LMGROUP: A Lightweight Multicast Group Key Management for IoT Networks

Nikbakht Bideh, Pegah

Published in:
Information Security Practice and Experience

DOI:
10.1007/978-3-031-21280-2_12

2022

LMGROUP: A Lightweight Multicast Group Key Management for IoT Networks

Pegah Nikbakht Bideh[1]

Electrical and Information Technology department, Lund University
pegah.nikbakht_bideh@eit.lth.se

Abstract. Due to limitations of IoT networks including limited bandwidth, memory, battery, etc., secure multicast group communication has gained more attention, and to enable that a group key establishment scheme is required to share the secret key among the group members. The current group key establishment protocols were mostly designed for Wireless Sensor Network, and they require device interaction, high computation costs, or high storage on the device side. To address these drawbacks, in this paper we design LMGROUP, a lightweight and multicast group key establishment protocol for IoT networks, that is based on Elliptic Curve Integrated Encryption Scheme and HMAC verification and does not require device interaction. We also suggest an algorithm for unpredictable group member selection. Our experimental result of implementing LMGROUP indicates it has low storage, low computation, and low communication costs. Furthermore, the formal security verification indicates LMGROUP is secure and robust against different attacks.

Keywords: IoT · Group Key Establishment · Key Management

1 Introduction

IoT networks have several challenges, one of the challenges is that the majority of devices are resource constrained which means that they have limited memory, battery, power, and limited computational resources. In IoT networks bandwidth is another challenge [1, 2], since increasing the number of devices makes the bandwidth and communication resources limited as well. Due to these limitations, multicast group communication has become more favorable in IoT networks, since, sending multicast messages to a group of devices is more efficient than sending unicast messages and overloading the network with multiple messages. Multicast group communication is particularly important where software updates or patches are required to be sent to a group of devices simultaneously.

In order to enable multicast secure group communication, a group key needs to be established in advance. There have been a variety of group key establishment methods proposed which will be described in detail in Section 2. Most of these schemes have been designed for WSN (Wireless Sensor Network) which do not take characteristics of IoT devices (limited resources) and networks (limited bandwidth) into account. On the other hand, most of these schemes require interaction between group members to get access to the shared key, but this is hard to achieve in IoT networks where devices have the least communications. Examples of such networks in smart cities or smart homes are where different sensors are placed in different places to gather information about temperature, air pollution, etc. These sensors gather data and send aggregated data periodically (in a unicast way) to a central server for further analysis and device to device communication is less likely.

The central server then should be able to send control commands or updates/upgrades and patches to the sensors in a multicast way. These commands and updates should not be sent in a broadcast way to all IoT sensors for availability reasons, since if any unexpected error happens, it can affect the availability of the whole network. As a result, the central server needs to group the IoT devices and decides about group membership, this can be done manually by the administrator or automatically to make the group membership less predictable to attackers. In this paper, we first suggest an automatic algorithm for unpredictable group member selection. After the selection of group members, a group key needs to be established to the group members. These group keys need to be renewed frequently due to changes in the group membership to provide forward secrecy. For that, we then design LMGROUP, a new lightweight multicast group key management scheme, that can work efficiently in small to large networks. LMGROUP does not have the problems of other group key establishment schemes for WSN including interactions between group members or heavy procedures on constrained devices. We implement LMGROUP and the experiments indicate it has efficient memory usage, communication, and computation costs. The experiments also show the scalability of LMGROUP. Finally, the formal security verification indicates our multicast scheme is secure against different attacks such as replay attacks. Our main contributions are:

- We suggest an algorithm for unpredictable group member selection.
- We design a new lightweight and multicast group key management scheme based on hybrid cryptography.
- We implement LMGROUP and indicate it is scalable and it has efficient memory usage, communication, and computation costs.
- We formally verify LMGROUP and indicate it is secure against different attacks.

The rest of this paper is organized as follows: in Section 2, the related work on group key establishment methods for WSN and IoT networks is presented. In Section 3, the details of LMGROUP including the suggested group member selection algorithm and our designed scheme are described. Implementation details are presented in Section 4. Performance evaluation and formal security verification are described in Sections 5 and 6. Finally the paper is concluded in Section 7.

2 Related Work

A variety of Group Key Management (GKM) methods had been proposed for WSNs and IoT networks and they were extensively reviewed in [3–5], in the case of used cryptography method they are divided into three categories: symmetric, asymmetric, and hybrid. In the case of key establishment authority, these methods can be divided into centralized, distributed, and hybrid methods as well [3]. Centralized methods are mostly applicable to networks with static topology while distributed approaches are more suitable for dynamic networks where nodes have high levels of mobility and they can join and leave the network quite often. The focus of our work is on centralized schemes and we do not review distributed schemes here. Among the reviewed schemes in [3], the most lightweight centralized approaches applicable to static small to large networks are: LKH [6], S2RP [7], TKH [8], and LEAP [9].

