REDUCING FREQUENCY OF GROUP REKEYING OPERATION

YunSuk Yeo, Sangwon Hyun and Tai-Myoung Chung

Computer Engineering,
Sungkyunkwan University, Suwon, South Korea

ABSTRACT

In the past, Ad-hoc networks were used in limited areas which require secure group communication without Internet access, such as the army or emergencies. However, Ad-hoc networks currently are widely used in variety applications like group chat, smart applications, research testbed etc. Ad-hoc network is basically group based network in the absence of access point so it is prevalent to provide group key approach to prevent information leakage. When we use group key approach, we need to consider which group key management method is the most suitable for the architecture because the cost and frequency of the rekeying operation remain as an unresolved issue. In this paper, we present analysis about existing group key management solutions for Ad-hoc network and suggest a new approach to reduce frequency of the rekeying operation.

KEYWORDS

Rekeying operation, Group key management, ad-hoc networks, Frequency of rekeying, Time-driven method

1. INTRODUCTION

There exist many group key management (GKM) methods to provide secure group communication. In general, two essential keys are used in the methods: a *traffic encryption key* (TEK) and a *key encryption key* (KEK). As illustrated in Fig. 1, all the group members share the same TEK, while each of them has a private KEK that is only shared with the group controller (GC) or key server (KS). When a group member wants to send a message M to the entire group, it broadcasts M encrypted with TEK, denoted by $\{M\}_{TEK}$.

Whenever a member leaves or joins the group, a new TEK must be generated and securely distributed to all the group members. Such process to update the TEK is commonly referred to as *rekeying*. A simple approach to rekeying is that the GC/KS unicasts each member the new TEK encrypted with the member's KEK so that the member can retrieve the new TEK by decrypting the received message with its KEK. However, this approach requires *O(n)* cost in terms of the number of messages, where *n* denotes the number of group members, and it is well known that this incurs "*broadcast storm*" problem; A broadcast storm occurs when a network system is overwhelmed by continuous multicast or broadcast traffic so it causes the whole network to melt down and lead to the failure of network communication. In addition, frequent rekeying in highly dynamic groups can also incur the broadcast storm problem.

Extensive research has been done to reduce the overhead of rekeying in two major research directions. The first category of research focuses on reducing the cost of each re-keying operation [1, 3, 5, 7, 9, 13, 14]. Especially, the local key hierarchy (LKH) method [1] reduces the cost into *O(log n)* by utilizing a binary key tree. However, in these approaches, the authors considered only the cost of rekeying without care about the frequency of rekeying would make their solution

Brajesh Kumar Kaushik et al. (Eds) : CCNET, CSIP, SCOM, DBDM - 2017
pp. 63– 72, 2017. © CS & IT-CSCP 2017 DOI : 10.5121/csit.2017.70507

incomplete so the solution cannot cope with dynamic and scalable groups. In ad-hoc network, very frequent rekeying operation will eventually cause a broadcast storm.

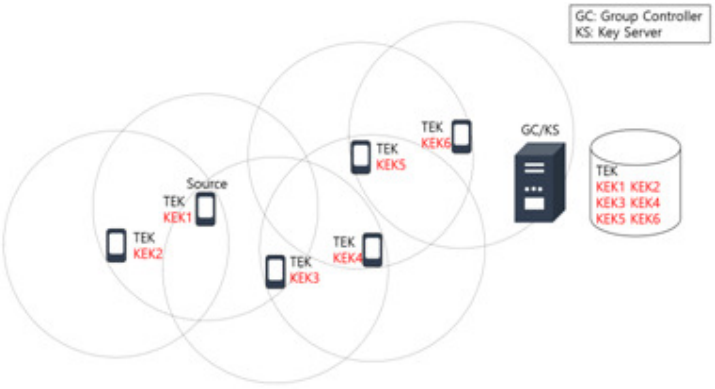

Figure 1. Ad-hoc Group Key Management

The second category of research focuses on reducing the frequency of re-keying [2, 4]. Especially, Kronos and Iolus adopt a time-driven method, which performs the rekeying operation periodically instead of performing it whenever member changes occur [2, 4]. The authors demonstrated that the time-driven method could considerably reduce the overhead of re-keying even though such limitations as no guarantee of forward secrecy and delays in adding new members. However, for highly dynamic groups, the cost of the time-driven re-keying could be *O(n)* as well [2]. In ad-hoc networks, O(n) cost usually causes "broadcast storm" situation due to centralized architecture (i.e., a GC/KS should generate *n* messages in worst-case). For these reasons, we cannot apply this method directly to GKM in ad-hoc networks.

