DOI: 10.21917/ijct.2018.0250

GPRKEY - A NOVEL GROUP KEY REKEYING TECHNIQUE FOR MANET

C. Shanmuganathan[1] and P. Raviraj[2]
[1]Department of Computer Science and Engineering, Manonmaniam Sundaranar University, India
[2]Department of Computer Science and Engineering, GSSS Institute of Engineering and Technology for Women, India

Abstract

A Mobile Ad hoc Network (MANET) is a collection of autonomous nodes or mobile devices that can arrange themselves in various ways and work without strict network administration. Ensuring security in mobile ad hoc networks is a challenging issue and most of the applications in mobile ad hoc networks involve group oriented communication. Mostly cryptographic techniques are used to provide the security to MANETs. Cryptographic techniques will not be efficient security mechanism if the key management is weak. The issue of packet loss in MANET that is caused due to multi casting and backward and forward secrecy results in mobility. Hence, we investigate on this issue and propose a method to overcome this scenario. On analysing the situation we find that frequent rekeying leads to huge message overhead and hence increases energy utilization. With the existing key management techniques it causes frequent disconnections and mobility issues. Therefore, an efficient multi casting group key management will help to overcome the above problems. In this paper we propose a novel group key rekeying technique named GPRKEY (Group key with Periodic ReKEYing) deal with scalability issue of rekeying and also analyze the performance of the newly proposed key management method using key trees. In this approach we use the periodic rekeying to enhance the scalability and avoid out of sync problems. We use sub trees and combine them using the merging algorithm and periodic re-keying algorithm. The GPRKEY is evaluated through NS-2 simulation and compared with existing key management techniques OFT (One-way Function Tree) and LKH (Logical Key Hierarchy). The security and performance of rekeying protocols are analyzed through detailed study and simulation.

Keywords:

Mobile Ad Hoc Networks, Periodic, Security, Rekey, Sub Tree

1. INTRODUCTION

Mobile ad hoc networks (MANETs) are known to have many security problems because of open medium, dynamic network topology, decentralized control, no centralized authority, lack of facilities in mobile devices and no clear rules for protection. Many mobile applications in MANET such as military, emergency response networks, M-commerce, online gaming, and combined work are based on the concept of group communications. While designing protocols for secure group communication systems in mobile ad hoc networks faces many technical difficulties and there are two ways of attack such as inside and outside attack. To deal with attacks from outside, one way is to use a symmetric key called the group key. The group key shared among all the users in a group. The group key will encrypt messages sent by a member (sender) in the group. Only group members (receiver) with the group key are able to decrypt the messages. Thus, the group key protects group communication information shared by authorized members. Since there is no fixed infrastructure support in MANETs, key management must be accomplished in a fully distributed manner. This creates additional processing and communication overheads whenever the group key is rekeyed because of a group member leave or joins frequently. Many mobile resources constrained such as bandwidth, memory size, battery life, and computational power affected the security [17]. Group formation or partitioning affects with many factors like eavesdropping and security threats, unreliable communication, no fixed infrastructure, and frequent changes in network topology because of user mobility [18]. Particularly, a gathering key administration framework can execute two sorts of getting to control: in reverse get to control and forward get to control. If the framework changes the gathering key after another client's joints, the new clients won't have the capacity to decode past gathering correspondences, this is called in reverse get to control. Instead of assigning the individual key for all users, a secret key is used for the entire group is called a group key [13]. When a new member joins a group, the group key is rekeyed immediately to ensure that the new member cannot decrypt old messages, this requirement is called backward secrecy [12]. When an existing member leaves the group, the group key is rekeyed immediately to ensure that future communications cannot be decrypted by the outside member, this requirement is called forward secrecy. An algorithm that deals with the generation, distribution, updating, and revocation of group keys is called a group key management protocol [27] [28].

