluster Key Update)
In level-1, update of CK of C_i as soon as a set of nodes V leaves / joins C_i.

```
Cluster_key_update()
Input : P: node willing to join or leave
        Ci : Cluster i
      type: join or leave
Output: Updated Cluster key CKi
1:  Initialize V to P
2:  if (type is join) call NM.Join(Ci, V)
3: else call NM.leave(Ci, V)
4: end if
5: return CKi be the updated CK for the new cluster
End Cluster_key_update()
```

Algorithm 5. (Procedure for Group Key Update)
Let CK updated in the cluster C_i as soon as a set of users V leaves/ joins C_i is CK_i.
In level-2

```
Group_key_update()
Input : P: node willing to join or leave
        Ci : Cluster i
      type: join or leave
      level:1 or 2
Output: Updated Cluster key CKi
1:  Initialize V to P
2: if(level==1) Cluster_key_update(Ci, V, type)
3: else if (type is join) call NM.group_Join(Ci, V)
3: else call NM.leave(Ci, V)
4: end if
5: Share updated key securely  to all nodes in the group
End Group_key_update()
```

Proposed scheme: NM-CHH-GKA

NM-CHH-IGKA

Let M_1, M_2, M_3..., M_n be the group members. Without loss of generality, for computation sake, divide these "n" members into $r = \lceil \frac{n}{l} \rceil$ clusters, where cardinality of each cluster C_1, C_2, ..., C_r is less than or equal to l and also let the last member of each cluster act as CH and let the last member of entire group act as the GC for the whole group.

Level-I: CK generation for any of the cluster C_i, $1 \leq i \leq r$.

Let $C_i = \{M_{i_1}, M_{i_2}, \ldots, M_{i_{l-1}}, M_{i_l}\}$ where M_{i_l} is the CH of C_i, $1 \leq i \leq r$.

Notice that the rth cluster may *not* have l members in it. However, the procedure remains the same with a different suffix other than l.

Step 1: The ith CH M_{i_l} forms $(l-1)$ two-party groups with the remaining members of that cluster $M_{i_1}, M_{i_2}, \ldots, M_{i_{l-1}}$ and generates two-party ECDH style keys $x_{K_{l,j}},\ 1 \le j \le l-1$ as follows:

i. The CH M_{i_l}, chooses a private key x_l and generates its public key $X_l = [x_l]P$
ii. Remaining cluster members $M_{i_j},\ 1 \le j \le l-1$, chooses private keys x_j and generates their respective public keys $X_j = \left[x_j\right]P,\ 1 \le j \le l-1$.
iii. The CH broadcasts its public key X_l to the remaining members of the cluster and each $M_{i_j}, 1 \le j \le l-1$ unicasts X_j to the CH M_{i_l}.
iv. After exchanging their public key each member M_{i_j} in the cluster C_i, computes its shared key $K_{l,j}$ with the CH M_{i_l} as follows:

$$\begin{aligned} K_{l,j} &= \left[x_j\right] X_l = \left[x_j\right][x_l]P \\ &= \left[x_j x_l\right] P \\ &= \left(x_{K_{l,j}},\ y_{K_{l,j}}\right), \quad 1 \le j \le l-1. \end{aligned}$$

v. Similarly, the CH M_{i_l} computes $(l-1)$ shared keys $K_{l,j}$ with the remaining cluster members $M_{i_j},\ 1 \le j \le l-1$ as follows:

$$\begin{aligned} K_{l,j} &= [x_l]\, Xj = [x_l]\left[x_j\right]P \\ &= \left[x_l x_j\right]P \\ &= \left(x_{K_{l,j}}, y_{K_{l,j}}\right), \quad 1 \le j \le l-1. \end{aligned}$$

Thus $x_{K_{l,j}} 1 \le j \le l-1$ are the $(l-1)$ shared keys between the CH M_{i_l} and other members M_{i_j} of the cluster C_i, where $1 \le i \le r$ and $1 \le j \le l$ in that order.

Step 2: Currently the CH calculates the $(l-1)$ public keys L_j, using two-party common keys $x_{K_{l,j}},\ 1 \le j \le l-1$ established in step 1, as below and sends it to respective M_{i_j}.

Public keys:

$$L_j = \left[\prod_{m=1, m\neq j}^{l-1} x_{K_{l,m}}\right] P, \quad for\ 1 \le j \le l-1.$$

After unicast messages are received by respective members M_{i_j} compute the CKs as under:

$$\begin{aligned} S &= \left[x_{K_{l,j}}\right] L_j \\ &= \left[x_{K_{l,j}}\right]\left[\prod\nolimits_{m=1,m\neq j}^{l-1} x_{K_{l,m}}\right] P \\ &= \left[\prod\nolimits_{j=1,}^{l-1} x_{K_{l,j}}\right] \mathrm{P} \\ &= \left(x_s, y_s\right). \end{aligned}$$

As CH be acquainted with all the common keys, it also establishes the CK as under:

$$S = \left[\prod_{j=1,}^{l-1} x_{K_{l,j}}\right] P = (x_s, y_s).$$

Thus x_s is the CK among the cluster members C_i.

Now, let the CK of C_i be $x_{s_i},\ 1 \le i \le r$.

Level-II: Let $M_{1_l}, M_{2_l}, \ldots, M_{r-1_l},\ M_{r_l}$ be the CHs and let $M_{r_l} = M_n$ be the GC.

Step 1: Let x_{s_i} be the CK of the respective cluster $C_i,\ 1 \le i \le r$ generated in level-I. First the GC M_{r_l} forms $(r-1)$ 2-party groups with the residual CHs and each CH M_{i_l} takes the CKs generated in level-I $x_{s_i},\ 1 \le i \le r$ as their private key respectively and computes their respective public keys as follows:

$$S_i = [x_{s_i}]P, \quad 1 \le i \le r.$$

The GC, M_{r_l} broadcasts its public key S_r to the remaining CHs $M_{i_l},\ 1 \le i \le r-1$.

After receiving each CH M_{i_j} computes the shared key between GC and itself as follows:

$$T_{r,i} = [x_{s_i}]S_r = [x_{s_i}][x_{s_r}]P = [x_{s_i}x_{s_r}]P = (x_{T_{r,i}},\ y_{T_{r,i}}), \quad 1 \le i \le r-1.$$

Each CH M_{i_l}, unicasts its public key x_{s_i} to GC M_{r_l} and then GC computes the $(r-1)$ shared keys with the remaining CHs as follows:

$$\mathrm{T}_{r,i} = [x_{s_r}]S_i = [x_{s_r}][x_{s_i}]P = [x_{s_r}x_{s_i}]P = (x_{T_{r,i}}, y_{T_{r,i}}), \quad 1 \le i \le r-1.$$

Thus $x_{T_{r,i}}\ 1 \le i \le r-1$ are the $(r-1)$ common keys between the GC M_{r_l} and the other CHs M_{i_l}, where $1 \le i \le r-1$.

Step 2: Currently the GC calculate the $(r-1)$-public keys U_i, by means of two party common keys $x_{T_{r,i}},\ 1 \le i \le r-1$, generated in step 1, and sends it to respective CHs $M_{i_l}\ 1 \le i \le r-1$ as follows:

Public keys:

$$U_i = \left[\prod_{j=1, j\neq i}^{r-1} x_{T_{r,j}}\right] P, \quad for\ 1 \le i \le r-1.$$

After receiving respective unicast messages, respective CHs M_{i_l} compute the GKs as follows:

$$K = [x_{T_{r,i}}]U_i = [x_{T_{r,i}}]\left[\prod_{j=1,j\neq i}^{r-1} x_{T_{r,j}}\right]P = \left[\prod_{j=1}^{r-1} x_{T_{r,j}}\right]P = (x_K, y_K).$$

In view of the fact that the GC knows every common key, it also generates the *GK* as under:

$$K = \left[\prod_{j=1}^{r-1} x_{T_{r,j}}\right] \mathrm{P} = (x_K, y_K).$$

Hence the x_K is the *GK* among the group members. Authentication is provided with a digital signature (DSig) as in [16].

NM-dynamic CCH protocol (NM-DCHH)

To address the dynamic events such as join and leave in GKA we proposed a *NM-DCCH-GKA by* introducing NM-CHH.Join protocol and NM-CHH.Leave protocol as follows:

NM-CHH.Join protocol The principal security prerequisite of member joining is the protection of the earlier *GK* from both the outsiders and the newly joining group members.

Suppose a node or a set of nodes *U* wish to join the group and intimates the same to GC. The GC adds *U* at the beginning of the cluster C_i where it belongs so that the CH remains the same. We proceed with NM-CHH-Join protocol as shown in Fig. 4.

NM-CHH.Leave protocol The principal security prerequisite when a member leaves is the protection of the succeeding (future) *GK* from both the outsiders and the earlier leaving group nodes.

We may assume that this member is not a CH without loss of generality, because if it is the GC and/or CH, naturally the preceding member will act as GC and/or CH and the procedure still remains the same.

Suppose a node or a set of nodes *U* want to leave the group and intimates the same to GC. We proceed with NM-CHH-Leave protocol shown in Fig. 5.

Level-I :
When a single or a set of users *U* when a member desires to join a cluster C_i, the CK CK_i is updated by implementing NM.Join protocol among $C_i \cup U$.
Level-II:
Step.1 Now update the *GK* by invoking NM.Join protocol among all the CHs with the contribution of updated CK of $C_i \cup U$.
Step.2 Each of the CH distributes the updated key to all its members through their respective CK encrypted links.

Fig. 4 NM-CHH.Join protocol

Level-I : When a single node or a set U of nodes want to go away from the cluster $C_{i,}$ the CK CK_i is updated by invoking NM.Leave Protocol among $Ci\backslash U$ serially.
Level-II:
Step-1: Now update the GK by invoking NM.Leave protocol among all the CHs with the contribution of updated CK of Ci\U.
Step-2 : Each of the CH distributes the updated key to all its members through their encrypted links

Fig. 5 NM-CHH.Leave protocol

Security analysis

Here we presented the security of (i) unauthenticated protocol (UP): the initial key agreement (NM-CHH.IGKA). (ii) the authenticated key agreement (AKA): the NM-ACHH and (iii) the dynamic authenticated key agreement (DAKA): NM-DCHH (NM-CHH.Join and NM-CHH.Leave) of proposed protocols separately.

