Design a MAC address based Authentication Protocol for Efficient Group Key Agreement

Babloo Kumar, Research Scholar[1] JNIT, Jaipur.
bablook61@gmail.com

Pranam Goyal[2], Associate Prof. SBNIT, Jaipur.
gpranam@gmail.com

Abstract- *The authentication protocols based on classic cryptography use public-key cryptosystems for establishing the common key and some of them have been proven to be secure but they require high amount of resources and they need large keys. This kind of protocols is secure enough even if a small key is used this is another advantage. In this paper we presents secure cryptographic authenticated group key transfer agreement and performance evolution of MAC addresses based protocol for distributed network environment. The protocol is developed for group communication and every member of the group has secret information and the communication start when all this information is put together. So, without online two members can't communicate to each others. We, also, present some situations where this kind of protocol is needed. It's well known that in wireless LAN, authenticating nodes by their MAC addresses is secure since it's not easy for an attacker to learn one of the authorized addresses and change his MAC address accordingly. In order to the MAC address prevent spoofing attacks, we propose to use dynamically changing MAC addresses and make each address usable for only one session. The scheme we propose does not require any change in 802.11 protocols and incurs only a small performance overhead. One of the nice features of our new scheme is that no third party can link different communication sessions of the same user by monitoring MAC addresses therefore our scheme is preferable also with respect to user privacy.*

Keywords: Authentication protocol, MAC addresses, Cryptosystem, Group key transfer protocol.

I.INTRODUCTION

MAC addresses have long been used as the singularly unique layer to network identifier in LANs. Through controlled, the organizationally unique identifiers (OUI) allocated to hardware manufacturers, globally unique MAC addresses for all LAN-based devices use in many cases today., the MAC address of a workstation is used as a unique identifier or as an authentication factor for granting varying levels of network or system privilege to a user. This method of client tracking and authentication is also employed in wireless 802.11 networks. Wireless LANs targeted by attacker's utilization of the ability to change their MAC address to circumvent network security measures: an attacker with minute skill might alter their MAC address in an effort to masquerade or hide their presence, MAC address to one that is otherwise authorized to bypass access control lists or to escalate network privileges [1]. In this paper, I demonstrate two methods of detecting wireless LAN MAC address spoofing. I also show how these methods can be used to detect the activity of devious WLAN attack tools.

1.1 Changing MAC Addresses

The phrase "MAC address spoofing" in this context relates to an attacker altering the manufacturer-assigned any other value of MAC addresses. This is different conceptually than traditional IP address spoofing where an attacker sends data from an arbitrary source address and does not expect to see a response to their actual source IP address. The spoofing of MAC address might be more accurately described as MAC address "impersonating" or "masquerading" since the data is not crafted by attacker with a different source than is their transmitting address. They continue to utilize the wireless card for transport purpose to its intended layer, transmitting and receiving from the same MAC source When an attacker changes their MAC address [2-3]. All 802.11 cards in use permit their MAC addresses to be altered, often with full drivers and support from the manufacturer. By using the Linux open-source drivers, MAC address can be changed by users with the ifconfig tool, or calling the ioctl() function with the SIOCSIFHWADDR flag with a short C program. Commonly windows users are permitted to change their MAC address by properties of their network card drivers selecting in the network control panel applet. An attacker may choose to alter their MAC address for several reasons, their presence including obfuscating on a network, to bypass access control lists, or to abbreviate an already authenticated user. Each is explained in greater detail as follows:

A. Obfuscating network presence

An attacker might choose to change their MAC address in an attempt to evade network intrusion detection systems (NIDS). A common example is an attacker executing a brute- force attack script with a random

MAC address for each successive connection attempt. Such an attack would go undetected by network activity analysis applications such as Net Flow that report upper-layer network activity or large quantities of traffic from a single source address.

B. **Bypassing access control lists**

Used as a basic form of access control on WLANs, administrators typically have the option to configure access points or neighboring routers to permit only registered MAC addresses to communicate on the network. An attacker could circumvent this form of access control by passively generate a list of MAC addresses and monitoring the network that are authorized to communicate., an attacker is free to set their MAC address to any of the authorized addresses, bypassing the intended security mechanism with the list of authorized MAC addresses in hand [4-5].

C. Authenticated user impersonation

Certain hardware WLAN security authentication devices rely on matching user authentication credentials to the source MAC address of a client. After a successfully authenticated user, the security gateway permits traffic based on a dynamic list of the authorized MAC addresses [6, 7, and 8].

