Multicast Security Model for Internet of Things Based on Context Awareness

Hussein Harb, Ashraf William , Omayma A. El-Mohsen
Switching Department
National Telecommunication Institute
Cairo, Egypt

Hala A. Mansour
Faculty of Engineering
Benha University
Cairo, Egypt

***Abstract*—Internet of Things (IoT) devices are resource constrained devices in terms of power, memory, bandwidth, and processing. On the other hand, multicast communication is considered more efficient in group oriented applications compared to unicast communication as transmission takes place using fewer resources. That is why many of IoT applications rely on multicast in their transmission. This multicast traffic need to be secured specially for critical applications involving actuators control. Securing multicast traffic by itself is cumbersome as it requires an efficient and scalable Group Key Management (GKM) protocol. In case of IoT, the situation is more difficult because of the dynamic nature of IoT scenarios.**

This paper introduces a solution based on using context aware security server accompanied with a group of key servers to efficiently distribute group encryption keys to IoT devices in order to secure the multicast sessions. The proposed solution is evaluated relative to the Logical Key Hierarchy (LKH) protocol. The comparison shows that the proposed scheme efficiently reduces the load on the key servers. Moreover, the key storage cost on both members and key servers is reduced.

Keywords—Internet of Things; Group Key Management, Context Awareness.

I. INTRODUCTION

The Internet of Things (IoT) refers to billions of interconnected smart objects equipped with sensors or actuators. Existing and evolving communication protocols are used to enable services to such smart objects and allow them to communicate among each other and with backend users of the Internet during different activities of sensing and controlling. IoT networks, without intelligence, are just wireless sensor networks. Intelligence in IoT is achieved using context awareness. Context awareness, as defined in [1], is the ability of a system to provide information or services to users using information of a certain entity where the entity can be a person, place, piece of software, or an object.

On the other hand, the IoT is naturally a resource-constrained network regarding CPU power, memory, and energy. Therefore, many of the IoT application scenarios rely on multicast operation to preserve bandwidth and enhance the sensing and control operation among the sensors and the actuators. Multicast transmits data efficiently between one sender or multiple senders to multiple receivers. However, the constrained nature and the massive size of the IoT network make it vulnerable to many security attacks.

In order to multicast information among a certain group securely, the traffic should be encrypted. So a common group key should be shared among all members of the group. Whenever membership changes, the group key should be updated. Hence, during the registration process, it is necessary to have strong authentication mechanisms to acquire the identity of the participants prior to distributing the key material. Thus, the main concern is around key management, key distribution, and access control for the key material [2].

Group Key Management (GKM) protocols are divided into three categories: centralized, decentralized, and distributed key management protocols [2][3][4]. Nevertheless, all the conventional GKM protocols under the above mentioned categories don't suit the dynamic nature of the IoT scenarios and applications. Consequently, many research directions are carried out to adapt these protocols to IoT networks [5][6][7]. Yet, most of the research related to GKM focuses on adapting these protocols by working on just one or two of the IoT aspects such as mobility, scalability, constrained nature of devices, application nature, network access technology, or addressing. They ignore that the majority of IoT scenarios need to work on most of these aspects combined together and they lack the ability to address the resulting issues of such combination.

In this paper, a new GKM security model is proposed; CASMA (Context-Aware Secured Multicast Architecture); to work with different IoT applications and to suite the dynamic nature of the IoT scenarios. CASMA relies on adding intelligence to the security operation by using a Context Aware Security Server (CASS) that is responsible for managing the secure multicast session and group key distribution operation. The key distribution operation itself is carried out through a set of Key Distribution Servers (KDS). The CASS collects context information from both members (sensors or actuators) and KDSs, and analyses it to assign those members to the appropriate KDS that will be able to serve them best, while controlling load balancing which improves both performance and scalability.

The solution advantages can be summarized in the following:

- CASMA is an open model that is not tied to a specific protocol or algorithm and can deploy any of the current or future protocols or algorithms.
- The allocation process is based on context-awareness, which improves both performance and scalability.

978-1-5386-4266-5/17/$31.00 ©2017 IEEE

- It can be deployed for both private and public application scenarios.
- It provides load balancing control among key servers
- It overcomes key servers' failover.