Theorem 4.2 addresses the security of unauthenticated static NM-CHH-IGKA and then the Theorem 4.3 deals with security of authenticated CHH protocol (NM-ACHH). Finally Theorem 4.4 states the security of dynamic authenticated CHH protocol (NM-DACHH).

Lemma 4.1 *The unauthenticated NM-GKA scheme depicted in "Background protocols" section is secure in opposition to passive opponent under ECDDH supposition, accomplishes forward secrecy and fulfils the accompanying:* $Adv_{NM}^{KA}(t,\ q_E) \leq 2Adv_{G}^{ECDDH}(t') + 2q_E/|G|$, *where* $t' = t + O(|\mathcal{P}|q_E t_{s.m})$, $t_{s.m}$ *is the time required to carry out scalar multiplications over* $G = E(F_p)$, $|\mathcal{P}|$ *is the amount of participants in the network and* q_E *is the amount of implemented queries that an opponent may ask.*

Proof The lemma's proof is depicted in [16] as a theorem. □

Theorem 4.2 *The unauthenticated static NM-CHH.IGKA protocol depicted in "Proposed protocol" section is secure against inactive opponent under ECDDH presumption, accomplishes forward secrecy and fulfils the accompanying:*

$$Adv_{NM-CHH}^{KA}(t,\ q_E) \leq Adv_{Symm}(t,\ 0,0) + \frac{1}{(r+1)q_E} Adv_{NM}^{ECDDH}(t') + \frac{1}{(r+1)(2q_E + |G|)},$$

where $t' = t + O(|\mathcal{P}_{max}|q_E t_{s.m})$, $t_{s.m}$ *is the time required to execute scalar multiplications over* $G = E(F_p)$, P_{max} = *maximum amount of users in a cluster of the network,* $r+1$ *is the amount of clusters formed in the network and* q_E *is the amount of implemented queries that an opponent may pose.*

Proof The verification regard as an opponent $\mathcal{A}$ who overcomes the security of proposed unauthenticated static NM-CHH scheme. Given $\mathcal{A}$, we build an enemy $\mathcal{B}$ assaulting the symmetric encryption plot (Symm); identifying with the achievement likelihood

of $\mathcal{A}$ and $\mathcal{B}$ gives the expressed consequence of the theorem. Before portraying $\mathcal{B}$, we initially characterize event Bad and bound its likelihood. Let Bad be the event to be the occasion that $\mathcal{A}$ can recognize a CK (which is a key concurred by the NM scheme) from a arbitrary value anytime amid its execution.

Let Prob [Bad] stands for $\text{Prob}_{NM-CHH}[Bad]$. Let Succ indicate the event that $\mathcal{A}$ succeed the game.

Notice that $r+1$ clusters are required in the network, in each execution of proposed protocol to form the GK:

i. The execution of NM-GKA protocol simultaneously for r clusters in level-I.
ii. The execution of the NM-GKA protocol among the r CHs in level-II.
iii. Symmetric encryption scheme: Symm for distributing the key among the clusters with respect to given CKs.

The opponent $\mathcal{A}$ performs $\boldsymbol{q_E}$ execute queries and accordingly carry out $\boldsymbol{r} \cdot \boldsymbol{q_E}$ executions of NM-GKA scheme in level-I and $1 \cdot \boldsymbol{q_E}$ executions of NM-GKA protocol in level-II respectively. Consequently performs a total of $(r+1)\boldsymbol{q_E}$.

$$\therefore Prob[Bad] \leq \frac{Prob[succ]}{(r+1)q_E}.$$

Now by definition,

$$Adv^{KA}_{NM,\mathcal{A}} = \left|2Prob[succ] - 1\right|$$
$$\Rightarrow Prob[succ] \leq \frac{1}{2}\left[1 + Adv^{KA}_{NM,A}\right].$$

Hence we have

$$Prob[Bad] \leq \frac{\frac{1}{2}\left[1 + Adv^{KA}_{NM,A}\right]}{(r+1)q_E} = \frac{Adv^{KA}_{NM,A}}{2(r+1)q_E} + \frac{1}{2(r+1)q_E}.$$

$\mathcal{B}$ simulates every oracle queries of $\mathcal{A}$ by implementing the unauthenticated static NM-CHH protocol all alone. Thusly, $\mathcal{B}$ can recognize the event of occasion Bad. $\mathcal{B}$ gives impeccable simulation to $\mathcal{A}$ so long as the occasion Bad does not happen. If at any point the event Bad happens, $\mathcal{B}$ prematurely ends and yield a random bit. Something else, $\mathcal{B}$ outputs whatever bit in the end yield by $\mathcal{A}$. So $Prob_{\mathcal{A},NM-CCH}[succ|Bad] = 1/2$.

Now,

$$
\begin{aligned}
Adv_{\mathcal{B},Symm} &= 2\left|Prob_{\mathcal{B},Symm}[Succ] - 1/2\right| \\
&= 2\left|Prob_{\mathcal{A},NM-CCH}\left[Succ \wedge \overline{Bad}\right] + Prob_{\mathcal{A},NM-CCH}[Succ \wedge Bad] - 1/2\right| \\
&= 2|Prob_{\mathcal{A},NM-CCH}[Succ \wedge \overline{Bad}] + Prob_{\mathcal{A},NM-CHH}[Succ|Bad]\ \left(Prob_{\mathcal{A},CHH}[Bad] - 1/2\right)| \\
&= 2\left|Prob_{\mathcal{A},NM-CHH}[Succ \wedge \overline{Bad}] + \left(\frac{1}{2}\right)Prob_{\mathcal{A},NM-CHH}[Bad] - 1/2\right| \\
&= 2\left|Prob_{\mathcal{A},NM-CHH}[Succ] - Prob_{\mathcal{A},NM-CHH}[Succ \wedge Bad] + \left(\frac{1}{2}\right)Prob_{\mathcal{A},NM-CHH}[Bad] - 1/2\right| \\
&\geq \left|2.Prob_{\mathcal{A},NM-CHH}[Succ] - 1\right| - \left|Prob_{\mathcal{A},NM-CCH}[Bad] - 2Prob_{\mathcal{A},NM-CHH}[Succ \wedge Bad]\right| \\
&\geq Adv_{\mathcal{A},NM-CHH} - Prob[Bad]
\end{aligned}
$$

Note that ever call upon its encrypting oracle E. Furthermore, the $\mathcal{B}$'s running time is at most t.

As $Adv_{B,Symm} \leq Adv_{Symm}(t,\ 0, 0)$, by assumption.

$$
\begin{aligned}
Adv^{KA}_{NM-CHH}(t,\ q_E) &\leq Adv_{Symm}(t, 0, 0) + Prob[Bad] \\
&\leq Adv_{Symm}(t,\ 0, 0) + \frac{Adv^{KA}_{NM,A}(t, (r+1)q_E)}{2(r+1)q_E} + \frac{1}{2(r+1)q_E} \\
&\leq Adv_{Symm}(t,\ 0, 0) + \frac{1}{2(r+1)q_E}\left[2Adv^{ECDDH}_{NM}(t') + \frac{2q_E}{|G|}\right] + \frac{1}{2(r+1)q_E} \\
&= Adv_{Symm}(t,\ 0, 0) + \frac{1}{(r+1)q_E}Adv^{ECDDH}_{NM}(t') + \frac{1}{(r+1)(2q_E + |G|)},
\end{aligned}
$$

when $t' = t + O(|P_m|q_E t_{sm}) = t + O((r+1)|q_E t_{sm})$, where $|P_m|$= maximum amount of clusters in the network $= r+1$

Hence by Lemma 4.1, we realize the theorem. □

We now present the security of the NM-ACHH in which the security is depends on that of unauthenticated schemes relied on fact that DSig (signature scheme) is secure.

Theorem 4.3 *The authenticated CHH scheme (NM-ACHH) is secure in opposition to active opponent under Elliptic Curve-Decision Diffie Hellman (EC-DDH) supposition, accomplishes forward secrecy and outputs the following:*

$$
Adv^{AKA}_{NM-ACHH}(t,\ q_E,\ q_S) \leq Adv^{KA}_{NM-CHH}\left(t',\ q_E + \frac{q_S}{2}\right) + |\mathcal{P}|Adv_{Dsig},
$$

where $t' = t + (|\mathcal{P}|q_E + q_S)t_{ACHH}$, *with* t_{ACHH} *is the time needed for carrying out of NM-ACHH by each of the party,* q_S *and* q_E *are respectively the maximum amount of Send and Execute query an opponent may pose.*

Proof Let $\mathcal{A}'$ be a opponent ambushing the AP. With this we construct an enemy $\mathcal{A}$ attacking the UP.

We initially confine the likelihood of the event Forge that $\mathcal{A}'$ outputs an authentic forge w.r.t publickey pk_i for some client $M_i \in \mathcal{P}$ before making the question corrupt (M_i).