This method provides many gathering correspondence applications, for example, pay-per-see, web-based educating, and offer quotes. Prior to these gathering focused multi-cast applications can be effectively conveyed, get to control system must be produced to such an extent that alone approved individuals can get to gathering correspondence. Forward mystery implies that a withdrawing part cannot acquire data about future gathering correspondence and in reverse mystery implies that a joining part cannot get data about past gathering correspondence. We expect the presence of a put stock in substance, known as the Group Controller (GC), which is in charge of refreshing the gathering key. This enables the gathering participants to scale to large gatherings [13].

2. RELATED WORK

Renuka et al. [4] propose a various levelled assemble key administration plans that is progressive and completely dispersed with no central authority and utilizing an ease rekeying system which is reasonable for huge and high mobility mobile ad hoc networks. The rekeying strategy requires just a single round in our plan and, Diffie Hellman Group Diffie Hellman, Burmester, and Desmedt it is a consistent three, while in different plans, for example, Distributed Logical Key Hierarchy and Distributed one way function trees, it relies on upon the number of individuals. We diminish the energy consumption of the keying materials by decreasing the quantity of bits in the rekeying message [22]. We

appear through examination and reproductions that our plan has less calculation, correspondence and energy consumption contrasted with the current plans [4]. Chrisment and Festor et al. [5] explains the problems and major issues about the key management for securing MANETs and gives a scientific categorization of these procedures in MANETs. Another approach, called BALADE, is likewise introduced. This technique increases the bandwidth consumption and energy. It depends on a consecutive portability and captivity of nodes [5]. A scientific classification of gathering key administration procedures for securing multicast communications in ad hoc networks [14], considering the qualities and criteria of such condition, which are nodes mobility support, energy efficiency and multi-hop awareness. We talked about these procedures and contrasted their agreeing with security and execution measurements, which are the security services (information secrecy and honesty, nodes verification and revocation), the capacity cost, the vulnerabilities or the shortcomings and the scalability [5]. Two sorts of contemporary improvements in cryptography are inspected. Enlarging uses of teleprocessing has offered ascend to a requirement for new sorts of cryptographic systems, which limit the requirement for secure key appropriation stations and supply the equivalent of written signature. Whitefield and Hellman propose approaches to take care of these presently open issues. It additionally talks about how the speculations of correspondence and calculations are starting to give the instruments to take care of cryptographic issues of long standing [6]. Striki and Baras [7] build up a protected, vigorous, and adaptable key administration plan for multicast communications in Mobile Ad Hoc Networks (MANETs). Most key dispersion plans of today are fundamentally intended for wire-line systems. Frequent node disappointments, network segments, inefficient computational capacities of wireless nodes, network delay, poor quality of signal, and so on, is a portion of the reasons why they neglect to work appropriately in MANETs. We order existing and recently created procedures in two families, contributory and non-contributory. We access them and compare their execution, Striki et al. concentrate on contemplating and creating key circulation methods that accomplish scalability and high execution of our framework without giving up the security level of the network. To this end, we likewise planned a various levelled two-level hybrid key administration scheme that uses a portion of the above procedures in the proper mixes to additionally diminish the capacity, communication and calculation expenses of nodes [7]. Many developing applications in MANETs include group-oriented communication. Multicast is an effective method for supporting group-oriented application, for the most part in portable condition with constrained bandwidth and restricted power [15] [16]. For utilizing such applications in an ill-disposed condition as military, it is important to give secure multicast communication. Key administration is the basic test in planning secure multicast communication. Multicast key dispersion needs to defeat the testing component of "1 affects *n*" phenomenon. To beat this issue, multicast group clustering is the best solution. Malik [1] proposed two plans based on different architectures. One of the plans enhances the security of the OFT plan. We demonstrated the flexibility of proposed plan by breaking down various cases. Other proposed plot enhances the execution of autonomous key hierarchy system (OFT) [21]. Lie et al. [24] proposed a change over LKH called One-way Function Trees (OFT). With OFT, a KEK is computed by members instead of attributed by the key server. OFT permits to decrease the quantity of rekey messages from $2\log_2(n)$ to just $\log(n)$.