The rest of the paper is organized as follows: Section II describes background information related to the secured multicast operation and the protocols used to achieve it along with an overview of the current related research directions concerning GKM in IoT scenarios. Section III covers the proposed GKM model architecture and operation. Section VI presents a mathematical evaluation and analysis of the proposed GKM model performance. Finally, section V concludes the paper and proposes future research directions.

II. Background and Related work

A. Secured Multicast Operation

In order to secure multicast traffic, a security policy is set up by Group Owner (GO) and passed to a trusted entity; the Group Controller and Key server (GCKS), which will manage the group security operation. Next, the group members register with the GCKS. Their registration is accepted only if they meet the security parameters defined in the security policy set up by the GO. At this time, group members are eligible for keys download.

The aim of such process is that all authorized users download the same key that is used to encrypt and decrypt multicast data messages. This shared key is called Traffic Encryption Key (TEK). In order to preserve the secrecy of the multicast data and to achieve forward and backward secrecy, the TEK needs to be updated periodically or each time a member joins/leaves the multicast group according to the rekeying mechanism defined in the group security policy [2][3][4]. Each authorized member that shares the TEK may need to download additional Keys, called Key Encryption Key (KEK) and Group Encryption Key (GEK), which facilitates the update of the TEK.

B. Group Key Management Protocols

Protecting group information is achieved by defining the security policy by the GO and enforcing it among group members. This is accomplished by using protocols such as Group Security Association Key Management Protocol (GSAKMP) [8] or Group Domain of Interoperation (GDOI) [9]. The GO creates the security policy rules for the group and expresses them in a policy token that is passed to the GCKS. The GCKS enforces such a policy by granting access only to members that fulfill the security policy. In this case, members will have the right to download the group keys. Whenever membership changes, the keys need to be updated according to the mechanism defined in the security policy. In general, protocols used in group key management update can be classified into centralized, decentralized, and distributed group management protocols.

In centralized group key management protocols, there is only one GCKS responsible for group key distribution and update. The GCKS shares a pairwise key with each member of the group and distributes group keys to group members on a point-to-point basis. Examples of protocols working in centralized fashion are Group Key Management Protocol (GKMP), Logical Key Hierarchy (LKH), One-way Function Tree (OFT), Centralized Flat Table (CFT), and Efficient Large-Group Key (ELK) [2][3][4].

In decentralized group key management protocols, the large group is divided into small subgroups. A group key is shared among all group members and every subgroup has its own subgroup key. The GCKS serves all members of the group. On the other hand, every subgroup has its subgroup key sever, which manages the subgroup key. Examples of protocols working in decentralized fashion are Scalable Multicast Key Distribution (SMKD), Iolus, Dual-Encryption Protocol (DEP), MARKS, Kronos, and Intra-Domain Group Key Management protocol (IGKMP) [2][3][4].

In distributed group management protocols, many members in the group are responsible for new group key generation and distribution. It has no group controller. The group key can be generated in a contributory fashion, where all members aid in generating the group key. However, this process becomes difficult as group members increase or if the key is generated by one of the members, which is not secure. The distributed group key management protocols are the most complex and difficult ones. Examples of distributed protocols are Group Diffie–Hellman Key Exchange (G-DH), Conference Key Agreement (CKA), Distributed Logical Key Hierarchy (DLKH), Distributed One-way Function Tree (DOFT), and Diffie–Hellman Logical Key Hierarchy (DH-LKH) [2][3][4].

C. Related Work

Most of the research work carried out in this domain is either very specific to an application or a rekeying protocol. For example, Li et al in [5] focuses on Privacy Preservation in Smart Buildings of the Smart Grid. It is based on Tree Group Deffie-Helman (TGDH) evaluating fault tolerance and performance. The paper assumed that the key server and the trust center are always available; it doesn't state how security is maintained in case of their failure.

Similarly Agrawal in [6] works on Secure Key Distribution Protocols in Smart Meter application focusing on preventing man-in-the-middle attack. The paper introduces just a security analysis with no performance evaluation.