II. Proposed Related Work

Group communication arises in many different settings, from low-level network groupware, and other multicasting to conferencing applications. Security services are needed to provide integrity and communication privacy. These services are not possible without a secure and efficient key distribution, authentication, and other mechanisms. In a secure communication, group members need a common group key to protect their messages exchanged as well as group key management for the distribution and computation of the GK., delivery of messages over the network to the right destination cannot be guaranteed unless the co/mmunication channel is secure. Group key management is a building block to provide such assurance. There are two types of schemes in group key management, group key distribution and group key agreement [9]. The group key distribution is assigned to one member in the group who then becomes the key distribution center. He/she computes the GK and distributes it to each member in the group. The group key agreement is suitable for peer-to-peer group communication [9]. In these groups the group key agreement protocol ensures that each member has an equal opportunity for generating the GK. One member takes the role of the Group Controller (GC), collects all the members' blind keys (public keys), broadcasts the group key computation tree structure to all members, and controls the overall group key computational processes.

III. Distributed Group Key Distribution (DGKD): a new class of GKM Protocols

A. Principle and assumption

There are some assumptions in existing schemes. In CGKD/DGKM, a secure channel is assumed to exist between the GC/SC and each of the potential group members/subgroup members. This secure channel is generally implemented by public key cryptosystems. In CGKA, which is typically based Die-Hellman key exchange which users from the Man-in-the-Middle attack, it is assumed that each group member is equipped with some authentication capability which is also implemented by public key cryptosystems. Similarly, DGKD assumes that every group member has a publicly known (unforgivable) public key. The new DGKD protocol adopts a tree structure and utilizes three basic mechanisms to implement distributed key generation and distribution: 1) the leaf key of a node is the public key of the corresponding group member and all the intermediate nodes' keys are secret keys, 2) the sponsor of a joining or leaving member initiates the key generation and rekeying process and sends the new keys to co distributors (i.e., the first round), 3) the co-distributors then help distribute the new keys to group members in a distributed/parallel manner (i.e., the second round). All group members have the equally trusted and same capability is. Also, they have equal responsibility, i.e. any group member could be a potential sponsor of other members or a co-distributor (depending on the relative locations of the member and the joining/leaving members in the tree). Thus there is no dependence on a single entity and even if a sponsor node fails a new sponsor for the joining/leaving member is chosen by other members. This improves the robustness of the protocol [10, 11, and 12].

B. Sponsor

A sponsor is a member and the sponsor of a sub tree is defined as the member hosted on the rightmost leaf in the sub tree (note: \rightmost" can be equally replaced with \leftmost"). Every node has an associated sponsor field as shown in Figure 1. The sponsor field at a particular node is updated when it is along the joining or leaving member's path. We show the joining algorithm for updating the sponsor field in Figures 2. When a member joins, the sponsor field along the joining member's path is updated from bottom to the root. If the new member's id is greater than the sponsor id of the node then update the sponsor id with the new member's id. This is continued until the root (See Figure 3). When m7 joins, the sponsor field along its path is up dated. The sponsor id of the node k6-7 is lesser than the id of m7, so it is updated to 111. Similarly the sponsor id's of nodes

k4-7 and k0-7 are updated to 111. Whenever the Sponsor id for a node is greater than

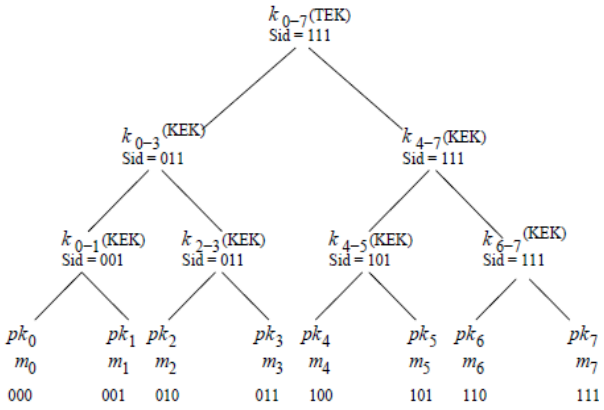

Fig.1. sponsor for each node showing by a tree.

the joining member's id then the check can be stopped. When a member leaves, every member checks along the path of the leaving member to update the sponsor lD. if a node has

Every member

- the joining members iterate over all the nodes
  Path from leaf to the root
- if the id is greater than the sponsor id for that node joining members
- sponsor id = members id joining

-continue

- else

-break the leaving member as the sponsor then they update the sponsored ld with the sponsor id/member id of the other child if exists. This continues up to the root (See Figure 4). When m7 leaves, the sponsored ld along its path is updated. Since the leaving member is the sponsor all along its path, the sponsor ld has to be updated by checking for the new sponsor for all the nodes, m6 becomes node k6-7 for the new sponsor. For node k4-7 the member ids of both its children are compared and the new sponsor greater becomes, in this case m6. This is continues until the root [13-14].