3. PROPOSED FRAMEWORK

We implement a merging algorithm for to merge the two sub trees involved in the event. In order to make the algorithm efficient to handle both join and depart requests, we have the merging algorithm as periodic balanced algorithm. There are two algorithms to combine two sub trees called ST_X and ST_Y assuming that ST_X is greater in height than ST_Y and also both the sub trees are of same out degree *f*.

3.1 MERGING ALGORITHM 1

The merging algorithm 1 is used under the condition when the maximum height between two sub trees ST_X and ST_Y is greater than or equal to 1.

Merging Algorithm defined as below:

Merging algorithm 1 is applied when the difference between $DMAX_{STX}$ and $DMIN_{STY}$ is greater than 1 and greater than or equal to 1. Only when both the conditions are satisfied the algorithm computes *DINSERT*.

Step 1: When $f > 2$, it searches for an empty child node in ST_X at both the level *DINSERT* or *DINSERT* - 1. If *DINSERT* = 0, then levels 0 and 1 are searched. When a node of such value exists, then the algorithm inserts ST_Y as the child of that particular key node.

Step 2: If the empty node is not identified in Step 1, then a suitable key node is ST_X at level *DINSERT* =0, a suitable key node at level 1 is marked. The marked key node is given by the one with the greater number of leaf node at level $DMIN_{STX}$.

Step 3: When $f > 2$, and an empty node is not found in step 1, the algorithm searches the root of ST_Y for the empty node. If exists, then the algorithm inserts the marked key node from step 2 as the child of ST_Y and inserts ST_Y at the old location of the marked key node.

Step 4: When $f = 2$ or $f > 2$, where step 1 to 3 failed to insert ST_Y into ST_X, then the algorithm creates a new key node at the old location of the marked key node (from step 2) and inserts the marked key node and ST_Y as its children.

Finally, the group communication is made to update the message to the affected members of the group.

3.2 MERGING ALGORITHM 2

Merging Algorithm 2 is used for combining the sub trees at the condition where the height difference is 0 or equal to 1. Merging algorithm 2 is implemented when the difference between $DMAX_{STX}$ and both $DMIN_{STY}$ and $DMAX_{STX}$ is 0 or equal to 1. The below steps are performed.

Step 1: For $f > 2$, it searches the root of ST_X to find an empty child key node. If it exists, then the algorithm inserts ST_Y at the empty child key node.

Step 2: For $f = 2$ or when Step 1 is not valid for $f > 2$, it creates a new key node at the root and inserts ST_X and ST_Y as its children.

The GC will now a multicast the update message to all the existing members of the group. Once the affected node IDs is updated the members can now differentiate the set of keys that they need in the rekey messages.

4. PERIODIC REKEYING ALGORITHM

In case of periodic rekeying, the key server will maintain a key tree that is different from the key tree [20]. It means that this key tree maintained by the key server holds information about null nodes that are empty key nodes so to maintain a complete balanced key tree [19]. Each node is named with an integer ID in breadth-first search order with the tree root named as 0.

During the end of the rekeying interval, all the join and leave requests are collected and executed and updates the key tree and generates a rekey sub tree. The agenda of merging algorithm are to

1) To reduce encrypted keys.
2) Update the key tree and
3) To make it easy for the users to identify the encrypted keys.

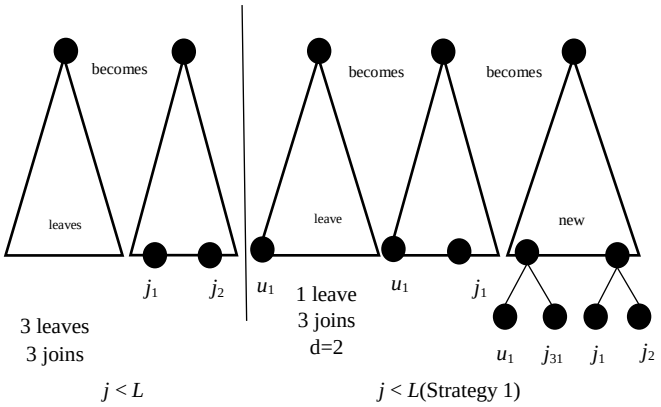

Fig.1. Example of merging algorithm for $j \neq L$

Firstly, the key tree is updated in the merging algorithm. When $j \leq L$, all the departed users J are replaced with newly joined users j with the smallest IDs. In this way, the number of encrypted keys is reduced. When $j < L$, here, not all the departed users, are replaced and hence those nodes that are not replaced are named as null nodes. If all the children of a node are null nodes then the entire node is named as a null node. If $j > L$, here, all L departed users are first replaced with L newly joined users. Still, the server needs to remain j-L new users. Hence, three strategies are investigated for the above mentioned objectives.