In [7] a centralized approach to distribute and manage group keys in ad hoc networks and Internet of Things is used. The proposal is applied to Vehicle-to-Vehicle communications in Vehicular Ad hoc Networks measuring the communication cost. The paper proposes performing batch leave operation based on a pre-determined leave time stated by members while joining the group. The paper assumes that each member knows the exact time to leave the group, which is not always the case. Also, the paper ignores leave events due to communication loss.

It is obvious that most of the research work carried out is tied to a certain type of application such as smart grids, internet of vehicles …etc., none of it is generic. In addition, those researches work on just one or two aspects of IoT assuming that the conditions of the application scenario is static which is not the case for IoT application scenarios which

have dynamic and varying nature regarding the network access technology, type of application, state of members and key servers and the load on them.

This raises the motivation to introduce our GKM model that adapts to the dynamic nature of IoT scenarios regarding application nature and scenario conditions.

III. The Proposed GKM Model

This section presents the new proposed solution. In subsection A, the new key management architecture, CASMA, is introduced. Next, subsection B explains its operation.

D. CASMA Architecture

The CASMA model is based on the IGKMP protocol architecture [2][3][4][10]. IGKMP is a decentralized key management protocol in which subgroups are called areas. Each area has one subgroup controller called Area Key Distributer (AKD). A common controller; Domain Key Distributer (DKD), is responsible for generating the group key and facilitating the co-ordination between AKDs. Each member belongs to one area only and registers with one AKD according to the location.

The CASMA architecture is shown in Fig. 1. CASMA follows the ITU IoT architecture reference model defined in [11]. According to the model, the architecture is composed of a Device layer, Network layer, Service Support and Application Support layer, and finally an Application layer.

CASMA is based on using a CASS server and a set of KDSs. Here, The KDSs are the DKD and the ZKDs. The DKD; as in IGKMP; is responsible for TEK generation and update, while the ZKDs replace the AKDs found in IGKMP. The ZKDs are the subgroup controller of their zones, where the zone represents the ZKD with its registered members. They are not tied to a certain area or location as is the case for AKDs. All these servers lie in the Service Support and Application Support Layer of the IoT Reference Model as shown in Fig. 1.

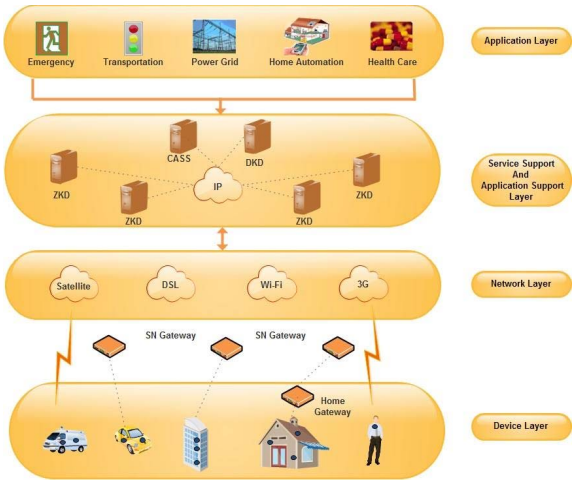

Fig. 1 CASMA Architecture

The CASS server acts as the trusted entity that is responsible for managing the security sessions, while the KDSs are responsible for key distribution and update.

The CASS collects context information from both members (sensors or actuators) and KDSs and stores it in its database. The information collected from sensors includes location, access network, multicast group to join …etc. On the other hand, the information from KDSs is that related to the supported multicast groups and the number of members associated with each group. The CASS server analyses the collected information and uses it to assign those members to the appropriate ZKD that will be able to serve them best, while controlling load balancing. This information is periodically updated. Therefore, the allocation process is based on intelligent context-awareness, which improves both performance and scalability.

CASS roles can be defined as follows:

- Acts as a Trusted Entity for security operation.
- Manages the operation of key distribution.
- Contains profile for context information of registered member (status, location, access network, multicast application running).
- Contains profile for context information of ZKDs (status, location, multicast groups supported, number of registered members in each group).
- Collects context information from members and KDSs and stores it. This information includes:
    - Access Network (Wi-Fi, Zigbee, 3G …etc.)
    - Geographical location
    - Multicast group to join
    - Load on ZKD servers
- Analyses collected information and uses it to assign the members to the appropriate KDS that will be able to serve them best, while controlling load balancing.
- Receives members' query-to-join requests.
- Assigns members (actuator nodes) to the appropriate ZKD to register with according to analyzed context information.
- Keeps track of active ZKDs.
- Assigns alternative ZKD to members in case of the main ZKD failure.
- Adjusts periodic rekey time according to application and network conditions.