C. Co-distributors

When a sponsor changes the keys along the path, it needs to distribute them. The sponsor has to distribute the keys to all the members whose keys have been changed. But it does not know the keys along the other paths to distribute the new keys. So, a co distributor is required to distribute them. The co distributor is the sponsor of a node on another path whose key is not known to the original sponsor. The sponsor encrypts the changed key with the co distributor's public key and broadcasts this information. Thus, the co-distributor helps the sponsor in distributing the changed common keys along the other paths.

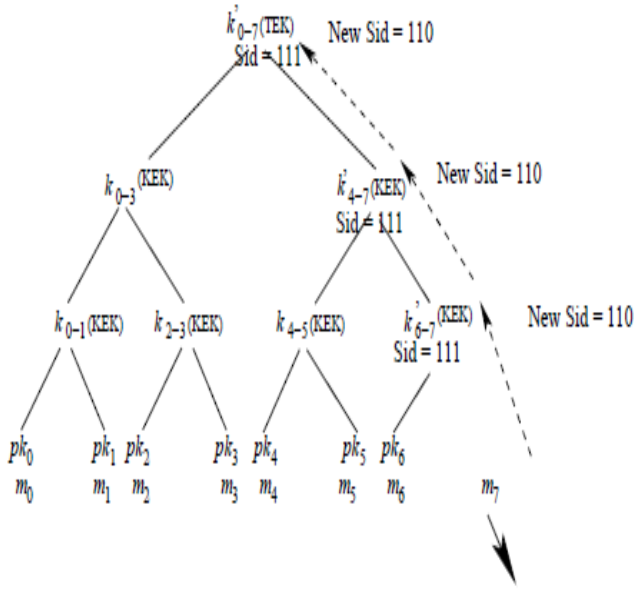

Fig.2. when a member joins field then updating the sponsor.

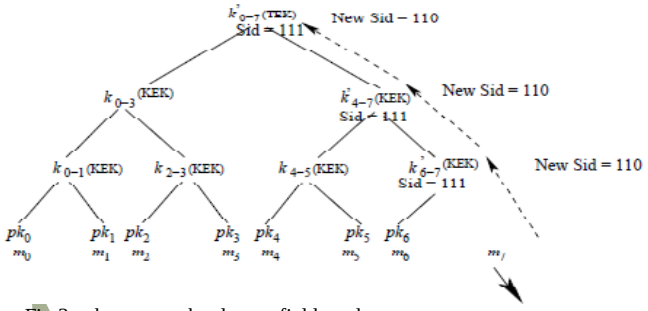

Fig.3. when a member leaves field, updates a sponsor.

D. Initial group key generation and distribution Protocol

Suppose n members m1... mn decide to form a group. They build a virtual key tree and select a sponsor to decide an order in which they join the tree. Every member updates the key tree by adding members in the key tree based on that order and they update the sponsor field in all the intermediate nodes. Then every member checks if it is responsible for generating any keys along its path. If so, it generates them and distributes the keys either directly or with the help of co-distributors. When two sponsors are responsible for generating the same key then the rightmost among them generates it. As more members join the key tree the sponsors and the height of the key tree increase. As illustrated in Figure 4, m7, m5, m3 and m1 are responsible for generating the keys.m7 generates the entire key (k6-7, k4-7, and k0-7) along its path to the root. Then it encrypt it as $\{k6-7, k4-7, k0-7\}pk6$, an where k4-5 is generated by m3 in the left sub tree along its path and the

root key which is generated by the right most sponsor m7 is sent to the co-distributor of the left sub tree m3 as follows. $\{k0-7\}pk3$