Claim *Let Forge be the incident that a signature of Dsig is forged by* $\mathcal{A}'$ *then*

$$Prob[Forge] \leq |\mathcal{P}| Adv_{Dsig}(t'). \tag{1}$$

Proof $\mathcal{A}'$ prepares a signature forger $\mathcal{F}$ to challenge *Dsig*-scheme. The aim of $\mathcal{F}$ preparation is that, when a publickey *PK* is given as input, $\mathcal{F}$ has permission to a signing oracle using PK, which generates a legitimate forgery (m, σ), i.e., $\gamma_{PK}(m, \sigma) = 1 \ni \sigma$ was not previously output by the signing oracle as a signature over m. The $\mathcal{F}$ chooses a client $M_f \in$ at random, and sets PK_f *to the PK*. For left over members, $\mathcal{F}$ legitimately generates key pair (private key, public key) by executing GKA protocol. In addition, $\mathcal{F}$ carryout the method, necessary for Initiating UP. At this moment $\mathcal{F}$ carryout $\mathcal{A}'$ as a subprogram $\in$ simulated queries from $\mathcal{A}'$ are as below:

- Execute (M)/Reveal (π_i^s)/Dump $((\pi_i^s))$/Test $((\pi_i^s))$: these questions are answered in an obvious manner.
- Send $((\pi_i^s), m)$: every private keys of M_i are aware to $\mathcal{F}$ when i $\neq f$, then, respond to queries subsequent to the particular protocol specifically. Conversely if $i = f$, then every M_i''s signing keys are unrecognized by $\mathcal{F}$ Incidentally, $\mathcal{F}$ can acquire message signature it needs by accomplishment to signing oracle related to *PK*.
- Corrupt (M_i). If i $\neq f$, *F* principally holds M_i''s private keys stands for long period, created itself. On the other hand if $\mathcal{A}'$ corrupts $M_i = M_f$, then, $\mathcal{F}$ terminates and returns "fail".

The displayed above simulation is marvelously ill defined from the authentic execution except if enemy $\mathcal{A}'$ represents the query corrupt (M_f). All the way through this simulation, $\mathcal{F}$ glances each send question from $\mathcal{A}'$, and keeps an eye in the unlikely event that it fuses an authentic pair (m, σ) using PK. If no such inquiry is posed till $\mathcal{A}'$ ends, at that point $\mathcal{F}$ closures and returns "fail". Else, $\mathcal{F}$ generates (m, σ) as real fraud w.r.t PK. Lemma 3 straight forwardly inferred from the manner in which the second case occurs with likelihood pγ[Forge]/n.

Currently we portray the improvement of attacking UP, that utilizes $\mathcal{A}'$ ambushing AP. $\mathcal{A}$ uses tlist and keep (session Ids, transcripts) in it. $\mathcal{A}$ makes (verification keys (pk_M), signing keys (sk_M)) for each customer $M \in P$ and check keys are given to $\mathcal{A}'$. At the point when the event Forge occurs, $\mathcal{A}$ rashly closures and outputs an arbitrary bit. Else, outputs a similar bit whatever $\mathcal{A}'$ outputs. $\mathcal{A}$ can recognize occasion of the event Forge $\mathcal{A}'$ in light of the fact that it knows sk_M and pk_M. The oracle questions of $\mathcal{A}'$ are imitated by $\mathcal{A}$ using its inquiries to the Execute Oracle (EO). The motto is to procure a transcript (T) of *UP* for every single Execute question of $\mathcal{A}'$. Besides for every one beginning send question, send_0 (M, I, *) of $\mathcal{A}'$. $\mathcal{A}$ then fixes legitimate sign with messages in T to secure a transcript (T') of AP and uses T' to answer request of $\mathcal{A}'$. since

by assumption, $\mathcal{A}'$ can't forge, $\mathcal{A}'$ is "compelled" to send out messages viably contained in T'. This system gives a decent simulation. The details are underneath:

Execute queries (EQs'): presume $\mathcal{A}'$ asks EQ ((M_{i1},d_1),...,(M_{ik}, d_k)) and so that occasions $\pi_{M_1}^{i_1} \ldots \pi_{M_k}^{i_k}$ are incorporated.

$\mathcal{A}$ characterizes $S = \{(M_{i_1}, d_1), \ldots, (M_{i_k}, d_k)\}$ and send out the EQ to its EO. It outputs a T by implementing *UP*. It attaches (s, t) to tlist and after that broadens T for the *UP* into T' for the *AP*. It offers T' to $\mathcal{A}'$.

Send queries (SQs'): the prime SQ means, $\mathcal{A}'$ asks an occasion to commence one more session, indicated by send_0. The opponent desires to use SQs' to commence a session between occasions $\pi_{M_1}^{i_1} \ldots \pi_{M_k}^{i_k}$ which are not yet used:

$$Send_0 = \left(M_{i_j}, d_j, \langle M_{i_1} \ldots M_{i_k} \rangle - M_{i_j}\right), \quad 1 \le J \le k.$$

These queries should not in an explicit order. $\mathcal{A}$ forms $S = \{(M_{i_1}, d_1), \ldots, (M_{i_k}, d_k)\}$ when these queries are prepared and sends an EQ to it's executing oracle. It outputs T and includes (S, T) to tlist.

Assume that signatures can't be forged, any progressive SQ to an event π_M^i is a really sorted out messages with a real signature. For each such SQ, $\mathcal{A}$ checks the question as depicted in the authenticated NM-CHH-GKA protocol. In the event that the confirmation overruled, $\mathcal{A}$ sets $acc_M^i = 0$, $sK_M^i = NULL$ and ends π_M^i. Else, $\mathcal{A}$ plays out the action to be completed by π in the AP. It finishes as under:

> Finds an sole entry (S,T) in tlist ϶ (M, i) ∈ S, such a novel entry exits for every one event by assumption. Presently from T, $\mathcal{A}$ finds best possible messages which is identified with the message transmitted by $\mathcal{A}'$ to π_M^i. From T, $\mathcal{A}$ gets following open information yielded by π_M^i and offers to $\mathcal{A}$.

Reveal/test queries (R Q/T Q): Suppose $\mathcal{A}'$ asks the RQ (M, i) or TQ (M, i) to an incident π_M^i for which $\text{acc}_M^i = 1$.. Currently the T' in which π_M^i took part has been predefined. Now first finds an sole entry (S,T) in the tlist ϶ (M, i) ∈ S. Imagine that, forge doesn't occur, T is sole unauthenticated transcript which is related to T'. Now asks proper RQ or TQ to any occasion incorporated in T and hand over a proportional payback to $\mathcal{A}'$ is just right. When Forge occurs, opponent $\mathcal{A}$ terminates and outputs an arbitrary bit.

$$Prob_{A',AP}[Succ|Forge] = 1/2.$$

$$\begin{aligned}
Now,\ Adv_{A,UP} &= 2\left|Prob_{A,UP}[Succ] - 1/2\right| \\
&= 2\left|Prob_{A',AP}[Succ \wedge Fo\bar{r}ge] + Prob_{A',AP}[Succ \wedge Forge] - 1/2\right| \\
&= 2\left|Prob_{A',AP}[Succ \wedge Fo\bar{r}ge] + Prob_{A',AP}[Succ|Forge]Prob_{A',AP}[Forge] - 1/2\right| \\
&= 2\left|Prob_{A',AP}[Succ \wedge Fo\bar{r}ge] + 1/2Prob_{A',AP}[Forge] - 1/2\right| \\
&= 2\left|Prob_{A',AP}[Succ] - Prob_{A',AP}[Succ \wedge Forge] + 1/2Prob_{A',AP}[Forge] - 1/2\right| \\
&\ge 2\left|Prob_{A',AP}[Succ] - 1| - |Prob_{A',AP}[Forge] - 2Prob_{A',AP}[Succ \wedge Forge]\right| \\
&\ge Adv_{A,AP} - Prob[Forge].
\end{aligned}$$

$$Adv_{NM-ACHH}^{AKA} \le Adv_{NM-CHH}^{KA}\left(t',\ q_s + q_e/2\right) + prob[Forge] \tag{2}$$

$\mathcal{A}$ asks an EQ in line with each EQ of $\mathcal{A}'$. Similarly poses an EQ in all sessions underway by $\mathcal{A}'$. Because, session consist of at least two instances, such as EQ is processed

after at least two SQs' of $\mathcal{A}'$. The max. no of such queries are $q_s/2$, where q_s is amount of queries posed by $\mathcal{A}'$. The maximum amount of EQs executed by $\mathcal{A}$ is $q_e + q_s/2$, where q_e is the amount of EQs' executed by $\mathcal{A}'$.

Already we have $Adv^{AKA}_{NM-ACHH}(t,\ q_E, q_S) \leq Adv^{KA}_{NM-CHH}\left(t',\ q_E + \frac{q_S}{2}\right)$ by supposition, from 1 and 2 we get,

$$Adv^{AKA}_{NM-ACHH}(t, q_E, q_S) \leq Adv^{KA}_{NM-CHH}\left(t', q_E + \frac{q_S}{2}\right) + |\mathcal{P}|Adv_{Dsig}$$

The statement of the theorem is yielded. □

We currently present the security of dynamic authenticated protocol (DAP): (NM-DACHH). Expecting that, DSig is secure, we can change over any enemy assaulting convention DAP into a opponent assaulting convention UP. We disregard Corrupt queries since our convention DAP does not utilize any long-time secret keys. Along these lines convention DAP obviously accomplishes forward secrecy.

Theorem 4.4 *The dynamic authenticated CHH scheme (NM-DACHH) depicted in "Proposed protocol" section fulfils the following:*

$$Adv^{AKA}_{NM-DACHH}(t, q_E, q_J, q_L, q_S) \leq Adv^{KA}_{NM-CHH}\left(t', q_E + \frac{(q_J + q_l + q_S)}{2}\right) + |\mathcal{P}|Adv_{Dsig}(t'),$$

where $t' = \mathrm{t} + (|\mathcal{P}|q_E + q_J + q_l + q_S)t_{DACHH}$, *with* t_{AHP} *is the time needed for carrying out of DACHH by each of the party* q_E, $q_S q_J, q_L$ *are in that order the maximum amount of Execute, Send, Join and Leave queries an opponent may pose.*

Proof Let $\mathcal{A}'$ be an opponent who tries to attack DAP. By means of this we build an opponent $\mathcal{A}$ who assaults UP. As in the preceding proof, we had the following claim.

Claim *Let Forge be the incident, that* $\mathcal{A}'$ *forged the signature, then*

$$\text{Prob[Forge]} \leq |\mathcal{P}|\ \text{AdvDSig}\ (t').$$

At the moment we present the creation of the passive opponent $\mathcal{A}$ assaulting UP that utilizes opponent $\mathcal{A}'$ assaulting DAP. Opponent $\mathcal{A}$ can implement the UP numerous times, among every subset of $\mathcal{P}$ and can acquire session key of scheme implementation by producing a RQ to any occurrence concerned in session. Now we demonstrate that $\mathcal{A}$ simulates itself Leave and Join questions of $\mathcal{A}'$ utilizing its own Reveal Oracles (ROs) and EOs. Opponent $\mathcal{A}'$ keeps up a Tlist to store sets of session IDs and transcripts. It likewise utilizes two records Llist and Jlist to be determined in future.