LKH (Logical Key Hierarchy) is a multicast rekeying approach for WSNs. In LKH the nodes are divided into subgroups based on a logical hierarchy and a symmetric key is assigned to each leaf node. LKH has a reasonable communication cost in most WSN networks since only the existing members in the subgroup receive the rekeying messages, but each member of the group needs to maintain the keys from the leaf to the root node path which causes additional storage and computation cost on the node's side [6]. S2RP (Secure and Scalable Rekeying Protocol) is similar to LKH with almost the same performance results but instead, it has added security to authenticate the rekeying messages through the use of a one-way hash function [7].

TKH (Topological Key Hierarchy) is another variant of LKH in which the logical key tree is mapped to the physical topology of the nodes in the network (key tree), this further reduces the communication cost of total rekeying messages. In TKH, based on the routing tree, the key tree can be constructed, the nodes attach to a parent node until they reach the group controller (sink node). Although TKH reduces the storage overhead on the node side it does not provide any key authentication mechanism.

In LEAP (Localized Encryption and Authentication Protocol) four types of keys are established to the nodes including an individual key, a pairwise key, a cluster key, and a global key. This scheme has low computation, communication, and storage overhead but for broadcast authentication, it relies on μTESLA [10] which requires synchronization between nodes, but the node synchronization is heavy and it is hard to achieve in IoT or WSN networks.

Another lightweight and decentralized group key establishment for IoT was proposed in [4]. This scheme is also based on logical hierarchy with one Key Distribution Center (KDC) and several Sub Key Distribution Centers (SKDCs) that can be used to avoid the single point of failure problem in centralized based schemes discussed above. Same as LKH based schemes each device needs to store the keys from the leaf to the parent path. Again, this scheme can have high storage costs on the device side which are dependent on the subgroup size. In [11] another centralized key distribution scheme based on key tree hierarchy was proposed which again has high storage overhead.

As mentioned above the problems of Key Tree Hierarchy-based key establishment methods are high storage, high computation costs, not providing any key authentication mechanisms, or requirement of time synchronization which makes them non-practical to use for IoT networks. Most of these methods are dependent on the contribution of members of the same group, which is difficult to handle in IoT networks, especially in networks where IoT nodes do not have interactions with each other.

Other than key tree based methods, there exist a variety of non-interactive key agreement protocols [12–16] which are applicable to WSNs or IoT networks. A secure group key establishment based on Elliptic Curve Cryptographic (ECC) operations was proposed in [12] for IoT networks. In this work, the authors have used one-way cryptographic accumulators to enable the connection between the gateway and IoT nodes to establish secure group keys and use them for further communications. This work has high computation cost since key establishment requires one signing and one signature verification on the IoT unit side.

In [15] two lightweight protocols based on ECC operations were proposed which was an improvement of the schemes proposed in [13, 14]. The protocols provide authenticity, confidentiality, and integrity but they are vulnerable to replay attacks [16] and they have high computational costs (especially protocol 1 which requires two signature verifications on the IoT unit side) which make them non-applicable to IoT environments. As an improvement of [15], the authors in [16] proposed a new key establishment protocol based on the Identity-Based Credentials (IBC) mechanism and ECC operations which is resistant to replay attacks as well. In this scheme, they have used HMAC verification instead of signature verification which is more applicable to IoT devices than heavy signature-based operations. Although this scheme has a lower computation cost in comparison to [15], they have a higher communication cost due to the increased number of transferred messages, even this scheme does not consider multicast communication and group key sharing.

Considering the problems of Key Tree Hierarchy-based methods, we address these problems and suggested a scheme, LMGROUP, that does not require time synchronization, has low storage (it does not require storing all the keys from the leaves to the root), and has low communication and computation overhead. LMGROUP is a multicast authenticated key establishment mechanism with hybrid cryptography. In LMGROUP we consider the advantages of the protocol in [16] including HMAC verification and replay protection, we modified the second protocol presented in [15], protocol 2, and proposed a new scheme that is applicable to IoT networks which will be explained in detail in Section 3.

3 Scenario and Scheme

In this section first, we provide the assumptions about the IoT network and the use case in which the proposed scheme is most suitable and then we present the details of LMGROUP scheme.

3.1 Assumptions

In our network, we assume two types of nodes: resource-rich nodes and constrained IoT nodes. Resource-rich nodes have not limited storage and processing capabilities and they can be used to perform heavy operations and are used to manage different groups of IoT nodes. Resource-rich nodes are referred to as servers throughout this paper. We assume to have a fault-tolerant centralized architecture with redundant servers available in the network. In order to avoid a single point of failure having redundant servers is required. We assume a network scenario in which IoT devices are stable or have low mobility. Devices can join or leave the network due to any reason, e.g. physical maintenance operations, adding new devices, or removing old devices from the network. Examples of use case scenarios are in smart buildings with smart lights or smart doors where the devices have fixed positions.