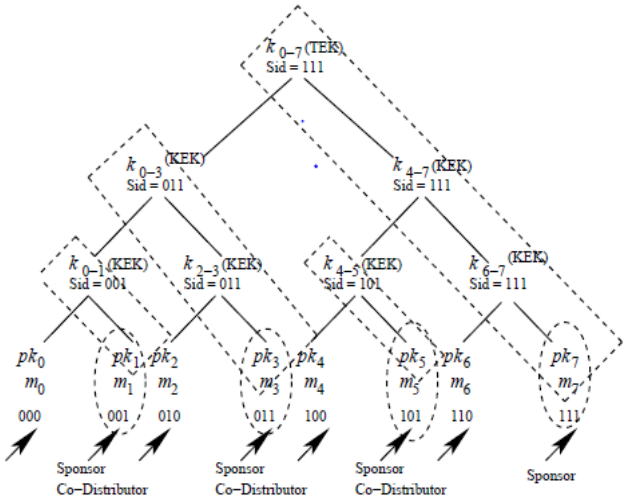

Fig.4. Example of Initial key generation

is broadcast and m3 will decrypt k0-7 and encrypts it as $\{k0-7\}k0-3$ and broadcasts it. Thus every member has the newly generated keys along its path. Only two rounds are required for this protocol, one round for generating keys and distributing along the path and another for co distributors to distribute them.

E. Join protocol

Step.1-- New member broadcasts request for join
Mn+1(PKn+1) →m1,…mn

Step.2--Each member
- adding a new member node for updates the key tree

Find for joining sponsor member:
- if sibling present, sponsor =sibling
- else sponsor = mn+1
- field along the path of the joining member to the root if required is updated by the sponsor.

Step.3-- If joining member's sponsor is it self
- generates new secret keys along the joining member's path and distribute them to other members co-distributors and to directly by encrypting with common key and broadcasting

Step.4--If co-distributor is itself
- Joining members sponsor with appropriate key and Broadcast sent encrypt the key

Suppose there are n members in the group m1,.....,mn. a new member mn+1 make a join request by broadcasting its public key PK. The rightmost member in the key tree authenticates the new member, decides the insertion location for the new member and broadcasts this information to other members. Additionally the rightmost member also sends the virtual key tree and list of public keys of other members to the new member. All other members update the key tree by adding a new member node in the specified location. Then every member checks to see if it is the sponsor of the joining member[15-16]. If the new member has a sibling it becomes the sponsor and generates new keys along the path. If there is no sibling then the joining member itself becomes the sponsor and generates the new keys along its path and distributes them. Members update the sponsor field appropriately if required. Figure 6 describes the join protocol and Figure 5 shows the protocol operation when a new member joins. When a new member joins, m7 determines the position (i.e., m5) and places the member there. m7 broadcasts the position of the new member to other members. All members also determine that m4 is the sponsor of m5. So the rekeying process m4 initiates as follows: 1) generates new keys k'4-5, k'4-7and K'0-7. 2) after determining the co-distributors m3 and m7, encrypts as follows and broadcasts: $\{k'4-7, k'0-7\}pk7$

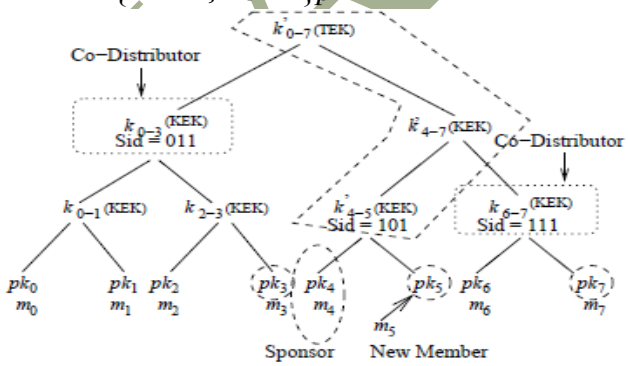

Fig.5. A new member joins m5, m4 is sponsor and m3, m7 are co-distributors.

$\{k'0-7\}pk3$ 3). m3 will decrypt k'0-7 and encrypt it as $\{k'0-7\}k0-3$ and m7 will decrypt k'4-7 and k'0-7 and encrypt them as $\{k'4-7\}k6-7$ and $\{k'0-7\}k4-7$ 4). m4 also encrypts and sends the keys to m5 as $\{k'4-5, k'4-7, k'0-7\}pk5$. As a result, all the members will get the new keys. When a new member joins, only the keys along its path to the root have to be changed and distributed, which can be achieved in two rounds with at most log2n keys being changed.