Opponent $\mathcal{A}$ creates signing/confirmation key pair (pkU, skU) for every client $\mathrm{U} \in \mathcal{P}$ and gives confirmation keys to $\mathcal{A}'$. If at any time the occasion Forge happens, opponent $\mathcal{A}$ prematurely ends and outputs an arbitrary bit. Else, $\mathcal{A}$ outputs no matter what bit is in the long run yield by $\mathcal{A}'$. Since the signing and confirmation keys, it can identify event of occasion Forge. $\mathcal{A}$ reproduces the oracle inquiries of $\mathcal{A}'$ utilizing its own questions to the ROs and EOs. We present particulars below.

EQs': these queries are replicated in Theorem 4.2 proof.

SQs': separately from regular SQ, two special send queries, Send_{L} and Send_{J} are there.

Let, set $S_1=\{(M_{ik+1}, d_{k+1}),\ldots,(M_{ik+l}, d_{k+l})\}$ of occurrences, needs to join gathering $S=\{(M_{i1}, d_1),\ldots,(M_{ik}, d_k)\}$, at that point $\mathcal{A}'$ will create Send_{J} $(M_{ij}, d_j, \langle M_{i1},\ldots, M_{ik}\rangle)$ query for every j, $k+1\leq j\leq k+l$. These queries commence Join (S, S_1) query. The occurrence in S might have previously implemented either (a) UP or (b) leave protocol or (c) join protocol. As a result, first $\mathcal{A}$ finds any of the subsequent form of a sole entry: (1) (S, T) in Tlist or (2) (S', S'', T) in Jlist with $\text{S}=S'\cup S''$ or (3) (S', S'', T) in Llist with $\text{S}=S'\setminus S''$. If no such entry, makes an EQ to its personal EO on S, obtains a transcript T and keeps (S, T) in Tlist.

Whenever $(S, T)\in$ Tlist, $\mathcal{A}$ fundamentally issues RQ to an event in S so as to accomplish the session key sk identified with T, calculates seed $x=H(sk)$ and plan the calculation for Join by questioning its EO (rolling out fitting improvements). At that point include signature in every message, acquires T' and stores $(S, S1, T')$ in *Jlist*. In this manner reproduces the transcript T' of Join utilizing self RO and EO. In the rest of the cases (2) and (3), produces T by and by thus $\mathcal{A}$ can simulate T' of Join from T.

Likewise, when an unused instances of $S_2=\{(M_{l1}, d_{l1}),\ldots,(M_{lm}, d_{lm})\}$ desires to leave $S=\{(M_{i1}, d_1),\ldots,(M_{ik}, d_k)\}$, then, $\mathcal{A}'$ will Send_{L} $(M_{ij}, d_j,(M_{i1},\ldots, M_{ik}))$ inquiry for every j, $\text{j}\in\{l_1,\ldots, l_m\}$. These inquires commences Leave(S, S_2) query. As stated in join member, first traces an entry (S, T) in Tlist or an entry (S', S'', T) in Jlist with $\text{S}=S'\ U\ S''$ or an entry (S', S'', T) in Llist with $\text{S}=S'\setminus S''$. If entry is missing, then $\mathcal{A}$ set up an inquiry to its personal EO on S, obtain T and adds (S, T) to Tlist.

$\mathcal{A}$ simulates protocol for Leave without anyone else's input and gets an altered T' from T as pursues: $\mathcal{A}$ distinguishes the situations in T where the new messages are to be infused or the old messages are to be supplanted by new. $\mathcal{A}$ do these alterations in T as indicated by protocol for leave depicted in Fig. 5 and gets an adjusted T' by fixing up fitting signature with each message. In this way $\mathcal{A}$ extends T into a T' for Leave protocol. $\mathcal{A}$ stores $(S, S2, T')$ in Llist.

Send_0 questions are replied as in Theorem 4.3. The typical send questions are prepared as in Theorem 4.3 with the accompanying changes.

Assume $\mathcal{A}'$ formulates a SQ to occurrence $\prod_M^i$. After appropriate check, discovers an entry $(S, T)\in$ Tlist, such that $(M, i)\in S$. The response to this inquiry is as in Theorem 4.3. If no such entry is found, then discovers a sole entry (S, S_1, T') in Jlist such that $(M, i)\in S_1$.

This implies the session for Join has just been started. At that point acquires the next public information for T' to be yield by $\prod_M^i$ (given all essential data has been achieved by Π_M^i by SQs from $\mathcal{A}'$) and forwards it to $\mathcal{A}'$. If discovers an sole entry (S, S_2, T') in Llist such that $(M, i)\in \text{S}_2$, then as above, the proper response to the question is found from T'.

Join queries (JQs): assume $\mathcal{A}'$ sends a JQ (S, S_1) where $S=\{(M_{i1}, d_1),\ldots,(M_{ik}, d_k)\}$ and $S=\{(M_{ik+1}, d_{k+1}),\ldots,(M_{ik+l}, d_{k+l})$. The occurrences $\Pi_{M_{i_{k+1}}}^{d_{k+1}},\ldots,\Pi_{M_{i_{k+l}}}^{d_{k+l}}$ desire to join the group $\Pi_{M_{i_1}}^{d_1},\ldots,\Pi_{M_{i_k}}^{d_k}$. $\mathcal{A}$ discovers an entry of the form (S, S_1, T') in Jlist. If no such entry, then the opponent $\mathcal{A}'$ doesn't give any output. Else, $\mathcal{A}$ returns T' to $\mathcal{A}'$

Leave queries (LQs): Assume $\mathcal{A}'$ sends a $LQ(S, S_2)$ where $S=\{(M_{i1}, d_1),\ldots,(M_{ik}, d_k)\}$ and $S_2=\{(M_{l1}, d_{l1}),\ldots,(M_{lm}, d_{lm})\}$ where $(M_{lj}, d_{lj})\in S$ for $1\leq j\leq m$. The occurrences $\Pi_{M_{l_1}}^{d_{l1}},\ldots,\Pi_{M_{l_m}}^{d_{lm}}$ desires to leave the group $\Pi_{M_{i_1}}^{d_1},\ldots,\Pi_{M_{i_k}}^{d_{i_k}}$ where $M_{i_j}\in\{M_{i_1},\ldots,M_{i_k}\}$ for

$1 \leq j \leq m$. $\mathcal{A}'$ discovered an entry of the form (S, S_2, T') in Llist. If no such entry, then the opponent $\mathcal{A}'$ is doesn't give any output. Else, $\mathcal{A}$ returns T' to $\mathcal{A}'$.

Reveal/Test (R/T) queries: assume $\mathcal{A}'$ sends the RQ(M, i) or TQ(M, i) for an occurrence Π_M^i for which $acc_M^i = 1$.. At this moment the transcript T' in which Π_M^i take part has been predefined. If T' related to the transcript of the AP then $\mathcal{A}'$ discovers the sole pair (S, T) in Tlist such that $(M, i) \in S$. Supposing that the occasion Forge does not occur, T is the sole unauthenticated transcript which relates to the transcript T'. Then sends the suitable RQ or TQ to one of the occasions concerned in T and returns the result to $\mathcal{A}'$. Else, T' is the transcript for Join or Leave, as the case may be. Because T' has been simulated by $\mathcal{A}$, is capable to calculate the updated session key and hence send an appropriate reply to $\mathcal{A}'$.

Providing Forge doesn't occur, the above simulation for $\mathcal{A}'$ is perfect. At the time Forge occurs, opponent $\mathcal{A}$ terminates and outputs a arbitrary bit.

So $Prob_{\mathcal{A}',AP}[Succ|Forge] = \frac{1}{2}$. By means of this, one can prove

$$Adv_{,UP} \geq Adv_{\mathcal{A}',DAP} - Prob[Forge]$$

The opponent $\mathcal{A}$ sends an EQ for every EQ of $\mathcal{A}'$. $\mathcal{A}'$ poses q_J, JQs and q_L, LQs. These inquiries are commenced respectively by $Send_J$ and $Send_L$ inquires of $\mathcal{A}'$. Currently every $Send_J$ and $Send_L$ inquiry of $\mathcal{A}'$ poses at most one EQ of. Consequently there are at most $q_J + q_L$ EQs posed by $\mathcal{A}$ to reply all the $Send_J$ and $Send_L$ inquiries of $\mathcal{A}'$. Also $\mathcal{A}$ poses an EQ for every session commenced by $\mathcal{A}'$ by means of SQs. Because a session engages at least two occurrences, such an EQ is prepared after at least two SQs of $\mathcal{A}'$. Consequently there are $(q_S - q_J - q_L)/2$ EQs of $\mathcal{A}$ to react to all other SQs of $\mathcal{A}'$, where q_S is the amount of SQs prepared by $\mathcal{A}'$. Consequently the total amount of EQs posed by is at most $q_E + q_J + q_L + (q_S - q_J - q_L)/2 = q_E + (q_J + q_L + q_S)/2$, where q_E is the amount of EQs posed by $\mathcal{A}'$. Furthermore since $Adv_{A,UP}(t', q_E, q_J, q_L, q_S) \leq Adv_{UP}^{KA}(t', q_E + q_{J/2} + q_{L/2} + q_{S/2})$ by assumption, we obtain:

$$Adv_{DAP}^{AKA}(t, q_E, q_J, q_L, q_S) \leq Adv_{UP}^{KA}(t', q_E + (q_J + q_L + q_S)/2) + Prob[Forge].$$

$$Adv_{NM-DACHH}^{AKA} \leq Adv_{NM-CHH}^{KA}(t', q_E + (q_J + q_L + q_S)/2) + |\mathcal{P}|Adv_{DSig}(t')$$

This implies the statement of the theorem. □

Comparative analysis

Here the proposed ECDH-based NM-clustering-based hybrid hierarchical group key agreement (NM-CHH-GKA) protocol has been compared with prevalent clustering based GKA protocols such as HKAP [25], GKA-CH [21], PB-GKA-HGM [31], AP-1/AP-2 [33], ACEKA [26], A-DTGKA [20], ACBGKA [18], ECDH-SKDM [43] and NM-setup [16] with regard to various characteristics such as pre required GKA protocol used, structure and limitations are in Table 2. Further we compare the proposed one with them in terms of communication and computational complexities in Table 3.