In this paper, we suggest a composition of the time-driven method and GKMPAN method [3], which is one of the main GKM approaches for ad-hoc networks, to eliminate *O(n)* problem in the time-driven method. GKMPAN is novel method of distributed GKM in ad-hoc network so it can remove bottleneck and *O(n)* problems of the time-driven method. We modify the rekeying process of GKMPAN to facilitate the installation of the time-driven model in ad-hoc environment and to solve key exhaustion problem of GKMPAN. With this approach, we can achieve following works:

- Reduce the frequency and the cost of rekeying operation with reliable security strength
- Conduct rekeying operation without "broadcast storm" with distributed architecture
- Solve key exhaustion problem of GKMPAN

The organization of the remainder of this paper is as follows. In Section 2, we explain basic Local Key Hierarchy (LKH) [1], GKMPAN [3] and the cost analysis of the time-driven LKH method. In Section 3, we propose the expanded GKMPAN. Section 4 evaluates expanded GKMPAN and Section 5 present related works. Finally, Section 6 presents our conclusions and future works.

2. BACKGROUND

2.1. Local Key Hierarchy (LKH)

In the LKH method [1], a GC/KS constructs and manages a binary key tree *T* illustrated in Fig. 2. The root of *T* corresponds to the TEK. All the remaining nodes hold KEKs (= K_n) and a leaf node is associated with a group member (= U_i). A member U_i holds the TEK and all the KEKs associated with the nodes from K_i to the root. For instance, U_8 holds K_8, K_{78}, K_{5678}, and TEK.

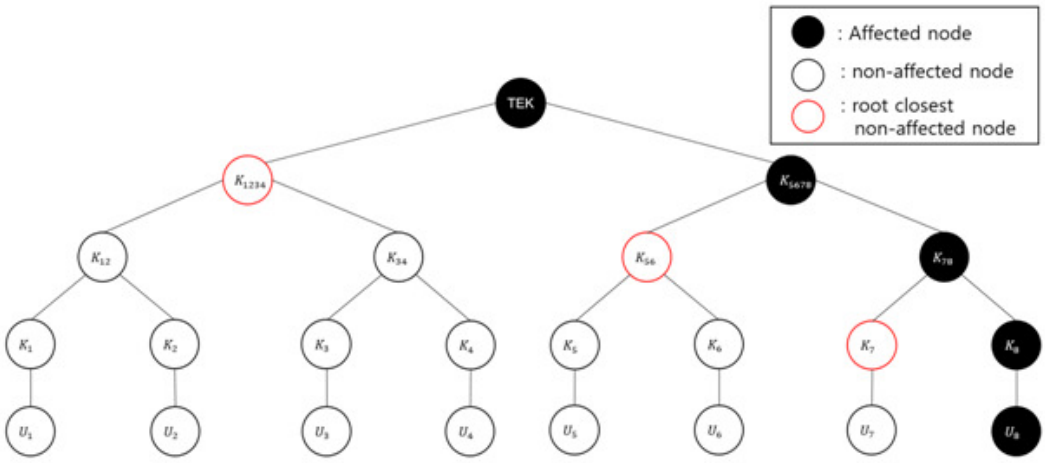

Figure 2. LKH Leaving Example

If U_8 leaves the group, then K_8, K_{78}, K_{5678}, and TEK should be changed to ensure forward secrecy. We call the nodes whose key must be changed as "affected nodes." To change these keys, the GC/KS only needs to send three different types of rekeying messages to the groups; $\{TEK\}_{K1234}$ for the members (U_1, U_2, U_3 and U_4), $\{TEK\}_{K56}$ for the members (U_5 and U_6), and $\{TEK\}_{K7}$ for the member (U_7).

If U_8 joins the group, then K_8, K_{78}, K_{5678}, and new TEK should be generated to ensure backward secrecy. The GC constructs and sends rekey messages to the users. A rekey message contains one or more encrypted new key(s), and a user needs to decrypt it with appropriate keys in order to get the new keys.