F. Leave protocol

Step.1-- Each member
- removing the leaving member node to updates the key tree
- Appropriately along the leaving member's path if required updates the sponsor field
-determination of the sponsor for changing keys along the leaving member's path

Step.2-- the leaving member is itself sponsored
- new secret keys generates to distributes them and to co-distributors and along the path directly to other members

Step 3: If co-distributor is itself : broadcasts the key sent by the leaving members Sponsor by encrypting it with the appropriate key

Assume that member ml leaves the group. Every member updates the key tree by deleting node ml and updates the sponsor field along the path if required. Then they determine the sponsor who generates new keys along the leaving member's path and distributes them [16-17]. If the leaving member does not have a sibling then the first sponsor along the leaving member's path becomes responsible for changing the keys along the leaving member's path AS fig. 6

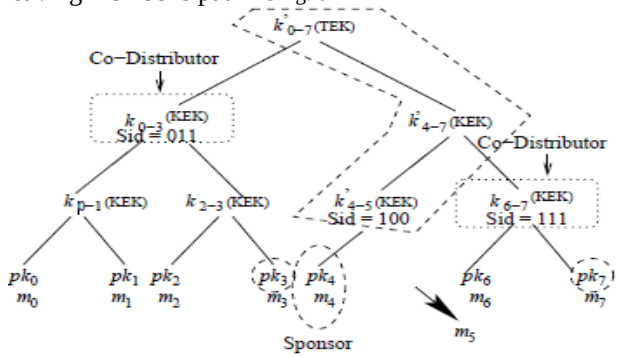

Fig.6. A member m5 leaves.

As shown in Figure 6, when a member m5 leaves, all the members will remove the node and determine that m4 is the sponsor of m5. When a member leaves only the keys along its path to the root have to be changed and distributed, which can be achieved in two rounds with at most log2n keys being changed.

G. Multiple joins protocol

Suppose m new members join, they make a join request by broadcasting their public keys. The rightmost member in the key tree authenticates the new members, decides the locations for all the new members such that minimal number of keys is changed and broadcasts this information to other existing group members. The rightmost member also sends the virtual key tree and existing members public keys to the joining members. Every member upon receiving this message updates its key tree by adding m new nodes in the determined positions [18]. In order to perform multiple joins in one aggregate operation, it is required to find the common keys shared by the joining in efficient way to the members. To achieve that we use an already proposed scheme, an efficient and scalable key tree based dynamic conferencing scheme called KTDC in [19] which uses an efficient algorithm for computing the shared keys. There will be multiple sponsors responsible for changing the necessary keys. But here the shared keys which both sponsors have in common and which need to be changed will be changed by the rightmost sponsor among the sponsors.

Step.1--Each member
- New member nodes updates key tree by adding
- all the paths of updates along the sponsor field Joining members
- need to be changed the computes keys
- who are responsible for changing these keys determines the sponsors

Step.2-- one of the joining members is itself for sponsor
-changes the secret keys along the joining member's path and distributes them to other members and directly to the co-distributors
-, check if right sponsor is itself if same key has to be changed
-change the key and distribute if rightmost sponsor

Step.3--If the itself co-distributor
- the joining members sent broadcasts the key sponsor by encrypting it with the appropriate key As shown in Figure 7, when new members join, m7 will determine

the available positions (i.e., m0, m1, m4, m5) and place the members there. m7 broadcasts this information to other group members. All members also know that m5 is the sponsor of m4 and m1 is the sponsor ofm0. They also

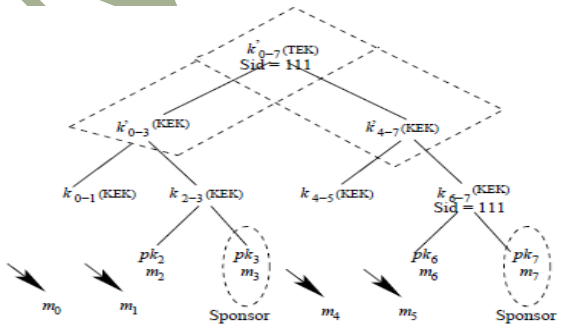

Fig.7 joins new member's m0, m1, m4 and m5.

know that m3 and m7 are responsible for sending the key tree structure and the public key list to the joining members. Since all the operations are done in parallel, rekeying can be achieved in two rounds by all the sponsors. When a network event causes all the previously occurred partitions to reconnect this is called a merge. Merge is similar to multiple join and this can also be achieved in two rounds which is better than that in TGDH.