Here Let the amount of nodes be "n" and choose $l = \lceil \sqrt{n} \rceil$ be the amount of clusters members such that $l \ll r$ and the amount of clusters $r = \lceil n/l \rceil$

Table 2 Summary of the key characteristics of cluster

	Authentication	Protocol used	Structure	Limitations and specific characteristics of the protocol
HKAP [25]	No	Some GKA	Users are gathered into clusters and shape a various levelled structure. The member chosen as CHs ought to be more effective devise and can transmit at higher level links	Grants the geographic revamping of members without setting off a key-revive each time a user be in motion starting with one cluster then onto the next
GKA-CH [21]	No	BD convention [12] in each layer of a roundabout various levelled group structure	The group is prearranged in "h" hierarchical layers encompassed by one/more clusters organized in a circle	All clusters have the same size
PB-GKA-HGM [31]	Yes	A password-based convention is implemented in every cluster	Constructs a level wise tree structure using 3 entities: the chief controller C in peak layer, different CHs S_i and several members (CM) in each cluster	Each CM ought to have a password and a pair wise secret key imparted to the CH. Each CH ought to hold a password and a pair wise secret key imparted to the GC. Passwords and secret keys are preloaded into hubs
AP-1 [33]	Yes	A variant of the BD scheme called DB [7] is used for CKs as well as GK	A CH is chosen from each of the cluster to take part in the creation of the complete GK	Communication between CHs should be one hop
AP-2 [33]	Yes	The CK are established using DB protocol and for GKDBS [37] is used	CHs are prearranged in a tree structure, which facilitates proficient management of dynamic membership alterations	The GKA implemented in AP-2 is pairing-based, which raises the cost of computation and the entire complexity of the scheme
ACEKA [26]	Yes	D-H and Joux's tripartite key agreement [35] for clusters of 2 and 3 nodes respectively	Members are clustered into sub-groups of size 2 or 3. In numerous CK trees, the root node stand for the CK. A spine key merge each and every CK trees. Virtual nodes are utilized to build the tree structure	Hubs are assembled into sub gatherings of size 2 or 3. A few GK trees. The root hub speaks to the CK. A spine key consolidates all the CK trees. Virtual hubs are utilized to build the tree structure
CDGH [30]	No	GDH.2 [36] GKA protocol is used to generate the keys	Members are clustered into identical sized perfectly balanced hierarchical structure	The members of the group don't have to store intermediate keys to create the session key
A-DTGKA [20]	Yes	The protocol is ID-based and employs pair wise keys for entity authentication	Hierarchical tree structure of clusters consisting of neighbour members	Member characteristics should be publicly accessible TA is required in the setup phase to produce the private keys for each node
ACBGKA [18]	Yes	Utilizes the Joux's [35] tripartite key assertion convention	Each of the clusters contains either two or three nodes	A KGC is essential for the creation of the private/public keys for every member
C-AT-DH [28]	No	The BD and T-GDH protocol are used for the generation of the cluster and group key respectively	Users are gathered into clusters named cliques	Clusters must have a explicit structure i.e., each CM can commune in one-hop with all other CMs
ECDH-SKDM [43]	No	CLIQUES key agreement	Hierarchical tree structure	The maximum amount of sub group members should be limited
Proposed-protocol NM-CHH-GKA	YES	NM-GKA [16] is used for CKs as well as GK	Clustering-based, Hybrid, Hierarchical (CHH) structure	The maximum amount of cluster members and number of clusters should be limited (scalable) in case of ad hoc networks

Table 3 Computation and communication complexities

Protocol	Total number of messages required	Total number of sequential scalar multiplications	The number of pairings
HKAP [25]	$5(l+r)-6$	$5(l+r)-12$	–
GKA-CH [21]	$2l+5r+8$	$6(l+1)$	–
PB-GKA HGM [31]	$2r+3l+1$	$2(l+r+5)$	–
AP-1/AP-2 [33]	$2l+4r$	$(3l+6r)$	–
ACEKA [26]	$3(n/2)+3$	$(5(n/2)+18)$	$21(n/2+1)$
A-DTGKA [20]	$(13n-7)/3$	$(5n+1)$	$(5n+1)$
ACBGKA [18]	$4n$	$(5n)$	$11/2\ (n)$
ECDH-SKDM [43]	$r+1$	n	–
NM-setup [16]	$2(l\cdot r-2)$	$2(l\cdot r)$	–
OUR-protocol (NM-CHH-GKA)	$2(l+r-2)$	$2(l+r)$	–

Table 4 Computation and communication energy costs

Computation cost	
Scalar multiplication	8.8 mJ
Pairings computation cost	47.0 mJ
Communication cost	
Transmitting a message	3.46 mJ
Receiving a message	2.40 mJ

From Table 3, it follows that the proposed protocol is optimal with reference to communication and computation expenses, facilitating the equal level of security with fewer key sizes. Further the proposed protocol is shown to be optimal for secure GKA over resource constrained networks like WSN and Mobile Ad hoc Networks (MANETS) and among ECDLP/DLP-based protocols confer in this paper.

With the end goal to acquire a improved guess for the energy cost of computation and communication for the scheme presented in this paper, we ascertained its energy utilization for a particular sensor. Particularly, we pick a sensor network involved by Tmote Sky gadgets by Texas Instruments with a most extreme 100 kbps data rate. As per [47] a sensor hub relied on the 133 MHz Strong ARM chip devours 8.8 mJ for a scalar multiplication and 47.0 mJ for a paring. Concerning the cost of communication, a 100 kbps radio handset module devours 10.8 μJ and 7.51 μJ for the communication gathering of one bit of information in that order.

For GKA scheme we utilize its EC-analog and in this manner suppose that the traded messages has the size of an EC-point. In the event that we utilize a 160-bit EC, the extent of its points (x, y) will be 320 bits. We would then be able to figure the expense for the reception and transmission by multiplying energy cost with its size in bits for the reception and transmission of a single bit. Table 4 outlines a scalar multiplication's energy costs, a pairing calculation and a reception and transmission of a message utilizing the specific gadget (Tmote Sky) and radio handset module of speed the 100 kbps.

From Table 3 the total amount of Sequential Scalar Multiplications and Messages if we use NM-GKA [16] protocol among all the nodes in the system are $2(l\cdot r), 2(l\cdot r-2)$ which may *not* be feasible for large WANETs. Consequently, we plan to use the same for

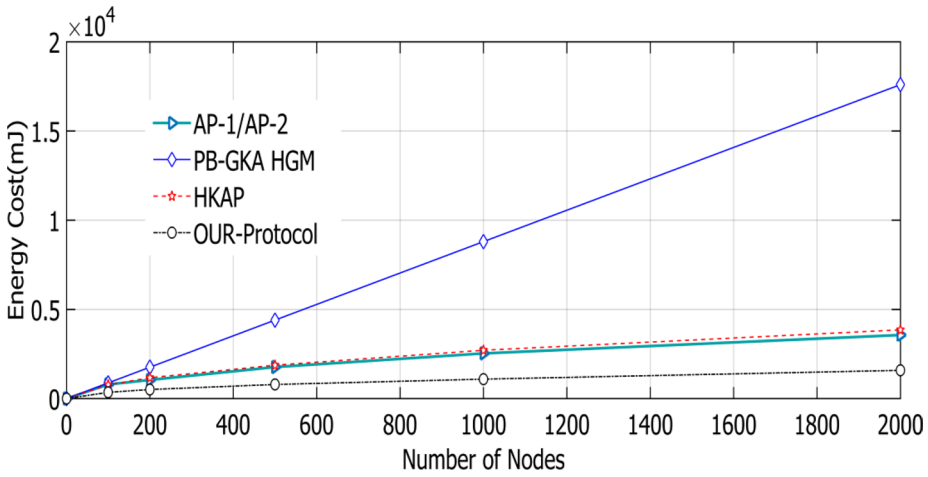

Fig. 6 Comparison of energy costs

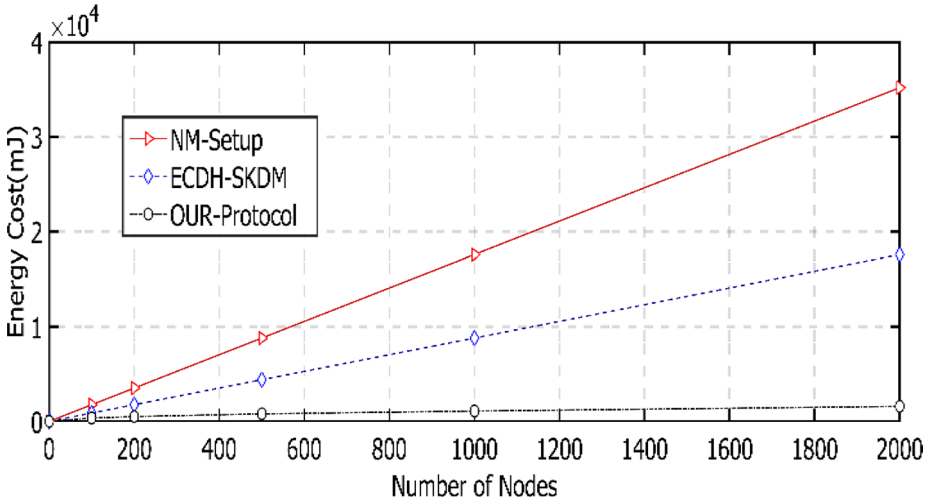

Fig. 7 Comparison of energy costs

each of the "r" cluster of "l" nodes each in *parallel* in level-I and then for all the "r" CHs in level-II hierarchically to establish the GK so the proposed protocol uses total amount of Sequential Scalar Multiplications and Messages $2(l + r)$, $2(l + r - 2)$ only.