By using the binary key tree, the LKH scheme effectively reduces the cost of rekeying from $O(n)$ to $O(log\ (n))$. We have applied cost-effective LKH scheme to effectively apply time-driven to ad-hoc networks and evaluated the cost.

2.1.1. Cost Evaluation of Time-Driven LKH

In Fig. 2, we count K_{1234}, K_{56}, and K_7 to calculate the cost of rekeying process because each group member can securely obtain a new group key by encrypting/decrypting message with the key of the root nodes of sub-trees composed of non-affected nodes; K_{1234}, K_{56}, and K_7. Similarly, we can calculate rekeying cost of time-driven method with LKH by counting the root nodes of sub-trees composed of non-affected nodes with breadth-first search (BFS). These root nodes are referred to as root of subtree (RS). Considering the time-driven case, we denote the number of members whose state changed within a period as m and the size of the group as n. For the simplicity, we assume that leaving and entering events are occurred in the same frequency, besides m and n are in the form of the power of two: 2, 8, and 64.

The Fig. 3 shows how to calculate the maximum number of rekeying messages in time-driven method. In time-driven method, there are many leaving and entering events during a cycle, so the tree has multiple affected branches. If a member U_i leaves the group, ($log\ n$) nodes are affected, because a member U_i has all the keys along the path from its leaf to the root. To get the maximum cost of rekeying, we treat the case that leaving nodes do not overlap, as is possible in the tree. In this situation, the rows up to a height of $log\ m$ from the root have fewer nodes than m in that rows; therefore, they do not have any unaffected nodes, because m nodes leave the group and a

leaving node follows one path from the leaf to the root in the tree *T*. In other words, all the nodes in the rows below a height of *log m* are affected by leaving or entering events.

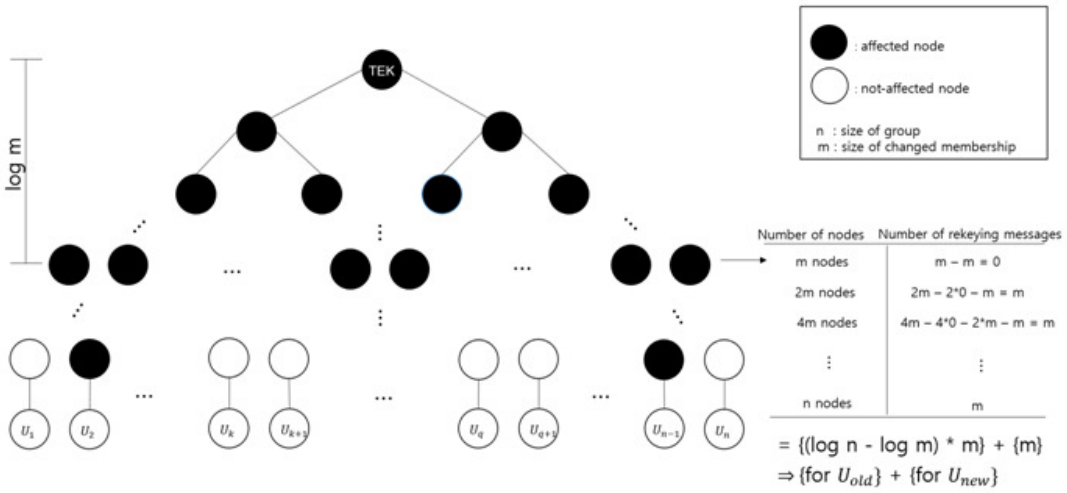

Figure 3. Maximum number of rekeying messages in the time-driven approach

Now, it is possible to find the RS nodes in each row by subtracting both the affected nodes and those already covered by the elements from parent RS nodes.

$$RS = E_{row} - E_{affected} - E_{covered\_list} \text{ (E: node Element)}$$

Once the RS nodes in a row have been found, the child nodes of them are inserted into the covered list, which is used to find the RS nodes of the next row. This sequence of operations is described right-below edge in the Fig. 3. For entering events, additional *m* messages are needed because each newbie should respectively receive its secure KEK. Hence, the total number of rekeying messages is

$$\text{Cost} = (\log n - \log m) \times m + m.$$

With this formula, if $n = 1024$ and $m = 16$, the number of rekeying messages exceeds 100 during a cycle, which means that a change of only 2% will cause a "broadcast storm" due to centralized model. The general equation for any form of *m*, which is not exactly a power of 2, is

$$\text{Cost} = (\log n - \text{floor}(\log m)) * m + m + 2 * (2^{\text{floor}(\log m)} - m).$$

2.2. GKMPAN

In GKMPAN [3], all node obtains a distinct subset of keys out of a large key pool from the Key Server (KS) and these keys are used as Key Encryption Keys (KEKs) for delivering group keys (TEK). This method has the advantage in that we can ignore the process of securely sharing KEKs between a GC and new member because it is already shared in offline mechanism. Moreover, this model uses hop-by-hop approach to eliminate bottleneck problem.