In our use case networks, device interaction is not possible. We consider the devices will not contribute in any way to group key establishment, but still in these network scenarios since there are usually many devices available, to keep the bandwidth as low as possible, group key establishment is preferred to be done in a multicasted way. In case during multicast group key establishment, one of the group members fails to receive or fails to update the group key, in its' next contact with the server, it can retrieve the group key in a unicast way.

We assume that each device has owned a symmetric master key denoted by k_m, this key is provisioned by the network administrator in the setup phase, the k_m keys are stored with each device identity securely on the server as well. Later, in the bootstrapping phase k_m is used to extract a session key, k_s, based on a key derivation function to establish a secure session with the server. We assume that the server decides on the group members and the devices themselves can not decide which group they want to belong to. The devices can belong to multiple groups at the same time, but messages encrypted with one group key can not be decrypted with another key.

3.2 Network scenario

In our network scenario, after the bootstrapping phase, the devices send their aggregated data back to the server periodically based on defined time intervals. Then after receiving an acknowledgment from the server, the devices go to sleep mode to reduce the energy consumption. The server needs to decide in advance about which devices need to be grouped, our suggested algorithm to decide on group members will be described below in Section 3.3. After deciding on the group members, the server will piggyback a hint along with the acknowledgment of the previous message to the IoT device. This hint carries information about the new group *ID* and it can be used as the seed of the key derivation function on the IoT unit side to extract an authentication key which further is going to be used to authenticate the group key establishment messages. The required communication between IoT devices and the server to derive the group authentication key is depicted in Figure 1. After receiving the new group *ID* by the IoT devices that exist inside the group, they use it to derive the key K^* which is the authentication key. The authentication key will be used during the group key establishment phase to protect the authenticity of multicast messages.

Other than the new group *ID*, the wake-up time should also be sent to the IoT devices, so that they will be able to wake up at the defined time to receive the multicast group key establishment messages. In order to avoid heavy time synchronization procedure on IoT unit sides, instead of sending the wake-up time, the server sends two other parameters to the devices: T and a window frame. T defines the number of seconds that the device needs to wake up after receiving the acknowledgment, and the window frame (which depends on the network latency, delay, etc., will be defined by the network administrator), defines the window frame in which the device should be active. As an example, if T is 6000 and the window frame is 60, then the device needs to be active from 5940 until 6060 seconds after receiving the acknowledgment.

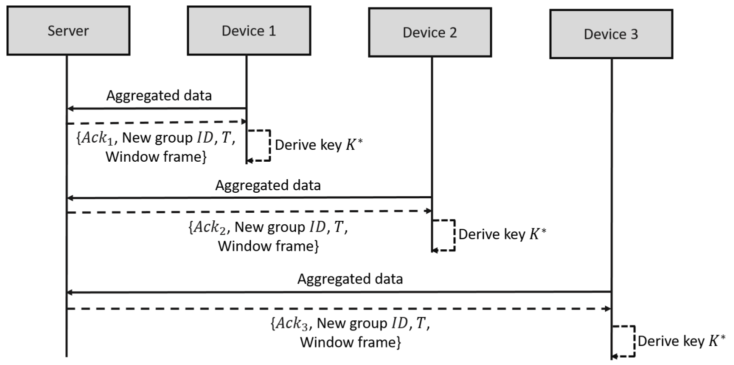
Server
Device 1
Device 2
Device 3
Aggregated data
{Ack_1, New group ID, T, Window frame}
Derive key K^*
Aggregated data
{Ack_2, New group ID, T, Window frame}
Derive key K^*
Aggregated data
{Ack_3, New group ID, T, Window frame}
Derive key K^*

Fig. 1. Communication between server and three IoT devices in the same group to derive authentication key

3.3 Group Member Selection

In WSN or IoT networks where nodes have more mobility, the nodes themselves can join or leave the groups based on different factors, e.g. signal strength, distance, etc. There have been some methods suggested for group member selection [17] or cluster head selection [18, 19] in WSN. These algorithms do not apply to our use case scenario or in general to the networks where nodes are stable and they cannot choose the grouping themselves, thus, we suggest an algorithm for group member selection that can be used by the server in our scheme.

Available IoT devices in the network are registered by the network administrator to the central server (and redundant servers). On the device registration, information including device identity number, device master secret, device public key, and device criticality level will be stored on the server. The device criticality level will be decided by the network administrator, and it depends on how critical the device's role is in the network. For example, in the case of smart doors in a hospital, the main entrance door has the highest criticality while the sub-doors have lower criticalities. We have considered three levels of criticality: low, medium, and high. These levels will be used for group member selection.