Strategy 1: In this strategy, the key server first separates the L replaced nodes to add the balance new users. Even after splitting the replaced nodes it is not sufficient to add all the remaining new users (i.e. $j > d \times L$), then the key server splits the nodes from left to right to add the new users. This strategy helps in reducing the number of encrypted keys since it first splits the replaced user nodes and the disadvantage is that if in case some users are changed then the key server needs provides new IDs to those users individually, and this can cause the key server bandwidth overhead.

Strategy 2: This strategy is efficient than the above strategy by reducing the number of encrypted keys. In this approach, the key server creates a tree for new users and grafts the tree with departed user node with the smallest height. But this approach does not maintain a balanced key tree as seen in the above strategy 1. The positive approach of this is that the ID of maximum only one user is modified and hence there is no requirement of updating new IDs to the remaining users.

Strategy 3: This strategy is proposed to make the task of identifying the encrypted keys that the users need. In this strategy, the following steps are repeats until all new users are added to the key tree. The null nodes with Ids between $d \times m+1$ and $d \times m+d$ are replaced by the users that are recently connected. Here the last node in the tree treated as neither user node nor null node. It is denoted as m. If in case there are still extra joining with user node ID $m+1$, the key server splits to add the children and moves the content of the user node to its left most child and then adds d-1 new user nodes. The main advantage of this strategy is that each remaining user can individually derive ID of its user node in spite of modification in the complete structure of key tree and drawback of this approach is it causes a huge number of encrypted keys. Comparison of the above three strategies in the case of $j > L$, our costing shows the sub tree rekey sizes is very small. The key server updates the state of the key nodes and makes the copy of the key tree when once the key tree is updated.

The nodes will have one of the following labels: Unchanged, Leave, Join, Replace.

First, the state of user nodes is marked.

- A user node is named as Unchanged only when it is changed by the following rules.
- A user node of an excluded user is marked as Leave and if it is not replaced then it is marked as replace.
- When a user node is a replacement of a null node it is marked as Join and even when it is split from a previous user node.

Now, other key nodes are marked.

- When all the children of key node are named as Leave then all the children are removed.
- Else, if all the children are named as unchanged, then we mark them Unchanged and remove them all.
- If all the children are Unchanged or Join then we mark it as Join and replace all unchanged children by virtual node consist of the old key of the key node.

If the node has one Replace of Leave child then we mark as Replace

5. REKEY TRANSPORT WORK LOAD

In the concept of rekeying, the individual needs only the encrypted key required for it and hence it need not receive all the packets [25]. Hence, in order to avoid overhead in unicasting the individual encrypted keys, the key server actually partitions the users into several small groups called as subgroups and then combines the encrypted keys of the subgroup users into a rekey message. This gets portioned into several rekey packets. The rekey message is the multicast to all the users in the subgroup. Finally, the user selects its packet based on how encrypted keys

are assigned into rekey packets. Presently the inquiry limits to how to adjust the workload of transporting these rekeys [26].