When a leaving event occurs, the GC broadcasts the list of revoked keys possessed by a leaving node, so that this revoked keys will not be used anymore in this group. The GC generates and sends a new TEK encrypted with K_m, the key is possessed by the maximum number of remaining nodes, to ensure forward security. With this broadcast message, however, some of the nodes cannot decrypt it because they do not contain K_m in their private key list. To fully deliver new TEK to its child nodes, the GC should resend the new TEK encrypted with other keys which are shared between GC and some of the excluded nodes based on a delivery tree.

When a node receives new TEK encrypted with K_m or other KEKs, then, it continuously sends new TEK to its child nodes in a manner like the GC along the delivery tree. Each node only cares the child nodes like an independent key server. Then, we can infer that the maximum overhead of each node is the number of child nodes of itself on the delivery tree.

We can apply the time-driven method to this approach because it eliminates the bottleneck problem, which could be found in the centralized model [5]; however, there is a problem that the number of available keys (KEKs) decreases as the number of removed nodes increases. GKMPAN [3] asserts that the problem can be solved using big l and small m. However, for performance, we should choose small *l* and large *m*, because if a small *l* and a large m are used, a key is shared with many nodes so that the node can share the new TEK with a very small number of messages

The biggest difference between GKMPAN and our method is that GKMAPN uses the pre-allocated KEKs to share the new TEK and we use the old TEK that was used in the previous rekeying stage. The advantage of our approach is that the dependency between performance and key exhaustion is eliminated. We excluded the keys held by the revoked nodes only during the rekeying stage of one cycle. When node *i* leaves the group, we can use the remaining keys in the rekeying stage, except for the keys that node *i* had in the key pool. However, in the next rekeying stage, the keys that were excluded can also be used.

3. OUR APPROACH (EXTENDED GKMPAN)

Table 1. Terminology

GC/KS	Group Controller/Key Server
KEK	Key Encryption Key
TEK	Transmission Encryption Key (group key)
l	Size of a key pool
K_m	The key is possessed by the maximum number of remaining nodes in network
m	The number of keys in the possession of a group member (= remaining node)
K_b	The key owned by node b
L_{leave}	List of leaving node at current stage
L_{join}	List of joining node at current stage

To solve the *O(n)* and bottleneck problem in the time-driven method, we propose a new rekeying strategy based on GKMPAN [3], which uses a pre-distributed set of keys. The main idea of our approach is to securely pass a new TEK encrypted with the old TEK "hop-by-hop"; $\{TEK\}_{old\_TEK}$. Before we explain details, the Table 1 describes terminologies which are used in last of the paper.

3.1. Model Description (Extended GKMPAN)

As described above, our model uses time-driven enabled GKMPAN approach to complement weak point of time-driven method. Our model is also hop-by-hop process so we assume that each node can easily acquire the information about the nodes within its one-hop distance. We also assume all group members have pre-distributed set of keys from key pool *l*, which will be used as KEKs. With the GKMPAN method, it is possible to know which keys a member possesses with only the id of the member. Because of this, a node can figure out which keys the surrounding nodes have. Moreover, asymmetric key mechanism is used, so the sender only needs to know the

public key of a key of the receiver. In this approach, we do not need to worry about which keys overlap between the sender and receiver because the all members have capability to encrypt a message with any key in *l* (i.e., a sender knows all the public keys of all the KEKs).