The group member selection should not be predictable by an outsider attacker, otherwise, the devices targeted for multicast group keying or update can become a target of DoS (Denial Of Service) attacks. Our suggested group member selection algorithm selects devices based on criticality levels and makes sure not all critical devices are in the same group. The algorithm works as follows:

Algorithm 1 Group member selection algorithm

```
Require: $n$, the number of group members, $N$, the total number of registered devices
    $H, M$, and $L \geq 0$ the number of devices with high to low criticality, respectively,
    such that $H + M + L = N$.
    $G_c \leftarrow \frac{N}{n}$,
    $H' \leftarrow \lfloor \frac{H}{G_c} \rfloor$, $M' \leftarrow \lfloor \frac{M}{G_c} \rfloor$, $L' \leftarrow \lfloor \frac{L}{G_c} \rfloor$, (The remaining members will be added to the
    groups later by the administrator.)
    $i \leftarrow 0$,
    while $i < G_c$ do
        $G[i] \leftarrow$ Take random members from $H$, $M$, $L$ with the size of $H'$, $M'$, and $L'$,
    respectively.
        $i \leftarrow i + 1$
    end while
    Return the groups, $G[i]$s.
```

As it can be seen in Algorithm 1, based on the number of available devices with different criticality levels, the members of a group will be formed randomly. If new devices join the network, they will get group membership on the next round of running the algorithm. In case a device leaves the network due to maintenance or replacement, the group will continue with previous members until the next round of running the algorithm. Leaving a group to join another group is not possible by the device, since the grouping process is only done by the server. The network administrator decides how often the group member selection algorithm should happen.

3.4 Designed scheme

As mentioned earlier in Section 2, two lightweight key establishment protocols based on ECC operations were proposed in [15], and an improved version of them was proposed in [16]. The second protocol presented in [15], protocol 2, and the improved protocol presented in [16] are the basis of our scheme. In order to better understand our designed scheme, here, we first briefly explain these two protocols and then we suggest our scheme.

Basis of LMGROUP The protocol 2 [15] uses ECIES or Elliptic Curve Integrated Encryption Scheme algorithm to establish a shared secret among the group members. In this scheme, an initiator (I) with several responders $U_j s$ in the network are considered and the group members are determined by the initiator. The random r is generated by I, then R is calculated as $R = rG$ (G is the base point as in ECDH). Then for each group member EC points $S_j s$ are computed by the initiator, $S_j = d_i Q_j + R$, and Q_j represents the public key of group members. The point $S_j = (x_i,y_j)$ will be encoded to another point (u_i,v_i) by calculating the hash over the point values. Then the encoded points will be XORed and the results will be concatenated to make the set P. The secret key of the group, k, is then the hash value over the XORed values of u_i. The $Auth$ value is calculated as $Auth = h(k \parallel R \parallel P)$, and finally, the multicast message that will be sent to the group members is: $Auth, C, R, U, P$, in which C is the counter value and finally a digital signature will be added to the message and the message will be broadcasted to all sensor nodes in the network. Each receiver first checks if its' identity is included in the U part of the message, if yes, then it verifies the signature and the counter value. If the verification is successful it computes u_j using R and its' private key as: $S_j = d_j Q_i + R$. The node will encode the point and finally, the values of the encoded point will be used to derive the group key. After that, the node verifies the authenticity of the key by checking if $Auth$ is equal to $h(k \parallel R \parallel P)$. Finally, the recipient nodes will send an acknowledgment to the initiator to finish the handshake. The problems of the above scheme are listed below:

- It is not protected against replay attacks;
- It requires heavy signature verification on the IoT unit side;
- The first message needs to be broadcasted to all sensor nodes in the network, this can cause many extra checking by IoT devices not belonging to the same group and can further cause extra overhead to the whole network.

In [16], the authors proposed a key management scheme that is quite similar to the work explained above [15]. In [16] HMAC verification is used instead of signature verification which makes it more energy efficient in comparison to [15]. In [16] the replay attack protection is considered but the scheme only works in a unicasted way and it can not be used for group key management. The other problem of the scheme [16] is that, although it has lower computation overhead than the work presented in [15] it causes higher communication overhead.

LMGROUP In our designed scheme, we take the advantages of these two protocols [15, 16] including multicast and ECIES based operations for group key sharing from protocol [15] and HMAC verification instead of heavy signature verification for authenticity from protocol [16]. We design LMGROUP that is lightweight in case of communication, computation, and storage overhead. We also suggest a replay protection mechanism and a group member selection technique, so that the messages are not required to be sent to all devices in the network and they can be sent to the target devices from the beginning. We describe the details of LMGROUP here.

The operation flow of our designed scheme is depicted in Figure 2. After deciding about the group members and pre-establishment of the authentication key K^* to group members by the server, as can be seen in Figure 2, the operation flow of LMGROUP which first begins on the server side is as follows:

– The server selects a random r and computes $R = rG$, in which G is the base point as in ECDH, and based on the number of group members in the range of *1 to n*, the point $S_j = dQ_j + R = (x_i,y_j)$ will be calculated, where d is the private key of the server and Q_j is the public key of group members.
– A unique random session ID will then be generated by the server that is used to protect against replay attacks.
– For each member of the group, $\overline{x_j} = \{\oplus_{i\neq j} x_i\} \oplus y_j$ will be calculated and then the set $Se = (\overline{x_1}, ..., \overline{x_n})$ will be formed, and the group key is calculated as $k = h(\oplus_i x_i)$ which is the hash over XOR values of x_i.
– The HMAC will be calculated over the fields of $\{ID,\ Auth, R, Se\}$ with the use of the authentication key K^* and it will be sent along with $\{ID,\ Auth, R, Se\}$ to all group members.