ZKD roles are as follows:

- Communicates with CASS Server, DKD, and members.
- Receives request to join messages from members.
- Each ZKD Creates a zone that contains itself and the members that register with it.
- Shares a unique key; Member-Private-Key (MbrPKey) with each accompanied member. This key is used to provide a secure unicast channel between the ZKD and the member.
- Shares a common Zone group Key (ZKey) between itself and all accompanied members. This key is used to assist TEK distribution from the ZKD to the associated members.
- The MbrPKey and the ZKey are both generated and downloaded from the ZKD to the members during the registration process.

- Shares a unique key ZKDkey with the DKD. This key is used to provide secure unicast channel between the ZKD and the DKD.
- Shares a common Domain group Key (DGkey) between the DKD and all ZKDs. This is used to assist in the distribution of TEK from the DKD to all ZKDs.
- The ZKDKey and the DGKey are both generated and downloaded from the DKD to the ZKDs during the registration process between ZKD and DKD.
- Updates members with TEK during rekeying.

DKD roles are defined as follows:

- Communicates with the CASS Server and the ZKDs.
- Shares the unique ZKDkey with each ZKD.
- Shares the common multicast DGkey with all ZKDs.
- Generates the TEK and transmit it to the ZKDs over a secured channel.
- Notified by the CASS in case of membership change or periodic rekeying.
- Updates TEK according to membership change or periodic rekeying.
- Keeps track of key synchronization among ZKDs.

It is worth mentioning that within each zone, rekeying can take place using the above mentioned MbrPKey and Zkey. It can also take place using any of the centralized rekeying protocols mentioned in section II-B (LKH, LKH+, OFT, CFT, and ELK). This solution can be implemented in either private application scenarios such as buildings automation, smart home, and buildings safety or public application scenarios such as in smart cities and Internet of vehicle applications.

E. CASMA operation

The following explains the rekeying operation in CASMA solution. This process takes place whether periodically or due to member change.

Member Join:

With every new member joining the group, the group key needs to be changed in order to assure backward secrecy (new members have no access to previous secured data). The messages flow is shown in Fig. 2.

First, the new member sends a "Query-to-Join" request to the CASS server. The CASS server analyses its database and selects the appropriate ZKD according to the scenario logic. The address of the assigned ZKD is sent to the joining member in the "Query-Response" message.

Second, the member sends a "Request-to-Join" message to the assigned ZKD. The ZKD authenticates the joining member and ensures that it meets the security policy specification. If it succeeds, the ZKD sends an "Accept-Join-Notification" message to the CASS server to update its database with the context information related to the ZKD (number of accompanied members relative to the multicast group) and to the member (successfully registered with the assigned ZKD).

Third, the ZKD sends a "Join-Notify" message to the DKD to update the TEK. Consequently, the DKD performs key update and multicasts the new key to all ZKDs encrypted with the DGkey. Finally, the ZKDs extract the TEK and multicast it to their zone members encrypted using their Zone key. Simultaneously, the ZKD with the new joining member updates its ZKey and sends the new keys (TEK and ZKey) to the new member encrypted with its private key (MbrPKey).

Member Leave:

When a member leaves a group, the multicast key and its zone key need to be updated to assure forward secrecy (the leaving member has no access to forthcoming data transmitted). The messages flow is shown in Fig. 3.

First, the leaving member sends a "Leave" request to its ZKD. Second, The ZKD forwards the “Leave” request to the DKD to perform TEK update. Third, the ZKD sends a “Leave_Notification” message to the CASS server to update its context information database. Fourth, the ZKD creates a new ZKey and sends this key to each one of the remaining zone members encrypted by each member private key (MbrPKey). Fifth, the DKD creates the new TEK and multicasts it to all ZKDs encrypted by the DGkey. Finally, each ZKD decrypts this message carrying the TEK, re-encrypts it using its ZKey, and multicasts it to its zone members.