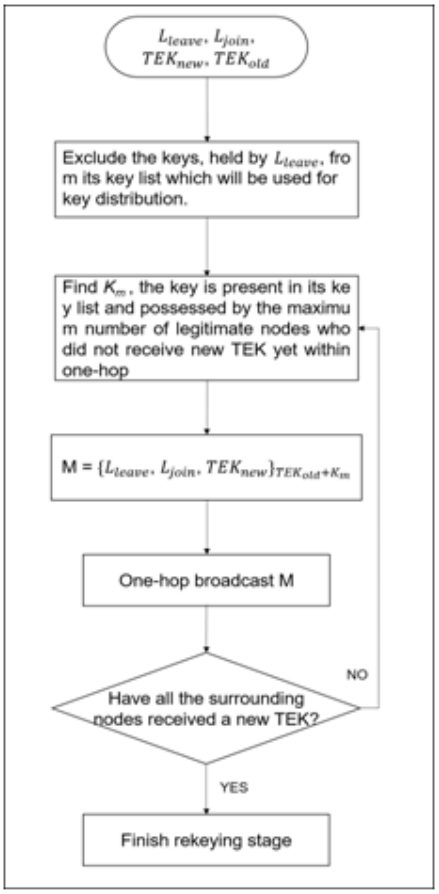

Figure 4. Flow Chart of a Rekeying Stage in Extended GKMPAN

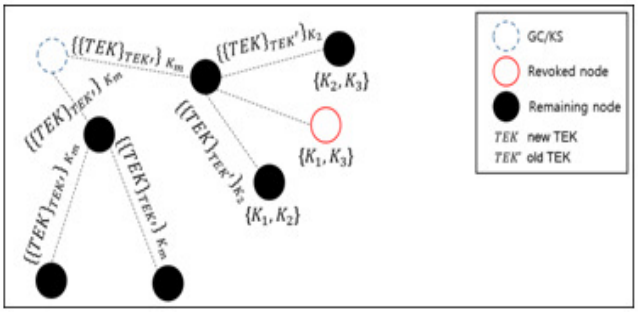

Figure 5. Hop-by-hop Encryption

After completing one cycle of the time-driven method, the GC obtains both a list of revoked members and newly joined members list. Then, the rekeying process starts and the GC securely sends a new group key (TEK), the revoked list, and the newly joined list to its surrounding nodes. The node receiving the rekeying message becomes the distributor and starts the process represented in Fig. 4. The legitimate nodes contain remaining nodes and the members in L_{join}.

As depicted in the Fig. 4, the first step of secure rekeying process is to figure out which keys can be used to distribute new group key by excluding the keys held by L_{leave}. The second step is to find K_m that the key is possessed by the maximum number of legitimate nodes who did not receive new TEK yet within one-hop. After finding K_m, the distributer sends a rekeying message e ncrypted with both the old TEK and K_m to the all nodes within one hop, $\{L_{leave}, L_{join}, TEK_{new}\}_{TEK_{old} + K_m}$. If all the neighboring nodes have received a new TEK, the rekeying stage ends. If not, the distributer starts again from the second step by finding K_m, which is possessed by the maximum number of legitimate nodes who did not receive new TEK yet within one-hop.

The role of K_m is to hide the new TEK from the nodes that are revoked at current stage (i.e., L_{leave}), because they know the old TEK. K_m is selected from the keys possessed by legitimate nodes except for the revoked nodes so it can hide new TEK from the revoked nodes.

The reason for encrypting the new TEK with the old TEK is to solve the key exhaustion problem of GKMPAN. The nodes without knowledge of the old TEK are unable to retrieve the new TEK; thus, it is unnecessary to consider whether previously eliminated nodes have some keys in their key pool because they do not have knowledge about the old TEK. So, we only need to care about the nodes which are revoked at current cycle (i.e., The KEKs that were excluded from the rekeying stage can be used in the next cycle).

For instance, in the rekeying stage, node v uses the old TEK and K_b to share a new group key. K_b is owned by node b which was revoked previously. Then, although revoked node b knows the K_b, it cannot decrypt the new group key encrypted with both the K_b and the old TEK (i.e., $\{TEK_{new}\}_{TEK_{old} + K_b}$) because it does not know the old TEK which is used in current time-driven cycle. From this point of view, we do not need to delete the K_b from the key pool even though the node b leaved the group. It means that the lifetime of the keys in the key pool is semi-permanent.

This approach has several advantages. First, in the best case each node of a group needs to send only one multicast message to propagate the new TEK and in the worst case each node is required to send a message to each of the surrounding legitimate nodes (i.e., remaining node + L_{join}). Second, this approach enables us to select a large l and small m, because we have already solved the problem of key exhaustion. By encrypting the new TEK with the old TEK, the keys owned by each group member are simply used to hide the new TEK from the nodes who have left current rekeying stage.