H. Multiple leaves protocol

Step.1--Each member
- by removing all leaving member nodes updates the

updates the key tree key tree
- the sponsor field along all the leaving members paths updates if required
- the sponsors determines to responsible for changing the Keys along the paths

Step.2--If sponsor for one of the leaving member is itself -generates new secret keys and distributes them to Co-distributors and other members directly
-if same key has to be changed, check if right sponsor is itself
- if rightmost sponsor, change the key and distribute

Step.3--If co-distributor is itself When multiple members leave, every member updates its key tree by deleting those member nodes and the sponsor fields along all the paths. Then they determine the keys that need to be changed and the sponsors responsible for changing those keys. There will be multiple sponsors and each sponsor regenerates the keys and distributes them. If two sponsors are responsible for changing the same key then the rightmost among the sponsors will change the key : broadcasts the key sent by the leaving members sponsor by encrypting it with the common key when several members m0, m1, m4 and m5 leave, every member

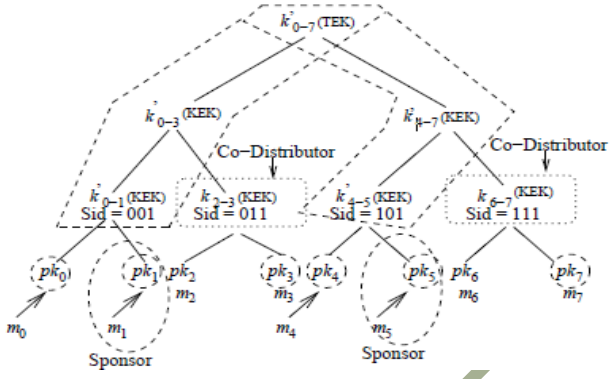

Fig.8. m0, m1, m4 and m5 leave Members

updates its key tree by deleting those member nodes. Every member also determines that m3 and m7 are the sponsors. In case of a network failure which causes disconnectivity, the group gets split and this partition can be dealt with as a multiple leave operation [20-21]. Thus, even for network partition the protocol requires only two rounds for regenerating and distributing the keys. This is a great improvement compared to TGDH which requires several rounds.

I. Authentication in DGKD

Most CGKA protocols do not contain an authentication component. Furthermore, the authenticated CGKA protocols are non-dynamic and/or non-scalable. In contrast, the new DGKD protocol is not only scalable and dynamic but also able to provide easy and strong authentication.

IV. Discussions

We discuss the performance and security of our protocol in this section and analyze the computation and communication costs for leave, multiple join, join and multiple leave operations. Tree based Group Di_e-Hellman (TGDH) is one of the most typical CGKA protocols in terms of efficiency and scalability, so we focus on the comparison between DGKD and TGDH. Key generation is independent, i.e., only the sponsor is involved, thus there is no need for synchronization with other members which is required in TGDH. In this sense, DGKD is more resilient to network congestion, delay and failure than TGDH. DGKD also has strong yet simple authentication. It is also collusion free because the new keys are independent of the old keys and no matter how many members collude they cannot get the keys [20]. Thus, it is unconditionally secure. Both TGDH and DGKD require two rounds for single join and leave operations. As for multiple join and leaving operations, DGKD requires two rounds but TGDH requires log(p) rounds where p is the number of members involved. DGKD uses public key encryption for sending the keys to co-distributors and secret key encryption for further distribution of keys (from the co distributors to the members). TGDH requires performing modular exponentiations which is in the same complexity as the public key encryption. In summary, DGKD is comparable and in some cases better than TGDH in terms of communication and computation costs [21].

V. Conclusion

We proposed a new class of GKM protocols for SGC with strong yet simple capability of authentication. The proposed protocol solves some serious problems in the existing protocols and is simple, scalable, efficient and robust. The future work is to test and the implement new protocol.

Reference

[1], C.-Y. Chou, J.C Lin, F. Lai, and K. P. Wu, “A distributed key management protocol for dynamic groups," 27th Annual IEEE Conference on Local Computer Networks, pp. 0113-0122, Nov. 2002

[2], D. Liu ,P. Ning, and K. Sun, “Cryptographic protocols/ networks ecurity: Enceinte self-healing group key distribution with revocation capability" Proceedings of the 10th ACM conference on Computer and communication security, pp. 231-240, Oct. 2003.