5.1 KEY ASSIGNMENT ALGORITHM

To enhance the execution of rekey transport procedure, it is required that a key assignment algorithm diminishes the quantity of packets Z_f that a client r needs to get. Additionally, the overhead of rekey transport likewise relies on upon the clients with the biggest quantities of packets to get. Along these lines, it is desired that key assignment algorithm additionally lessens the variance of $\{Z_f\}$. Given these requirements, we consider three key task assignment algorithms; namely, Depth First Assignment (DFA), Breadth First assignment (BFA) and Recursive Breadth First Assignment (R-RFA). The above key assignment algorithms do not copy the encrypted keys normally. That is each packet is allocated to a separate encrypted key. We have also proposed and inspected an alternate calculation called the client situated Assignment (CSA). The preferred standpoint of the UKA calculation is that it designates most of the encoded keys for a customer into a comparable bundle, and along these lines, each customer needs to get only a solitary parcel that is, $Z_f = 1$ for all gatherers. The weakness of this algorithm, nonetheless, is that some encoded keys are replicated into a few bundles, and such duplication can command data transfer capacity overhead, especially when MTU is little or when beneficiary loss rates are low.

For BFA and DFA, the key server explores a rekey sub tree using either breadth first or depth first, and allows progressively the 31 scrambled keys into packets. By on a level plane looking at a rekey sub tree, BFA assembles keys from different customers in a round-robin way. This fairness for each customer diminishes the difference of $\{Z_f\}$. On the other hand, BFA spreads the keys of a customer into various packets, and increment the average of $\{Z_f\}$. By vertically following along a way, DFA first accumulates the keys for one customer, and subsequently goes to the accompanying customer. In this way, we expect that the average of $\{Z_f\}$ is littler for DFA. Be that as it may, since the shared encoded keys are allotted to the customers arranged some time recently, such inclination causes a bigger change of $\{Z_f\}$. To get the upsides of both BFA and DFA, we consider R-BFA. BFA algorithm demonstrates our R-BFA algorithm.

This algorithm begins by calling R-BFA (root), where the root is the root node a rekey sub tree.

Algorithm R-BFA ($node_{id}$)

$node_{id}$ extraordinarily recognizes a node in a rekey sub tree.

PKT is a worldwide variable, signifying a rekey packet.

Family(i) is the set containing i and its immediate children.

Step 1: Create a local FIFO queue and assign to Q.

Step 2: Assign the $node_{id}$ into Q

Step 3: Repeat the following Step 4 to 6 until Q is not empty.

Step 4: Pop the head component and assign to i

Step 5: If packet has free space store Family ($node_{id}$)

Then put all the child node of i into Q and

Put all the encrypted keys of i into PKT

Else create another rekey packet and assign to PKT

Step 6: Call R-BFA(i)

Step 7: Repeat the following till Q is not empty

Step 8: pop the head (Q) and assign to i

Step 9: call R-BFA(i)

To better grasp R-BFA, we contrast its conduct and that of BFA. Right, when there is still space in the present packet, R-BFA carries on simply like BFA, and in this way has execution with respect to difference like that of BFA. In any case, when the present packet is full and another parcel is made, as opposed to continuing on a level plane looking at on the worldwide rekey sub tree (as BFA will do), R-BFA does BFA inside a neighbourhood sub tree. In this way, R-BFA collects more related keys and lessens the normal value of $\{Z_f\}$ differentiated and that of BFA.

6. SIMULATION RESULTS

We compare our proposed system with OFT and LKH by varying number of nodes inside the network.

6.1 SCENARIO DESCRIPTION

The simulation results, and comparative analysis is carried out by using the Network Simulator - 2 is presented below. We evaluate the performance and effectiveness of the proposed technique GPRKEY through this simulation. The simulation environment and the performance metrics are shown in Table.1. This simulation is performed for the network size varying from 40 nodes to one hundred nodes and by the varying height of the tree. Finally, a comparative study of the proposed technique GPRKEY with the existing protocols OFT [23] and LKH [21] results are presented in the graphs.

Table.1. Simulation Parameters

Parameters	Values
Simulator	NS-2.33
Topology	Random
Number of nodes	50,100,150,200,250
Wi-Fi Data Rate	1Mbps
Propagation model	Free space
Physical Model	Wireless phy
Antenna Model	Omni Antenna
Queue Size	50
Traffic Type	CBR,UDP
Mobility Model	Random Way Point
Routing Algorithm	GPRKEY
Packet Size	512
Mac protocol	802.11 standard

The Fig.2 and Fig.3 shows that our proposed GPRKEY has less broadcasting cost while leave and joins when compared with existing OFT [30] and LKH schemes. We prove that our scheme gives better costs even in large groups [24]. The Fig.4 shows that average time for key verification and it shows that our proposed scheme takes lesser time than OFT and LKH. The Fig.5 shows the average hops between a pair of nodes for overall network, and it shows our proposed scheme used minimum hops than OFT and LKH scheme.