Then the flow continues on the device side:

– The group members upon receiving the message, first verify the HMAC, then each recipient device uses R and its own private key d_j to construct the point $S_j = d_jQ + R = (x_j, y_j)$.
– Then, the device extracts its own $\overline{x_j}$ from the received set Se, and then it can derive the group key as $k = h(\overline{x_j} \oplus x_j \oplus y_j)$, $\overline{x_j}$ contains other x and y_j value, therefore the similar values of y_j values will be removed from the calculation and only all x values will be XORed as expected.
– After key derivation, the $Auth$ should be checked if it is equal to $h(k \| R \| Se)$ or not. If $Auth$ is valid then an acknowledgment will be calculated as $Ack_j = h(k \| ID \| Q_j)$, in which k is the extracted group key, ID is the received session ID by the device and Q_j is the device public key.
– The acknowledgment along with device ID (device identity number) will be sent back to the server.

The flow then is finalized on the server side:

– The server uses the group key and device public key Q_j to verify the acknowledgment. On a successful verification, the authenticity of the derived group key by the device is verified as well.

4 Implementation

We have implemented LMGROUP in a real testbed setup. For the implementation, we have used ESP32-S2[1] a popular IoT development board representing

[1] https://www.espressif.com/en/products/socs/esp32-s2

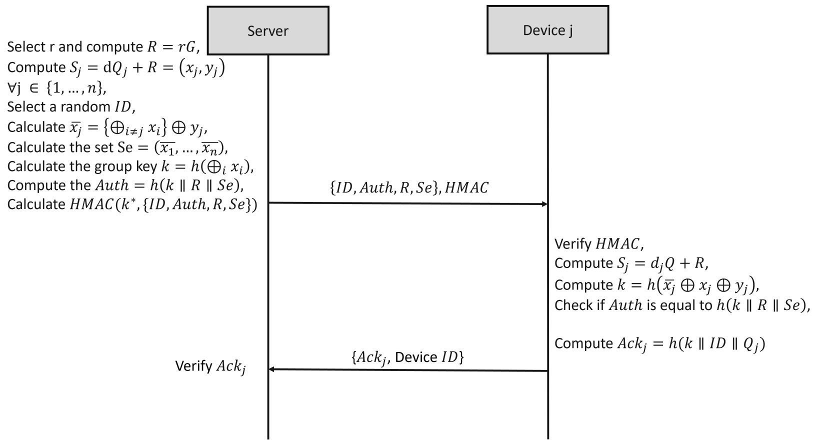
Server
Device j
Select r and compute $R = rG$,
Compute $S_j = \mathrm{d}Q_j + R = (x_j, y_j)$
$\forall j \in \{1, \ldots, n\}$,
Select a random ID,
Calculate $\overline{x_j} = \{\bigoplus_{i \neq j} x_i\} \oplus y_j$,
Calculate the set Se = $(\overline{x_1}, \ldots, \overline{x_n})$,
Calculate the group key $k = h(\bigoplus_i x_i)$,
Compute the $Auth = h(k \parallel R \parallel Se)$,
Calculate $HMAC(k^*, \{ID, Auth, R, Se\})$
$\{ID, Auth, R, Se\}, HMAC$
Verify $HMAC$,
Compute $S_j = d_j Q + R$,
Compute $k = h(\overline{x_j} \oplus x_j \oplus y_j)$,
Check if $Auth$ is equal to $h(k \parallel R \parallel Se)$,
Compute $Ack_j = h(k \parallel ID \parallel Q_j)$
$\{Ack_j$, Device $ID\}$
Verify Ack_j

Fig. 2. Operation flow between server and a sample device in LMGROUP

IoT devices (the implemented code for device side and server side is available[2]). ESP32-S2 is a low power and single core Wi-Fi Microcontroller SoC, which has high performance with a rich set of IO capabilities. ESP32-S2 has cryptographic hardware accelerators for enhanced performance, and it integrates a rich set of peripherals, with different programmable GPIOs that can be configured to provide USB OTG, LCD interface, UART, and other common functionalities and in our implementation, in order to annotate measurements we have used UART interface. In our measurements, we used Otii Arc[3] device as a power analyzer to record and measure real time currents and voltages using UART logs.

We have used SHA256 for the hash function and for HMAC we have used HMAC-SHA256. For the hash function, we have used hardware acceleration on ESP32-S2. On the IoT device side for the ECC point addition and multiplication, we have used the Mbed TLS library.