The diagram in the Fig. 5 illustrates a simple case. In short, the GC/KS sends a new TEK encrypted with the old TEK to surrounding nodes, then they subsequently forward the new TEK to their neighbors securely by encrypting it with K_m, which is K_2 in this example. Here, we do not consider other processes, such as key distribution or message verification, because these properties can be referenced from GKMPAN [3].

4. PERFORMANCE EVALUATION

We evaluated our proposed method by calculating the number of messages a node v should send to share a new group key in four different cases. In the Fig. 6, n, l, and m, respectively, denote the number of surrounding nodes of a node v, the size of the key pool, and the number of keys possessed by a node. In evaluation, we assume that there is no collision. Each node evenly has random leaving and joining probability (0.1 to 0.9). We count all messages in the group and compute average message counts from a node v. Moreover, in our and GKMPAN approach, some nodes may not get a new TEK because the all keys possessed by a node could be excluded from current rekeying stage due to the revoked nodes. So, we also evaluate failure ratio according to n, l and m.

In Fig. 6, The red line shows the failure ratio based on the right-side y axis, and the black line shows the number of required messages based on the left-side y axis. As shown in a and b, they depict the number of messages and failure ratio when the number of neighbor nodes (n) and the size of the key pool (l) are fixed. When m is small, the number of necessary messages decreases as the value of m increases. However, as the value of m increases from a specific value, the number of required messages increases. In c and d, they depict the number of messages and failure ratio when the number of neighbor nodes (n) and the key set size of each node (m) are

fixed. The results of *c* and *d* are similar to those of *a* and *b*. As the value of *l* increases before the specific value, the number of necessary messages decreases, but it increases thereafter.

For a large *m* and small *l*, the failure ratio is small, and for a small *m* and large *l*, the failure ratio is large. In addition, a high probability of failure ratio means that legitimated nodes are less likely to have the same keys, and this result can be seen in the above experiment.

The efficiency of the algorithm depends on how many legitimate nodes have the same keys. If *m* is sufficiently small compared to *l*, we can see that the number of keys needed is reduced because the number of keys that overlap each other increases. If *m* is large enough for *l*, the number of keys that a node has is large, so there are more keys to be excluded for revoked nodes in the key pool. This reduces the probability that the legitimated nodes will have the same key, which makes the efficiency worse.

In the same simulation environment, when the population is 100, it is shown that a central node of the time-driven LKH sends an average of 72 ~ 75 messages, and in case of 200, it sends close to 150 messages. It means (n * 3/4) messages are required to share a TEK in time-driven LKH in normal cases. LKH is affected by the total number of nodes, but our approach is much more effective than LKH because it is only affected by the number of neighbor nodes, not the whole.

5. RELATED WORKS

In Ad-hoc networks, we can categorize the GKMs into three sub groups; Centralized approach, Decentralized approach, and Distributed approach. In Centralized approach, there is only one GC/KS. This server is responsible for the generation, the distribution and the renewal of the group key (TEK). Centralized approach is suffered from the *O(n)* problem and bottleneck problem. GKMPAN [3], CKDS [12], Kaya et al. solution [13] belong this group. In Decentralized approach, it divides the group into several sub groups to relieve *O(n)* problem and bottleneck problem. However, this approach requires several decryption and re-encryption operations of multicast message, when it passes from a sub-group to another. ILOUS [4], AKMP [9], and BLADE [5] belong to this group.

The one of Decentralized approaches is BLADE [5]. Its basic idea is to divide the multicast group dynamically into clusters. Each cluster is managed by a local controller which shares with its local members a local cluster key. The multicast flow is encrypted by the source with the traffic encryption key TEK and sent in multicast to all the group members. The source of the group and the local controllers from a multicast GLC (Group of Local Controllers) share beforehand a session key called KEK_{CCL}. Each new local controller has to join this group and receive the session key KEK_{CCL} from the source of the group, encrypted with its public key. It could support mobility and insure energy efficiency.