Computational complexities using graphs

Figures 6, 7 and 8 indicates comparison on computational energy cost of proposed NM-CHH-GKA protocol with reference to number of nodes for establishing GK and shown that the proposed one is the optimal when compared to the other protocols. So the proposed NM-CHH-GKA works with lower computational cost and better efficiency when compared to existing protocols. So It is suitable for recourse constrained networks such as WANETS.

Communication complexities using graphs

Figures 9, 10 and 11 indicates comparison on communication energy cost of proposed NM-CHH-GKA protocol with reference to number of nodes for establishing GK and

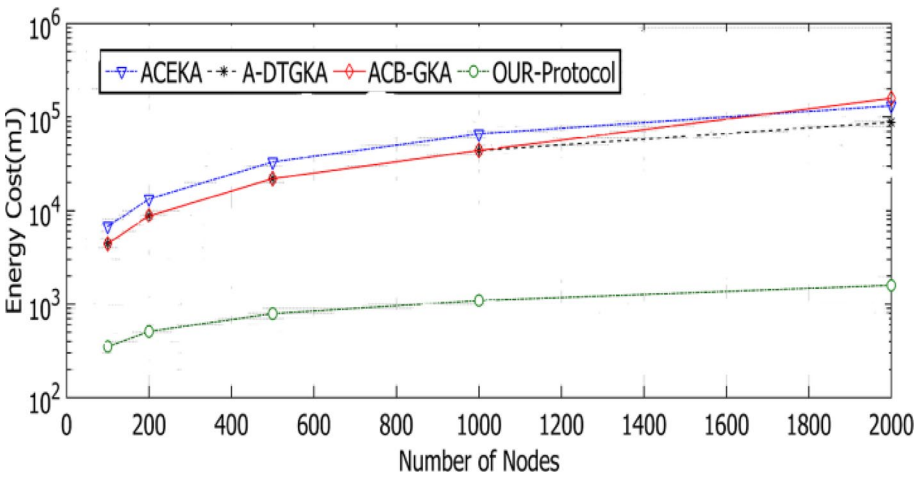

Fig. 8 Comparison of energy costs

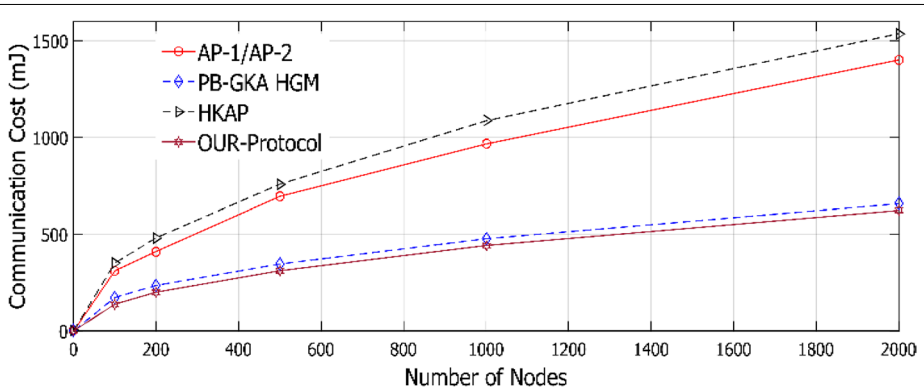

Fig. 9 Communication cost comparisons

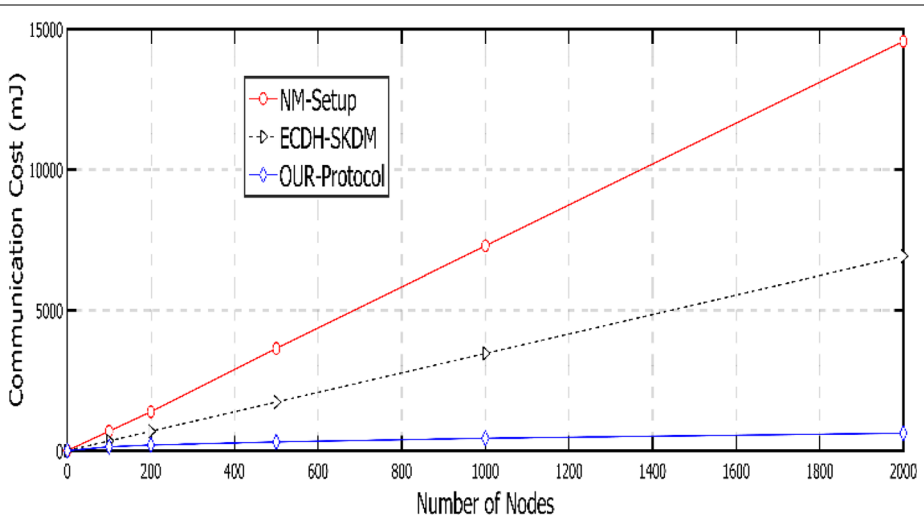

Fig. 10 Communication cost comparisons

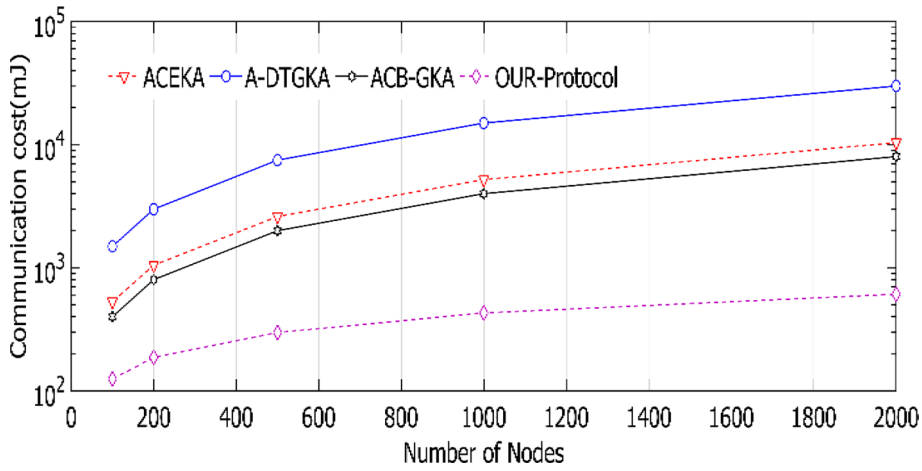

Fig. 11 Communication cost compariso

shown that the proposed one is the optimal when compared to the other protocols. So the proposed NM-CHH-GKA works with relatively low communication overheads and greater competence when compared to existing protocols. So It is fitting for recourse embarrassed networks such as WANETS.

Experimental results

For *Experimentation* Linux environment was used running on a system with configuration 2.4 GHz Celeron(R) CPU with 512 MB of memory. A NS-2 simulator was used to establish a hierarchical arrangement of nodes in tree topology format. A Crypt++ Library 5.2.1 was utilized to implement NM-CHH-GKA scheme, different libraries were used to develop algorithms for the key sharing, encryption and decryption algorithm. NS-2 libraries were used to establish the TCP connection and communication among the nodes to share the packets (max 1000 bytes), to support multicasting or unicasting in the derivation of key as well as data sharing.

For each examination, we ran the protocol for 10 times and calculate the average computation times for different operations such as level-I group formulation, level-II group formulation, Computation of $K_{i,j}$ values, Computation of L_i values, Computation of individual CKs SKi/CK_i computation, and GK with the following tabulated NS-2 parameters in Table 5.

Experimental results for computational times

Let the quantity of members be "n" and choose $l = \lceil \sqrt{n} \rceil$ number of cluster members such that $l < r$ and r $= \lceil n/l \rceil$. We presented the experimental results for computational time with respect to amount of nodes, quantity of clusters; quantity of members in a cluster are tabulated in detail in Table 6. Further we present the experimental comparative analysis between NM-Setup and NM-CHH Setup in Table 7.

Table 5 Experimental NS-2 parameters

Simulation parameter	Value
Simulator	NS-2
Traffic type	FTP over TCP
Bandwidth	2 Mb
Router queuing	DropTail
Link noise	None
Communication mechanism	Multicast/unicast
Topology	Tree topology
Packet size	1000 bytes
Packet rate	1 Mb

Table 6 Experimental Computation times for various group sizes

Members (n)	Cluster members ($l = \lceil \sqrt{n} \rceil$)	Cluster size ($r = \lceil n/l \rceil$)	Setup time (ms)		Level-1 cluster key computation time (ms)		Level-2 computation time (ms)		Total group key generation time excluding setup time (ms)
			Level-I	Level-II	Members	Cluster heads	Cluster heads as members	Group head	
100	10	10	28	30	76	42	74	41	233
200	15	14	45	45	90	300	94	156	640
500	25	20	84	60	104	308	102	217	641
1000	32	32	104	104	107	260	107	260	734
2000	45	45	156	156	120	295	120	295	830

Table 7 Experimental comparative analysis

Group size	Group key generation time in ms		Number of times
	NM-setup	NM-CHH-setup	NM-CHH-setup is faster than NM-setup
100	1587	233	6
200	9988	640	15
500	96,522	641	150
1000	676,197	734	921
2000	4,733,379	830	5702

The experimental results through graphs

Various scenarios of experimental results of NM-CHH-GKA scheme are presented in Figs. 12, 13, 14 and 15. Further we presented comparison of computation time between NM and NM-CHH in Fig. 16.

Figure 12 indicates comparison between setup time for GKA in level-I and level-II. We can observe that setup time in both levels NM-CHH-GKA are mostly same because we are using same NM.Setup in both levels.