[3] D. Hutchison and S. Rafaeli, “Hydra: A decentralized group key management" Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enter prises, pp. 62-67, June 2002.

[4] Y. Desmedt and M. Burmester, "Ancient and secure conference key distribution" Security Protocols Workshop, pp. 119-129, 1996.
[5] L. R. Dondeti, "Enceinte private group communication over public networks," Phd. Dissertation, CSE UNL, 1999.
[6] I. Ingemarsson, D. Tang and C. Wong, "A conference key distribution system," IEEE Transactions on Information Theory, vol. 28, pp. 714-720, Sept. 1982.
[7] D. Steer, L. Strawczynski, M. Wiener, and W. Di_e, "A secure audio teleconference system," Advances in Cryptology CRYPTO'88, LNCS, Springer-Verlag, vol. 403, pp. 520-528, Aug. 1990.
[8] G. Tsudik, M. Steiner and M.Waidner, "Di_e-Hellman key distribution extended to group communication," ACM Conference on Computer and Communications Security, pp. 31-37, Mar. 1996.
[9] Y. Kim, A. Perrig, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups" In Proceedings of the 7th ACM Conference on Computer and Communications Security, pp. 235-244, Nov. 2000.
[10] A. Perrig, Y. Kim, and G. Tsudik, "Communication-enceinte group key agreement," In Information System Security, Proceedings of the 17th International Information Security Conference IFIP SEC'01, pp. 229-244, June 2001.
[11] A. Bakkardie, "Scalable multicast key distribution," RFC 1949, 1996.
[12] B. Chor and A. Beimel, "Interaction in key distribution schemes," Advances in Cryptology - CRYPTO'93, LNCS, Springer, Berlin, vol. 773, pp. 444-457, 1994.
[13] A. Beimel and B. Chor, "Communications in key distribution schemes," IEEE Transactions on Information Theory, vol. 42, pp. 19-28, 1996
[14] R. Blom, "An optimal class of symmetric key generation systems," Advances in Cryptology - EUROCRYPT'84, LNCS, Springer, Berlin, vol. 209, pp. 335-338, 1985.
[15] A. Cresti and C. Blundo, "Space requirements for broadcast encryption," Advances in Cryptology EUROCRYPT'94, LNCS, Springer, Berlin, vol. 950, pp. 287-298, 1995.
[16] C. Blundo, L. A. F. Mattos, and D. R. Stinson, "Generalized Beimel-Chor scheme for broadcast encryption and interactive key distribution," Theoretical Computer Science, vol. 200, pp. 313-334, June 1998
[17] A. Herzberg, C. Blundo, A. D. Santis S. Kutten, U. Vaccaro, and M. Yung, "Perfect secure key distribution for dynamic conferences," Advances in Cryptology - CRYPTO'92, LNCS, Springer, Berlin, vol. 740, pp. 471-486, Aug. 1993.
[18] Y. Desmedt and M. Burmester, "A secure and enceinte conference key distribution system" Advances in Cryptology EUROCRYPT'94, LNCS, Springer, Berlin, vol. 950, pp. 275-286, May 1995.
[19] I. F. Bob Briscoe, "Nark: receiver-based multicast non repudiation and key management" Proceedings of the 1st ACM conference on Electronic commerce, pp. 22-30, Nov. 1999.
[20] H. Bettahar, Y. Challal, and A. Bouabdallah, "Sakm: a scalable and adaptive key management approach for multicast communications," ACM SIGCOMM Computer Communication Review, vol. 34, pp. 55-70, Apr. 2004.
[21] W. Chen and L. R. Dondeti, "Recommendations in using group key management algorithms," DARPA Information Survivability Conference and Exposition, vol. 2, pp. 222-227, Apr. 2003

Authors Profile:

Mr. Pranam Goyal was born in Bikaner, Rajasthan, India, in 1973. He received the M. Tech. degree in Computer Science and Engineering and the Ph.D. degree submitted at Birla Institute of Technology, Mesra, Ranchi in 2011.Since 2009, he has been an Associate Professor and Head of Department at Shri Bhawani Niketan Institute of Technology and Management, Jaipur. His research areas are in Knowledge Management Technologies.

Babloo Kumar was born in meerut,utter pradesh, in 1979. He received the B.Tech degree in computer engineering from UP Technical University, in 2005, and currently pursuing M.Tech. degree from the jagannath university. His research areas are innetwork security.