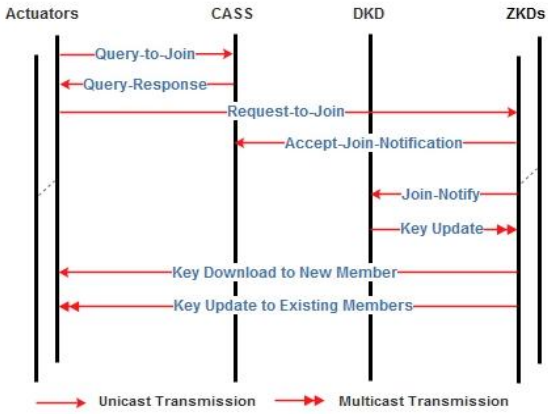

Fig.2 Join Multicast Group Message Flow

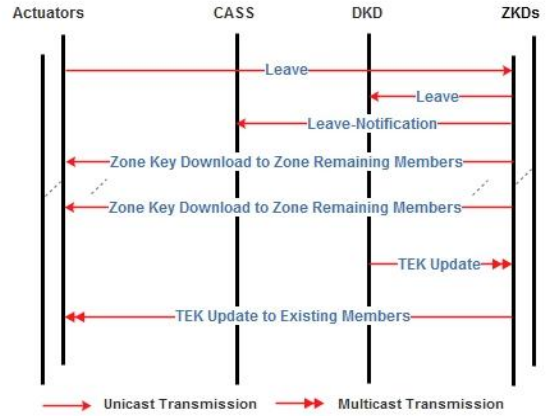

Fig.3 Leave Multicast Group Message Flow

Periodic Rekeying:

The CASS server periodically informs the DKD server to update the Keys to avoid security breaches. By keeping track of the server status, the CASS informs the DKD about the

appropriate time for rekeying to avoid network and server loading. Furthermore, the CASS can adjust the time interval needed for periodic rekeying according to application type and strength of the encryption key.

IV. Performance Analysis

As previously mentioned in section III-A, rekeying within each zone can take place using one of the centralized key management protocols. Consequently and in order to evaluate the performance of CASMA, a comparison is held between two cases. In the first one, LKH is used in the traditional centralized way. In the second case, LKH protocol is used for rekeying with CASMA. .

The LKH protocol is chosen for comparison as it is considered one of the most efficient rekeying protocols. The operation of LKH begins by building a hierarchal key tree with members acting as leaves and the GKCS acting as the root of the tree. Each node in the tree represents a key in the key hierarchy. The root key is the TEK used to secure the group data. The leaf key is the unique key shared between the node and the key server. The intermediate nodes represent keys that are a set of KEKs used to help in the distribution and update of TEK. According to the tree structure, each member stores all the keys in the path from itself to the root. With members change, all the keys in its path need to be changed to maintain forward and backward secrecy [12].

In our evaluation, the LKH key tree is built as a balanced binary tree. The context information that is used here by the CASS server for members' distribution among ZKDs is the load on these servers. The members register with the first ZKD until a threshold level is reached. Next, the members will begin to register with the next ZKD. The threshold level used in this evaluation is sixteen. In a balanced binary tree, this threshold gives rise to five keys to be stored in each member per multicast group. This is considered appropriate for such constrained devices.

The evaluation studies two critical parameters for multicast security in IoT: storage cost in members and key servers and communication cost during the join and leave operations.

F. Storage Cost

Storage cost is defined as the number of keys stored in each member and in the key servers.

Member Keys:

According to the LKH protocol, each member stores $\log_2(n)+1$ keys, where n represents the number of members [12]. In our proposal, each zone implements LKH. Accordingly, the number of keys stored in each member is $\log_2(n_{mz})+2$, where n_{mz} represents the number of members in each zone. This number originates from the fact that, each zone member stores $\log_2(n_{mz})+1$ keys according to LKH (which covers the individual MbrPKey and Zkey). Additionally, each member stores the global TEK.

Fig. 4 shows the members storage cost for both traditional LKH and CASMA. In traditional LKH, as the number of members increases, the number of stored keys also increases. In our proposal, the members are distributed among many ZKDs with a maximum number of sixteen members per zone. This leads to a maximum of six keys to be stored per zone member. As a result, much less keys are stored in members using CASMA compared to the traditional LKH protocol.