In Distributed approach, all group members cooperate and generate TEK, to establish secure communications between them. This approach eliminates the bottleneck problem, but generally it suffered from *O(n)* problem. Our approach is also included in this category without *O(n)* problem with time-driven method. Several papers propose the solution for this category [9, 10, 14, 15, 16]. Chiang et al. [16] proposes a GKM protocol for MANETs based on GPS measures and on the GDH (Group Diffie Hellman) [11]. In this protocol, during protocol initialization, each node in the ad hoc network, floods its GPS information and its public key to all the others nodes, although the authors assume that the protocol does not rely on any certification authority. Using the GPS information received from others nodes, each group member can build the network topology. When a source wants to multicast the data flow to the group members, it computes the minimum multicast tree, based on the Prűfer algorithm and then sends message with GDH keys.

6. CONCLUSIONS

The Ubiquitous Network Society suggests a world in which many ad-hoc environments can be applied. Moreover, the number of applications, such as Vehicular ad hoc networks (VANETs) or Smartphone ad hoc networks (SPANs), relying on the ad-hoc multicast technique is increasing. However, no efficient GKM method has been suggested yet so, in this paper, we propose an improved GKM scheme with the aim of enhancing the efficiency GKM in Ad-hoc network. The dominant factor of our scheme is reduced frequency of rekeying operation, whereas other schemes were more concerned with the cost. Considering the feasibility of GKM, we believe our method is more practical than others. In the future, we devise modified version of extended GKMPAN to apply this to wired networks as distributed GKM.

ACKNOWLEDGEMENTS

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.2016-0-00078, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning).

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2010-0020210).

REFERENCES

[1] C.K. Wong, M. Gouda, and S. S. Lam, Secure Group Communications Using Key Graphs, ACM SIGCOMM (1998).

[2] S. Setia, S. Koussih, S. Jajodia, and E. Harder, Kronos: A Scalable Group Re-Keying Approch for Secure Multicast, IEEE Symposium on Security and Privacy, 2000.

[3] Sencun Zhu, Sanjeev Setia, Shouhuai Xu, Sushil Jajodia, GKMPAN: An Efficient Group Rekeying Scheme for Secure Multicast in Ad-Hoc Networks, IEEE, 2004.

[4] Suvo Mittra, Iolus: A Framework for Scalable Secure Multicasting, ACM Sigcomm, 1997.

[5] Mohaned-Salah Bouassida, Isabelle Chrisment, and Olivier Festor, Group Key Management in MANETs, International Journal of Network Security, Vol.6, No.1, PP.67-79, Jan. 2008.

[6] Yacine Challal, Hamida Seba, Group Key Management Protocols: A Novel Taxonomy, International Journal of Information Technology, Vol.2, 2005.

[7] Yan Sun, Wade Trappe, and K. J. Ray Liu, A Scalable Multicast Key Management Scheme for Heterogeneous Wireless Networks, IEEE/ACM, Vol. 12, No 4, 2004

[8] W.T. Li, C.H. Ling, M.S. Hwang, Group Rekeying in Wireless Sensor Networks, International Journal of Network Security, Vol, 16, No.6, 2014

[9] H. Bettahar, A. Bouabdallah, and Y. Challal, "An adaptive key management protocol for secure multicast," in 11th International Conference on Computer Communications and Networks ICCCN, Florida USA, Oct. 2002.

[10] W. Diffie and M.E. Hellman,"New directions in cryptography," IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644-654, 1976.

[11] I. Ingemarson, D. Tang, and C. Wong, "A conference key distribution system," IEEE Transactions on Information Theory, pp. 714-720, Sep. 1982.

[12] M. Moharrun, R. Mukkalamala, and M. Eltoweissy, "Ckds: An efficient combinatorial key distribution scheme for wireless Ad Hoc networks," in IEEE International Conference on Performance, Computing and Communications (IPCCC'04), pp. 631-636, Apr. 2004.

[13] T. Kaya, G. Lin, G. Noubir, and A. Yilmaz, "Secure multicast groups on Ad Hoc networks," in Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks, pp. 94-102, 2003.

[14] S. Rahul, and Hansdah. "An Efficient Distributed Group Key Management Algorithm". In Proceedings Tenth International Conference on Parallel and Distributed Systems, 2004. ICPADS 2004, pages 230 -237, California.

[15] L.R. Dondeti, S. Mukherjee, and A. Samal. "Comparison of Hierarchical Key Distribution Schemes". IEEE Globcom Global Internet Symposium, 1999.

[16] T. Chiang and Y. Huang, "Group keys and the multicast security in Ad Hoc networks," in Proceedings of the 2003 International Conference on Parallel Processing Workshops, pp. 385, 2003.