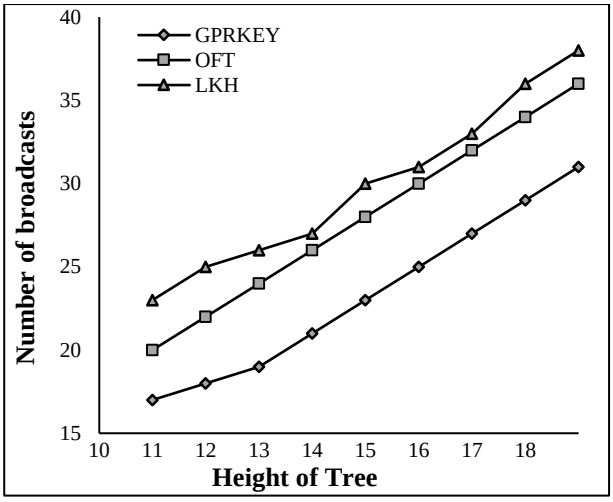

Fig.2. Comparison Graph for broadcasting cost while nodes Leaves from the group for varying Height of the Tree

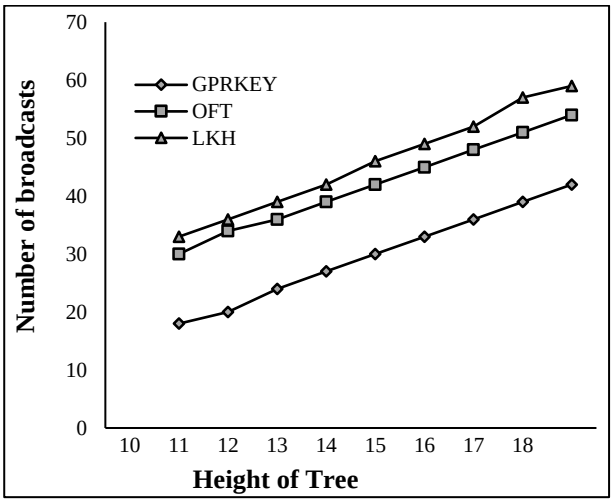

Fig.3. Comparison Graph for broadcasting cost while nodes Joins into the group for varying Height of the Tree

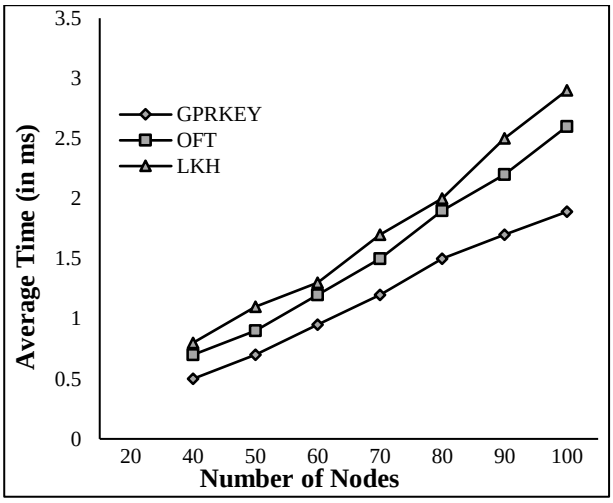

Fig.4. Comparison Graph for Average Time for Key verification for varying number of nodes