Server Keys:

In traditional LKH, there is only one key server that stores 2n-1 keys [12]. In CASMA, there are two types of servers: the DKD and the ZKDs. For the ZKD, the number of keys stored is $2n_{mz}+2$. This number originates from implementing LKH within the zone, which gives rise to $2n_{mz}-1$ stored keys in each ZKD. Additionally, each ZKD stores three extra keys; the Zkey, the ZKDkey, and the TEK. Adding these three keys to the existing 2n-1 keys results in $2n_{mz}+2$ keys stored in each ZKD. For the DKD, the number of keys stored is n_z+2, where n_z is the number of ZKDs. This number represents the individual ZKDkey of each ZKD in addition to the DGkey and TEK.

Fig. 5 shows the key servers storage cost. For the single key server in traditional LKH, the storage load increases linearly with the addition of new members. For CASMA, the load is nearly constant. This is due to the distribution of members among several servers. CASMA is clearly much more efficient as it balances the load among the servers and avoids overloading a single server. Hence, it outperforms in terms of scalability. Furthermore, members are re-assigned to other servers in case their primarily assigned server fails providing resilience and redundancy.

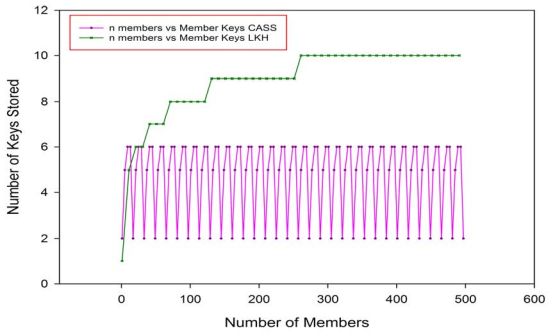

Fig.4 Multicast Group Members Storage Cost

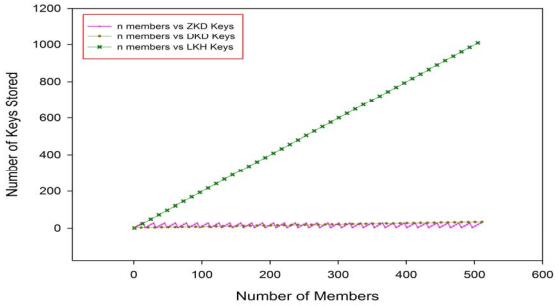

Fig.5 Multicast Group Key Servers Storage Cost

G. Communication Cost

Communication cost is defined as the number of required rekeying messages sent by the key servers during the join and leave operations.

Communication Cost during Join operation:

In LKH, the number of join messages is $2*\log_2(n)$ with single key distribution per message [12]. On the other hand, when a new member joins the group in CASMA, the DKD multicasts the new TEK to all ZKDs in one message. In the unaffected zones, the ZKD multicasts this TEK to its zone members in only one message as well. For the zone with the new joining member, the ZKD updates the other members with the new ZKey according to LKH with $2*\log_2(n_{mz})$ messages. Additionally, it multicasts the new TEK to all zone members using the new ZKey. Therefore, the total number of messages transmitted by this ZKD is $2*\log_2(n_{mz})+1$.

The number of Join messages versus the number of members is shown in Fig. 6 for both CASMA and traditional LKH. The number of messages in CASMA does not exceed 9, which is less than half the number in LHK. CASMA surpasses traditional LKH in accommodating large groups of members and is hence more scalable.

Communication Cost during Leave operation:

For a binary tree and a single key distribution per message, the number of leave messages in LKH is $(2*\log_2(n))-1$ messages [12]. In CASMA, when a member in a certain zone leaves the group, the ZKD in this zone updates the ZKey of its tree using $(2*\log_2(n_{mz}))-1$ messages. Next, the DKD multicasts the new TEK to all ZKDs using one message. Finally, all ZKDs multicast the new TEK to their zone members in one message too. This includes the ZKD with the leaving member. As a result, the total number of messages transmitted by the ZKD with the leaving member equals $(2*\log_2(n_{mz}))$.