4.1 Testbed and Environmental Setup

Our testbed consists of 10 ESP32-S2 boards and 7 out of these 10 boards are grouped by the server to receive the group key. After the server decides the group members, it calculates: S_j, $\overline{x_j}$ (j in range of 1 to 7), Se, k. $Auth$ and HMAC (as explained in Section 3.4). The calculation of these values can be done at any time between the time the group members have been decided until the time the devices wake up, based on the server workload during this time. During the specified wake-up time window, the devices wake up and wait to receive the group key information from the server. Our testbed setup and further communications between the server and IoT devices are depicted in Figure 3. As shown in Figure 3, the multicast group key message will be sent to all of the grouped devices

[2] https://github.com/pegahnikbakht/Multi-key-share
[3] https://www.qoitech.com/otii/

by the server. The devices after receiving, verifying the message, and extracting the group key will send back an acknowledgment to the server. Whenever the server receives acknowledgment from all members, the group key is established and can be used for further communications. The server has a specified timeout for receiving the acknowledgments from all group members, if the server does not receive the acknowledgment from any of the group members, it will send again the group key information to those members in a unicast way later.

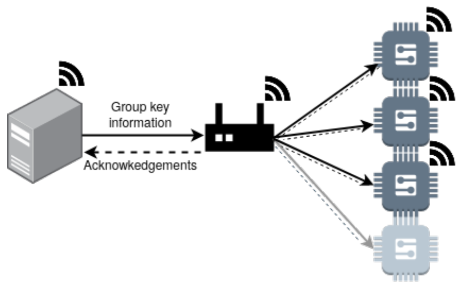

Fig. 3. Testbed setup for LMGROUP

5 Performance Evaluation

In order to show the efficiency of our group key establishment scheme, we measure the communication, computation, and storage overhead of our scheme.

5.1 Communication and Computation Cost

In order to calculate the communication overhead, the number of transferred bytes between the server and IoT devices during the group key establishment have been calculated, in the calculations, the number of bytes in the acknowledgment is also included. The number of bytes in the first message from server to the devices are $145+32*n$ bytes (n is the number of devices inside the group), which consists of 16 bytes of ID, 32 bytes of HMAC, 32 bytes of $Auth$, 64 bytes of R, and $32*n$ bytes of Se which depends to the number of group members n. The response back from the device includes an acknowledgment (32 bytes) and a *Device ID* (12 bytes) which is in total 44 bytes. Therefore the total number of transmitted bytes between server and devices are $189+32*n$ bytes.

In order to compute the computation overhead on the device side, the energy consumption and the time was measured from the time the device receives the group key information from the server until it sends back the acknowledgment. We have done the measurements using two different elliptic curves with the same security level, Secp256r1 (prime field curve) and Secp256k1 (Koblitz curve). The total time elapsed for the key establishment on the server was also measured which is the time from when the key establishment message was sent until all acknowledgments from the devices in the multicast group have been received and verified. The results of computation overhead are indicated in Table 1. As can be seen single EC point addition and multiplication of Secp256r1 consumes more energy and requires more time than Secp256k1 since prime field curves are few bits stronger than Koblitz curves [20]. Therefore using the prime field curve Secp256r1 for the group key establishment would require more time and energy on both the device side and the server side. The required time for single acknowledge verification does not depend on the curve type and it is almost equal for both curves as can be seen in Table 1, but since Secp256r1 requires more processing time on the device side, the time between the arrival of different acknowledgments as well as the total key establishment time will increase. From Table 1, we can also conclude that the majority of used energy and time on the IoT side is due to ECC operations, therefore hash function and MAC function operations are considered negligible.

Table 1. Computation overhead of LMGROUP with two different curves

Curve		Energy (μwh)	Time (ms)
Secp256k1	Total key establishment (IoT side)	32.0809	405.7857
	Total key establishment (Server side)	-	570.3442
	Single Ack verification (Server side)	-	0.1129
	Time between arriving Acks (Server side)	-	0.8210
	Singel EC point addition and multiplication	25.6172	321.3448
Secp256r1	Total key establishment (IoT side)	39.0281	494.0000
	Total key establishment (Server side)	-	692.0681
	Single Ack verification (Server side)	-	0.1226
	Time between arriving Acks (Server side)	-	1.3483
	Singel EC point addition and multiplication	33.5333	414.1000

According to the measured values in Table 1, we tried to simulate how long the whole key establishment time (from the time the server sends the key establishment message until it receives and verifies all the acknowledgments from group members) takes on the server side for larger group sizes. The results for different group sizes are depicted in Figure 4. As can be seen the key establishment time increases by increasing the group size, and for large group sizes (larger than 100) this increase is more noticeable. Although the key establishment time for larger group sizes increases but for instance this increase for a group of 448 nodes is still less than a second, hence LMGROUP is scalable to large size networks as well. Note that based on the latency threshold in the network [21] an appropriate group size should be selected. As mentioned earlier curve Secp256k1 has better performance than Secp256r1 as can be seen in Figure 4.