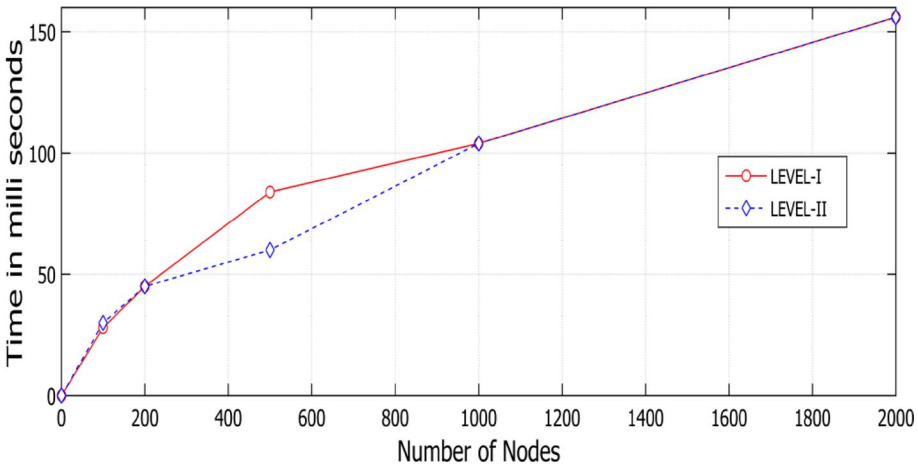

Fig. 12 Setup time: level-I vs level-II

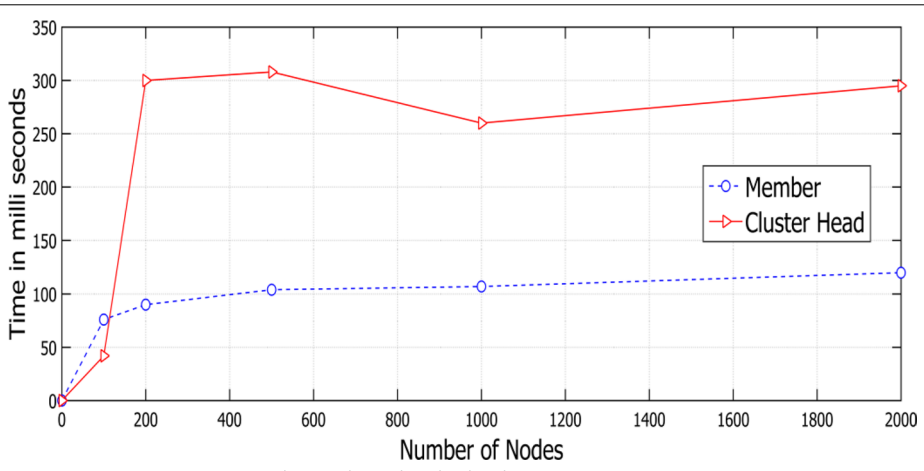

Fig. 13 Computation time: member vs cluster head in level-I

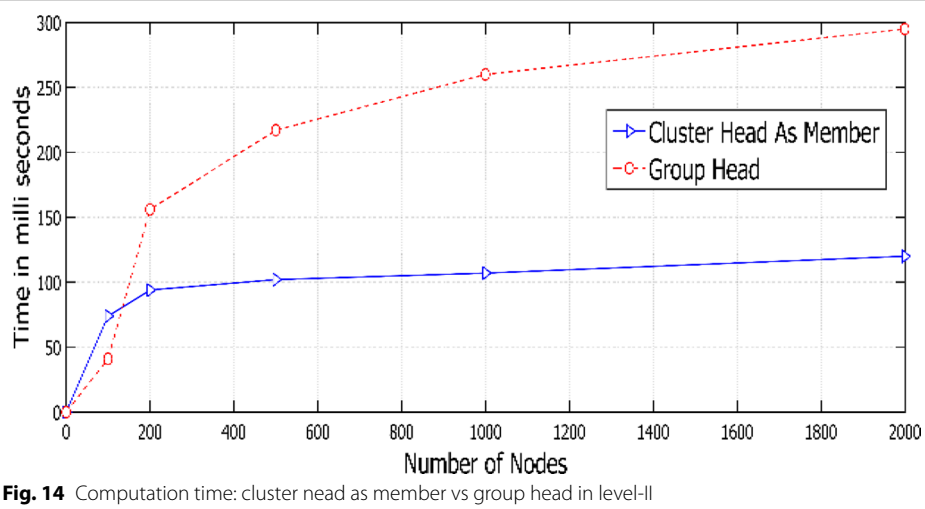

Fig. 14 Computation time: cluster nead as member vs group head in level-II

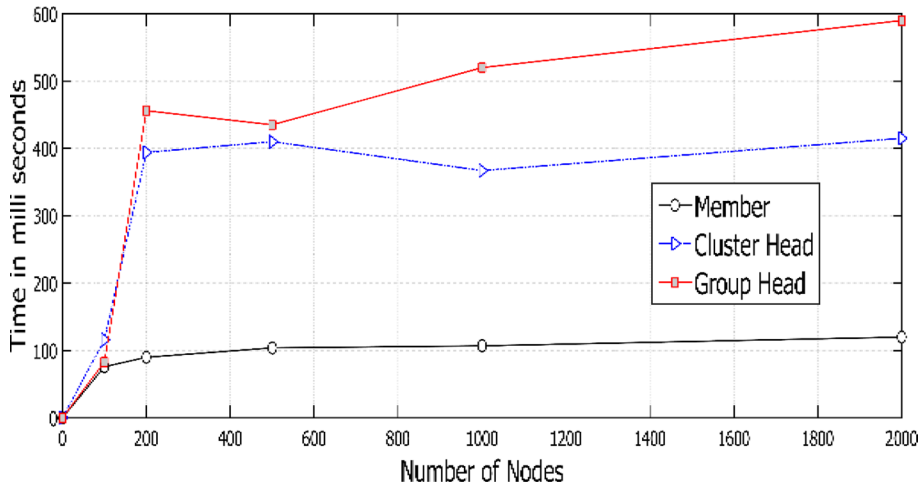

Fig. 15 Computation time: member vs cluster head vs group head in entire GK generation

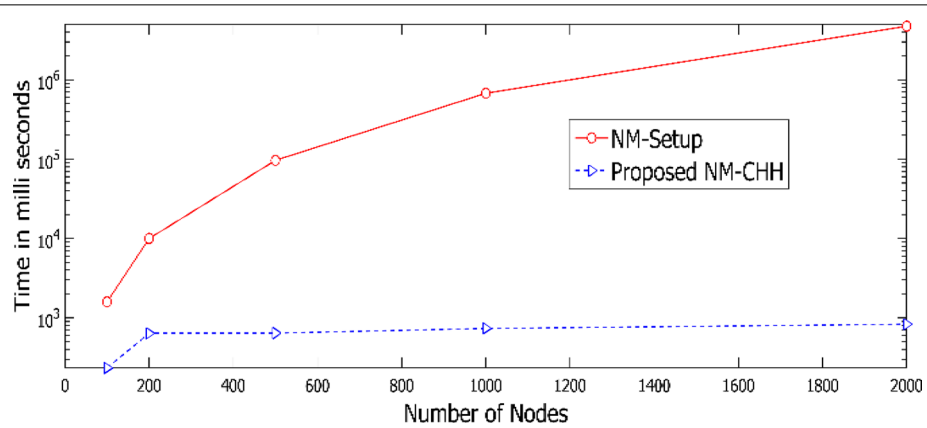

Fig. 16 Comparison of computation time between NM-setup and NM-CHH

Figure 13 indicates comparison between computation time of member and cluster head in level-I. We can observe that the computation load on cluster head is relatively higher than individual members in level-I of NM-CHH-GKA.

Figure 14 indicates comparison between Computation time of cluster head as a Member and Group Head in level-II. We can observe that the computation load on Group Head is relatively higher than individual cluster head in NM-CHH-GKA.

Figure 15 indicates comparison of computation time among individual Member, Cluster Head, Group Head in Entire GK Generation. We can observe that the computation load on Group Head is relatively higher than individual cluster head which is relatively higher than individual members in NM-CHH-GKA.

Figure 16 indicates comparison of computation time between NM.Setup and NM-CHH-GKA. We can observe that the computational load on NM-CHH-GKA is highly reduced relative to NM.Setup by splitting large group into a certain number of clusters.

The findings in "Computational complexities using graphs", "Communication complexities using graphs" and "Experimental results" sections are the complexities of NM-CHH-GKA in the context of computation, communication and experimental results

respectively when compared to existing protocols. From these sections we can conclude that our protocol is optimal with respect to all the three dimensions. So NM-CHH-GKA is suitable for recourse constrained networks such as WANETS.

Conclusion and future scope

In this paper a new scalable NM-CHH GKA protocol was proposed based on parallel computing for large dynamic groups with less computational capabilities. Novel architectural design of our protocol provides flexibility and reduces cryptographic workload. The two level NM-CHH-GKA scheme allows on hand NM-GKA scheme to implement at cluster level to achieve scalability and robustness without sacrificing efficiency. The advantage of hierarchical management includes freeing the group controller looking after several members, enhancing security, improving scalability together with all cluster requiring minimal space for dealing with protocol. As a key management technique, proposed protocol uses cluster-based hybrid hierarchical scheme reducing rekeying workload of the networks while limiting the failure to local cluster without affecting other clusters. Comparative analysis showed that proposed protocol provides better performance in view of both communication and computation expenses. Further we established a formal security model for the proposed NM-CHH-GKA under cryptographic assumptions.

Security of CHH-GKA in WANETs is inadequate in the presence of node misbehaviour and internal attacks. It is because an opponent may start security attacks with the security keys obtained from compromised nodes. To isolate misbehaving node from legitimate data transmission as a future scope we plan to integrate trust enhanced module using Fuzzy Trust Based rules to NM-CHH GKA to develop a trust enhanced secure clustering framework for WANETs.

Acknowledgements
I would like to thank my parents, family members and Management of Sri Vasavi Engineering College, Tadepalligudem who encouraged and supported me to do this work. Further I am very much thankful to reviewers and Journal Authorities.