Fig. 7 shows the communication cost versus the number of members for both CASMA and traditional LKH. Once again, the number of messages in CASMA is less than half the number in LHK. The new proposal surpasses traditional LKH in scalability.

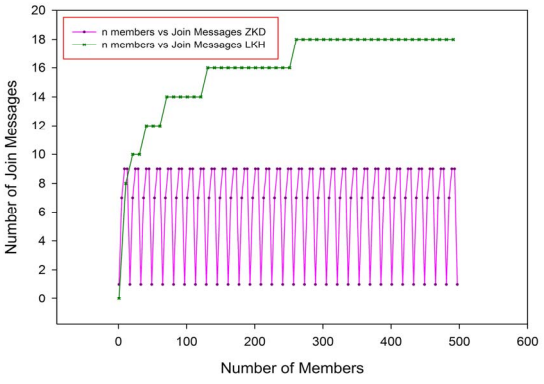

Fig.6 Join Messages Communication Cost

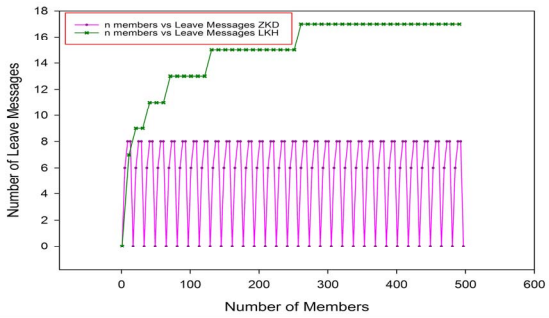

Fig.7 Leave Messages Communication Cost

V. Conlusion and Future Work

This paper introduces a new GKM security model; CASMA. The model deals with the dynamic nature of IoT application scenarios using a context aware security server. This server assigns members to the appropriate key server to register with and obtain the keying materials based on collected context information. The proposal is evaluated by measuring and comparing both communication and storage costs of CASMA to the traditional LKH protocol. The evaluation shows a significant improvement in performance, scalability, resilience, and redundancy when CASMA is used instead of traditional LKH.

References

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness", Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, London, UK, 1999, pp. 304–307.

[2] Raju Barskar and Meenu Chawla, "A Survey on Efficient Group Key Management Schemes in Wireless Networks", Indian Journal of Science and Technology, Vol 9(14), April 2016.

[3] Bibo Jiang, Xiulin Hu, "A Survey of Group Key Management," International Conference on Computer Science and Software Engineering, 2008.

[4] Sandro Rafaeli, David Hutchison, "A Survey of Key Management for Secure Group Communication", Journal ACM Computing Surveys (CSUR), Volume 35 Issue 3, Pages 309-329, September 2003.

[5] Depeng Li, Zeyar Aung, Srinivas Sampalli, John Williams, Abel Sanchez, "Privacy Preservation Scheme for Multicast Communications in Smart Buildings of the Smart Grid", Journal, Smart Grid and Renewable Energy (SGRE),Vol 4 No. 4, July 2013.

[6] Navneet Agrawal, "Secure Key Distribution Protocol with Smart Meter", International Journal of Current Engineering and Technology, Vol.5, No.5, Oct 2015.

[7] Luca Veltri a, Simone Cirani a, Stefano Busanelli b, Gianluigi Ferrari, "A novel batch-based group key management protocol applied to the Internet of Things", Elsevier Journal, Ad Hoc Networks, Volume 11, Issue 8, November 2013, Pages 2724–2737.

[8] Harney, Meth and Colegrove, "GSAKMP: Group Secure Association Key Management Protocol", RFC 4535, June 2006.

[9] Weis, Rowles and Hardjono, "The Group Domain of Interpretation", RFC 6407, October 2011.

[10] T.Hardjono, B.cain, and L.Monga. "Intra-domain Group key Management for Multicast Security". IETF internet Draft, September 2000.

[11] International Telecommunication Union - ITU-T Y.2060 - (06/2012) - Next Generation Networks - Frameworks and functional architecture models - Overview of the Internet of things.

[12] D.Wallner, E.Harder and R.Agee. “Key Management for Multicat: Issues and Architecture”. RFC 2627, June 1999.