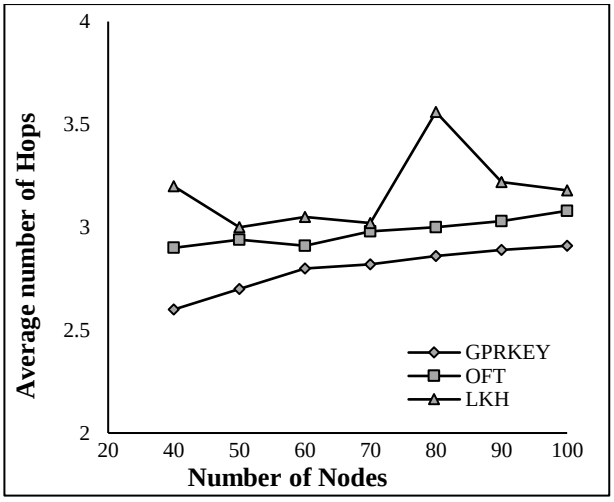

Fig.5. Comparison Graph for Average Number of Hops between a pair for varying number of nodes

7. CONCLUSIONS

We have investigated on the scalability property of the proposed algorithm in reliable group rekeying and also analysed the performance of GPRKEY. The rekeying process is done periodically unlike the rekeying that happens after each join or leave in order to eliminate out-of-sync issues. We use merging algorithm to combine the sub trees. The experimental result on re-key transport shows that it has a positive impact on the reliability factor and also contains sparseness property. By our extensive simulations, we show that our proposed GPRKEY is better than OFT and LKH by reducing the broadcasting cost and communication overhead. This method spends the only minimum number of hops for communication, and time took for key verification is less compared to existing methods.

REFERENCES

[1] Muhammad Yasir Malik, "Efficient Group Key Management Schemes for Multi Cast Dynamic Communication Systems", Available at: https://eprint.iacr.org/2012/628.pdf, Accessed on 2012.

[2] Bing Wu, Jie Wu, Eduardo B. Fernandez and Spyros Magliveras, "Secure and Efficient Key Management in Mobile Ad Hoc Networks", *Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium*, pp. 21-25, 2005.

[3] Zainab R. Zaidi, Brian L. Mark, "Mobility Tracking Based on Autoregressive Models", *IEEE Transactions on Mobile Computing*, Vol. 10, No. 1, pp. 32-43, 2009.

[4] A. Renuka and K.C. Shet, "Hierarchical Approach for Key Management in Mobile Ad hoc Networks", *International Journal of Computer Science and Information Security*, Vol. 5, No. 1, pp. 87-95, 2009.

[5] M.S. Bouassida, Isabelle Chrisment and Oliver Festor, "Group Key Management in MANETs", *International Journal of Network Security*, Vol. 6, No. 1, pp. 67-79, 2008.

[6] D. Whitefield and Martin E. Hellman, "New Direction in Cryptography", *IEEE Transactions on Information Theory*, Vol. 22, No. 6, pp. 73-77, 1976.

[7] Maria Striki and John S. Baras, "Key Distribution Protocols for Secure Multi Cast Communication Survivable in MANETs", *Proceedings of IEEE Military Communications Conference*, pp. 112-116, 2003.

[8] D. Suganyadevi and G. Padmavathi, "Dynamic Clustering for QoS based Secure Multi Cast Key Distribution in Mobile Ad Hoc Networks", *International Journal of Computer Science*, Vol. 7, No. 5, pp. 11-16, 2010.

[9] Dijiang Huang and Deep Medhi, "A Secure Group Key Management Scheme for Hierarchical Mobile Ad Hoc Networks", *Ad Hoc Networks*, Vol. 6, No. 5, pp. 560-577, 2007.

[10] Chung Kei Wong and Simon S. Lam, "Keystone a Group Key Management Systems", *Proceedings of International Conference on Telecommunication*, pp. 1-5, 2000.

[11] Roger G. Kermode, "Scoped Hybrid Automatic Repeat Request with Forward Error Correction", Proceedings of ACM *International Conference on Applications, Technologies and Architectures*, pp. 11-15, 1998.

[12] S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication", *ACM Computing Surveys*, Vol. 35, No. 3, pp. 309-329, 2003.

[13] W.C. Ku and S.M. Chen, "An Improved Key Management Scheme for Large Dynamic Groups using One-Way Function Trees", *Proceedings of the International Conference on Parallel Processing Workshops*, pp. 391-396, 2003.