We have also compared the computation and communication overhead of LMGROUP to the schemes presented in [15, 16], and the results are depicted in Table 2. Different operations are indicated as follows: PM for ECC point multiplications, PA for point addition, h for hash function operation, SV for signa-

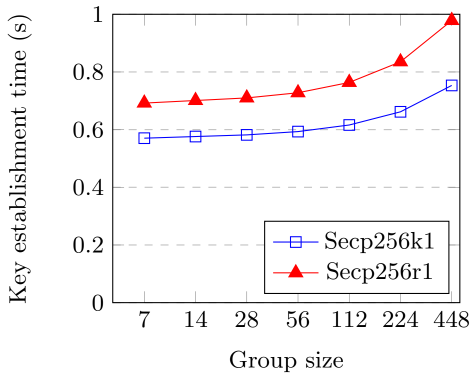

Fig. 4. Total key establishment time of LMGROUP for different group sizes

ture verification, *HM* for HMAC operation, *MM* for modular multiplication, *SE* for symmetric encryption, and *SD* for symmetric decryption. As hash function and HMAC operations are negligible, we can conclude from Table 2 that the protocols presented in [15, 16] obviously have more computation overhead than LMGROUP due to heavy signature verification operation and also having more ECC or modular operations in schemes [15, 16], respectively. In scheme [15], the authors have originally used the curve Secp160r1, which has less byte overhead than Secp256r1 or Secp256k1 (the curves used in LMGROUP), in order for the scheme [15] to be comparable with our scheme, we have considered a curve with 256 bits modules in [15] as well. The results of computation overhead comparison are also indicated in Table 2. Considering the communication overhead represented in Table 2, we calculate the communication overhead for different values of n or the group size, and the results are depicted in Figure 5, as can be seen LMGROUP has slightly higher overhead than the protocol [15] and this is due to the fact that in scheme [15] SHA128 was used as the hash function which generates 16 bytes lower overhead than SHA256 which was used in LMGROUP. The protocol [16] is not a multicast protocol and as can be seen, it has the highest byte overhead, increasing the group size will cause a significant increase in its' communication overhead.

Table 2. Communication and computation overhead comparison of LMGROUP with the protocols presented in [15, 16]

	LMGROUP	Protocol [15]	Protocol [16]
Computation Overhead (number of operations)	PM+PA+3h+HM	PM+PA+5h+SV	2PM+2MM+ h+2HM+SE+SD
Communication Overhead (number of bytes)	$189 + 32 * n$	$146 + 18 * n$	$128 * n$

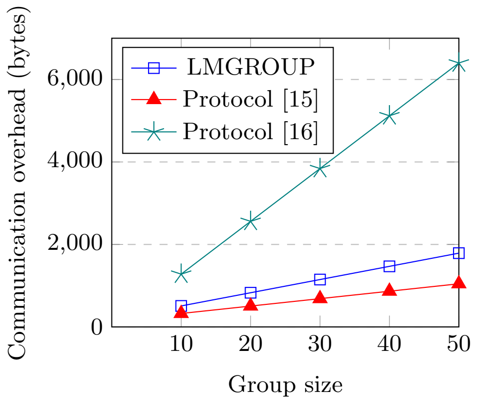

Fig. 5. Communication overhead comparison of LMGROUP and protocols [15, 16]

5.2 Storage Overhead

In LMGROUP the following information needs to be stored on the device side: public key of the server, device private and public keys, device *ID* and some other variables regarding the used ECC curve. We measured the memory footprints of our multicast key establishment scheme using ESP32-S2 and the results are shown in Table 3. DRAM specifies the RAM usage which is assigned to zero and non-zero values at the startup of the program. IRAM indicates the total executable code which is executed from IRAM, and D/IRAM specifies the total size of DRAM and IRAM together. Flash code specifies the total size of executable code which is executed from the flash cache or IROM. Flash rodata on the other hand indicates the total size of read-only data that is loaded from the flash cache or DROM. Finally, total image size indicates the estimated total binary file size of the program which includes the whole size of all used memory types. As indicated in Table 3, RAM and ROM usage of LMGROUP are reasonable on the IoT device side, and considering the limitations of resource constrained devices LMGROUP is applicable to be used in such devices.

Table 3. Memory footprints of LMGROUP

	D/IRAM(B)	Flash Code(B)	Flash rodata(B)	Total image size(B)
LMGROUP	110855	467427	102736	681018

6 Formal Security Verification

In order to formally verify the security properties of LMGROUP, we have used ProVerif [22]. ProVerif uses Dolev-Yao model [23] for the adversary model and it can be used to formally verify the security properties of cryptographic protocols. Applied pi calculus [24] is used in ProVerif as the modeling language. In our protocol modeling, we first start with the declaration phase where different components of the protocol including variables, functions, and channels are declared. We used different types in our ProVerif model to declare the type of variables such as key, nonce (used for session *ID*), point (used for ECC point), and bitstring. The term [private] in front of some variable definitions indicates that those variables are not known by the attacker. Our modeled protocol using ProVerif is also available[4].