Authors' contributions
The first author VSN conceived of the presented idea and developed the theory and performed the computations. The second author verified the analytical methods and security analysis. The first author VSN encouraged the third author to implement and supervised the findings of this work. All authors discussed the results and contributed to the final manuscript. All authors read and approved the final manuscript.

Funding
Not currently in receipt of any research funding for this paper.

Data availability statement for the data used in this manuscript
The Experimental data used to support the findings of this study are available from the corresponding author upon request, with this readers can access the data supporting the conclusions of the study.

Competing interests
The authors declare that they have no competing interests

Author details
[1] Department of Computer Science and Engineering, Sri Vasavi Engineering College, Tadepalligudem, Andhra Pradesh 534101, India. [2] Department of Computer Science and Engineering, Anil Neerukonda Institute of Technology & Science, Visakhapatnam, Andhra Pradesh 530003, India. [3] Department of Mathematics, Andhra University, Visakhapatnam, Andhra Pradesh 530003, India.

Received: 10 December 2018 Accepted: 22 June 2019
Published online: 08 July 2019

References

1. E-Bashary M, Abdelhafez A, Anis W (2015) A comparative study of group key management in MANET. Int J Eng Res Appl 5(8):85–94
2. Boneh D, Franklin M (2001) Identity-based encryption from weil pairing. In: Proceedings of crypto 2001, LNCS, vol 2139. Springer-Verlag, Berlin, pp 213–229
3. Burmester M, Desmedt Y (2005) A secure and scalable group key exchange system. Inf Process Lett 94(3):137–143
4. Manulis M. Security-focused survey on group key exchange protocols. http://eprint.iacr.org/2006/395
5. Scott M, Costigan N, Abdulwahab W. Implementing cryptographic pairings on smart cards. http://www.iacr.org/2006/144
6. Barreto PSLM, Kim HY, Scott M (2002) Efficient algorithms for pairing based cryptosystems. In: Proceedings of crypto 2002, LNCS, vol 42. Springer-Verlag, Berlin, pp 354–368
7. Dutta R, Barua R (2008) Provably secure constant round contributory group key agreement in dynamic setting. IEEE Trans Inf Theory 54(5):2007–2025
8. Dutta R, Barua R (2005) Constant round dynamic group key agreement. In: Proceedings of ISC 2005, LNCS, vol 3650, Springer-Verlag, Berlin. pp 74–88
9. Dutta R, Barua R. Overview of key agreement protocols. http://eprint.iacr.org/2005/289
10. Dutta R, Barua R, Sarkar P (2004) Provably secure authenticated tree based group key agreement. In: Proceedings of ICICS'04, LNCS, vol 3269. Springer-Verlag, Berlin, pp 92–104
11. Kim Y, Perrig A, Tsudik G (2004) Tree-based group key agreement. ACM Trans Inf Syst Secur 7(1):60–96
12. Kleinrock L, Kamoun F (1977) Hierarchical routing for large networks; performance evaluation and optimization. Comput Netw 1(3):155–174
13. Basagni S (1999) Distributed clustering for ad hoc networks. In: Proceedings of the international symposium on parallel architectures, algorithms, and networks (ISPAN), IEEE, Perth, Australia, pp 310–315
14. Steenstrup M (2001) Cluster-based networks. C.E. Perkins, Addison Wesley, Boston, pp 75–138
15. Szczechowiak P, Oliveira L, Scott M, Collier M, Dahab R (2008) NanoECC: testing the limits of elliptic curve cryptography in sensor networks. In: 5th European conference on wireless sensor networks—EWSN 2008, lecture notes in computer science, vol 4913. Springer-Verlag, Berlin, pp 305–320
16. Naresh VS, Murthy NV (2015) Provably secure group key agreement protocol based on ECDH with integrate signature. Secur Commun Netw 9(10):1085–1102
17. Bemmoussat C, Didi F, Feham M (2013) Cluster based routing protocol in wireless mesh network. In: International conference on computer applications technology (ICCAT), Jan 2013, pp 1–6
18. Belding-Royer EM (2002) Hierarchical routing in ad hoc mobile networks. Wirel Commun Mob Comput 2(5):515–532
19. Virtanen SE, Nikander P (2004) Local clustering for hierarchical ad hoc networks. In: Proceedings of WiOpt: modeling and optimization in mobile, ad hoc and wireless networks, pp 404–405
20. Abdel-Hafez A, Miri A, Oronzo-Barbosa L (2007) Authenticated group key agreement protocols for ad hoc wireless networks. Int J Netw Secur 4(1):90–98
21. Teo JCM, Tan CH (2005) Energy-efficient and scalable group key agreement for large ad hoc networks. In: Proceedings of the 2nd ACM international workshop on performance evaluation of wireless ad hoc, sensor, and ubiquitous networks, pp 114–121
22. Galbraith S, Harrison K, Soldera D (2002) Implementing the Tate pairing. In: Proceedings of algorithm number theory symposium—ANTS V, LNCS, vol 2369. Springer-Verlag, Berlin, pp 324–337
23. Klaoudatou E, Konstantinou E, Kambourakis G, Gritzalis S (2011) A survey on cluster-based group key agreement protocols for WSNs. IEEE Commun Surv Tutor 13(3):429–442
24. Klaoudatou E, Konstantinou E, Kambourakis G, Gritzalis S (2008) Clustering oriented architectures in medical sensor environments. In: International workshop on security and privacy in e-health, Barcelona, March 2008. IEEE CS Press, pp 929–934
25. Yao G, Ren K, Bao F, Deng RH, Feng D (2003) Making the key agreement protocol in mobile ad hoc network more efficient. In: 1st international conference on applied cryptography and network security—ACNS 2003, lecture notes in computer science, vol 2846. Springer-Verlag, Berlin, pp 343–356
26. Shi H, He M, Qin Z (2006) Authenticated and communication efficient group key agreement for clustered ad hoc networks. In: 5th international conference on cryptology and network security—CANS 2006, lecture notes in computer science, vol 4301, Springer-Verlag, Berlin, pp 73–89
27. Gomathi K, Parvathavarthini B, Saravanakumar C (2017) An efficient secure group communication in MANET using fuzzy trust based clustering and hierarchical distributed group key management. Wirel Pers Commun 94(4):2149–2162
28. Hietalahti M (2008) A clustering-based group key agreement protocol for ad hoc networks. Electron Notes Theor Comput Sci 192:43–53
29. Li X, Wang Y, Frieder O (2002) Efficient hybrid key agreement protocol for wireless ad hoc networks. In: Proceedings of IEEE international conference on computer communications and networks, pp 404–409
30. Abdel-Hafez A, Miri A, Oronzo-Barbosa L (2006) Scalable and fault-tolerant key agreement protocol for dynamic groups. Int J Netw Manag 16(3):185–201
31. Teo JC, Tan CH (2007) Denial-of-service resilience password-based group key agreement for wireless networks. In: Proceedings of the 3rd ACM work-shop on QoS and security for wireless and mobile networks (Chania, Crete Island, Greece), October 22. ACM, New York, pp 136–143
32. Hussain K, Abdullah AH, Iqbal S, Awan K, Ahsan F (2013) Efficient cluster head selection algorithm for manet. J Comput Netw Commun 2013(7):1–7
33. Dutta R, Dowling T (2009) Secure and efficient group key agreements for cluster based network. In: Transactions on computational science IV: special issue on security in computing, lecture notes in computer science, vol 5430. Springer-Verlag, Berlin, pp 87–116
34. Diffie W, Hellman M (1976) New directions in cryptography. IEEE Trans Inf Theory 22:644–654

35. Joux A (2000) A one round protocol for tripartite Diffie–Hellman. In: Algorithmic number theory symposium—ANTS IV, LNCS, vol 1838. Springer-Verlag, Berlin, pp 385–394
36. Steiner M, Tsudik G, Waidner M (1996) Diffie–Hellman key distribution extended to group communication. In: Proceedings of the 3rd ACM conference on computer and communications security. ACM Press, New York, pp 31–37
37. Barua R, Dutta R, Sarkar P (2003) Extending Joux's protocol to multi party key agreement. In: Progress in cryptology—INDOCRYPT 2003, lecture notes in computer science, vol 2904. pp 205–217
38. Naresh VS, Murthy NV (2015) A new two-round dynamic authenticated contributory group key agreement protocol using elliptic curve Diffie–Hellman with privacy preserving public key infrastructure. Sadhana 40:2143–2161
39. Chen Y, Zhao M, Zheng S, Wang Z (2006) An efficient and secure group key agreement using in the group communication of mobile ad hoc networks. In: International conference on computational intelligence and security, IEEE Press, pp 1136–1142
40. Ayman ELS (2014) A new hierarchical group key management based on clustering scheme for mobile ad hoc networks. IJACSA 5(4):208–219
41. Krishna P, Vaidya NH, Chatterjee M, Pradhan DK (1997) A cluster-based approach for routing in dynamic networks. In: ACM SIGCOMM computer communication review, pp 49–65
42. Dutta R, Dowling T (2011) Provably secure hybrid key agreement protocols in cluster-based wireless ad hoc networks. Ad Hoc Netw 9(5):767–787
43. Niu Q (2014) ECDH-based scalable distributed key management scheme for secure group communication. J Comput 9(1):153–160
44. Balasubramanian A, Mishra S, Sridhar R (2005) Analysis of a hybrid key management solution for ad hoc networks. In: IEEE wireless communications and networking conference. IEEE Press, New York, pp 2082–2087
45. Katz J, Yung M (2003) Scalable protocols for authenticated group key exchange. In: Advances in cryptology—CRYPTO 2003, lecture notes in computer science, vol 2729. Springer-Verlag, Berlin, pp 110–125
46. Bresson E, Chevassut O, Pointcheval D (2002) A dynamic group Diffie–Hellman key exchange under standard assumptions. In: Proceedings of Eurocrypt 2002, LNCS, lecture notes in computer science, vol 2332. pp 321–336
47. Tan CH, Teo JCM (2006) Energy-efficient ID-based group key agreement protocols for wireless networks. In: 2nd international workshop on security in systems and networks—SSN 2006, IEEE Press, New York

Publisher's Note