[14] C. Siva Ram Murthy and B.S. Manoj, "*Ad Hoc Wireless Networks: Architectures and Protocols, Portable Documents*", 1st Edition, Prentice hall, 2004.

[15] S. Singh, M. Woo and C. Raghavendra, "Power Aware Routing in Mobile Ad Hoc Network", *Proceedings of International Conference on Mobile Computing and Networking*, pp. 181-190, 1998.

[16] V. Rodoplu and T. Meng, "Minimum Energy Mobile Wireless Networks", *IEEE Journal on Selected Areas on Communications*, Vol. 17, No. 8, pp. 1333-1344, 1999.

[17] J. Zhu and X. Wang, "Peer: A Progressive Energy Efficient Routing Protocol for Wireless Ad Hoc Networks", *Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 3, pp. 1887-1896, 2005.

[18] S. Yi, P. Naldurg and R. Kravets, "Security-Aware Ad-Hoc Routing for Wireless Networks", *Proceedings of 24th ACM International Conference on Mobile Ad Hoc Networking and Computing*, pp. 299-302, 2001.

[19] Wee Hock Desmond Ng, Michael Howarth, Zhili Sun and Haitham Cruickshank, "Dynamic Balanced Key Tree Management for Secure Multi cast Communications", *IEEE Transactions on Computers*, Vol. 56, No. 5, pp. 590-605, 2007.

[20] X.S. Li, Y.R. Yang, M. Gouda and S. Lam, "Batch Re-Keying for Secure Group Communications", *Proceedings of 10th International Conference on World Wide Web*, pp. 525-534, 2001.

[21] J. Pegueroles and F. Rico-Novella, "Balanced Batch LKH: New Proposal, Implementation and Performance Evalution", *Proceedings of 8th IEEE International Symposium on Computers and Communication*, pp. 421-427, 2003.

[22] A. John Prakash, V. Rhymend Uthariaraj and B. Lydia Elizabeth, "Efficient Key Management Protocol with Predictive Rekeying for Dynamic Networks", *Proceedings of 2nd International Conference on Green High Performance Computing*, pp. 321-324, 2016.

[23] W. Ng and Z. Sun, "Multi-Layers LKH", *Proceedings of IEEE International Conference on Communications*, pp. 334-337, 2005.

[24] Q. Liu, Song Ningning, and Zeng Wenlu, "Research of Centralized Multicast Key Management for LEO Satellite Networks", *Proceedings of IEEE International Conference on Cyberspace Technology*, pp. 111-114, 2013

[25] X.B. Zhang, S.S. Lam, D.Y. Lee and Y.R. Yang, "Protocol Design for Scalable and Reliable Group Re-Keying", *IEEE/ACM Transactions on Networking*, Vol. 11, No. 6, pp. 908-922, 2001.

[26] S. Setia, S. Koussih, S. Jajodia and E. Harder, "A Scalable Group Re-Keying Approach for Secure Multi Cast", *Proceedings of IEEE Symposium on Security and Privacy*, pp. 221-224, 2000.

[27] Y. Kim, A. Perrig and G.Tsudik, "Group Key Agreement Efficient in Communication", Available at: http://www.ics.uci.edu/~gts/paps/jkpt02.pdf, Accessed on 2001.

[28] M. Hajyvahabzadeh, E. Eidkhani, S.A. Mortazavi and A.N. Pour, "A New Group Key Management Protocol using Code for Key Calculation: CKC", *Proceedings of International Conference on Information Science and Applications*, pp. 1-6, 2010.

[29] S. Zhao, R. Kent and A. Aggarwal, "A Key Management and Secure Routing Integrated Framework for Mobile Ad Hoc Networks", *Ad Hoc Networks*, Vol. 11, No. 3, pp. 1046-1061, 2013.

[30] Bin Tian, Yang Xin, and Shoushan Luo, "A Novel Key Management Method for Wireless Sensor Networks", *Proceedings of 3rd IEEE International Conference on Broadband Network and Multimedia Technology*, pp. 1-6, 2011.