The main functions used in the modeling of our key establishment scheme are hash function, ECC point multiplication and addition, MAC, and XOR, these functions are modeled as follows:

```
fun hash(bitstring):bitstring.
fun mul(bitstring,bitstring):bitstring.
fun add(bitstring,point):point.
fun mac(key,nonce,bitstring,point,bitstring) : bitstring.
fun xor(bitstring,bitstring):bitstring.
```

The functions can have different input and output types, as an example MAC function takes five inputs of type key, nonce, bitstring, point, and bitstring and it generates an output of type bitstring.

After declaring variables and functions, we defined different queries, these queries are used to check whether the protocol has specific security properties or not. We verified security properties including secrecy, authentication, and correspondence through different queries. These security properties can protect against various attacks including: 1) Man in The Middle attack, 2) Replay and Impersonation attacks, and 3) Denial Of Service attack. The queries used to verify the secrecy properties to protect against these attacks are described below.

6.1 Man in The Middle Attack Protection

Man in The Middle Attack (MITM) can be protected through secrecy property. To verify secrecy we have used the following queries:

```
query attacker ( da ).
query attacker ( db ).
```

In our ProVerif model, we modeled two devices, Device A and B, modeled as *Da* and *Db*, and a server modeled as *S*. In the above queries, *da* and *db* represent the private keys of devices A and B, and the queries check whether the attacker can gain any knowledge about the private keys of those devices or not. The result of ProVerif verification indicates that the above queries are successfully verified and the attacker can not get access to *da* and *db* and further can not get access to the information required in generating the group key.

[4] https://anonymous.4open.science/r/Multi-key-share-C4AF/

6.2 Replay and Impersonation Attacks Protection

We have used correspondence property to verify protection against replay and impersonation attacks. To prove correspondence, we have used these queries:

```
query a:bitstring,b:nonce,c:bitstring,d:point,e:bitstring;
event(termDevice(a,b,c,d,e))==>event(initserver(a,b,c,d,e)).

query a:bitstring,b:bitstring;
event(termserver(a,b))==>event(initDevice(a,b)).
```

As it can be seen four different events are used in the above queries: *initDevice* and *termDevice* refer to initiating and terminating the device respectively, and *initserver* and *termserver* refer to initiating and terminating the server. The inputs of the events in the first query represent HMAC, *Session ID*, *Auth*, *R*, and *Se* and the inputs of events in the second query are device *ID* and acknowledgement. The above queries are satisfied if for each occurrence of the event *termDevice*, there is a previous execution of *initserver*, and also if for each occurrence of *termserver* there is a previous execution of *initDevice*. These correspondence relations protect against replay and impersonation attacks. The above queries are successfully verified in ProVerif.

6.3 Denial Of Service Attack Protection

To protect against DoS attacks, we have used the authentication property, if all of the messages are authenticated in the protocol then it can protect against DoS attacks, we have used *HMAC* verification to verify the message in LMGROUP. To prove authentication again the correspondence assertion (the relationships between the execution of events) is used in ProVerif and the same queries used in Section 6.2 are used to verify authentication. ProVerif has successfully verified correspondence as well.

7 Conclusion

In this paper, we propose LMGROUP a lightweight and multicast group key establishment scheme for IoT networks. In LMGROUP the IoT devices do not need to interact with each other to gain the shared key, instead, a central server is used to select the group members and send group information to the members (having some redundant servers is preferred to avoid single point of failure). In this paper, we suggest an unpredictable group member selection algorithm based on the criticality level (which is decided by the network administrator) of the devices in the network. LMGROUP uses ECIES based operations for sharing the group key and it uses HMAC verification instead of heavy signature verification to authenticate the group key establishment messages. The evaluation result of implementing LMGROUP on a real testbed setup indicates it is lightweight and scalable and can be used in small to large size networks. The results also indicate

LMGROUP has low storage, low communication, and computation costs. Furthermore, we also formally verify LMGROUP and prove it is secure and robust against different attacks including replay and DoS attacks.

Acknowledgments

This work was financially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP).

References

1. S Sujin Issac Samuel. A review of connectivity challenges in iot-smart home. In *2016 3rd MEC International conference on big data and smart city (ICBDSC)*, pages 1–4. IEEE, 2016.
2. Yuang Chen and Thomas Kunz. Performance evaluation of iot protocols under a constrained wireless access network. In *2016 International Conference on Selected Topics in Mobile & Wireless Networking (MoWNeT)*, pages 1–7. IEEE, 2016.
3. Omar Cheikhrouhou. Secure group communication in wireless sensor networks: a survey. *Journal of Network and Computer Applications*, 61:115–132, 2016.
4. Maissa Dammak, Sidi-Mohammed Senouci, Mohamed Ayoub Messous, Mohamed Houcine Elhdhili, and Christophe Gransart. Decentralized lightweight group key management for dynamic access control in iot environments. *IEEE Transactions on Network and Service Management*, 17(3):1742–1757, 2020.
5. Alessandro Piccoli, Marc-Oliver Pahl, and Lars Wüstrich. Group key management in constrained iot settings. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2020.
6. Chung Kei Wong, Mohamed Gouda, and Simon S Lam. Secure group communications using key graphs. *IEEE/ACM transactions on networking*, 8(1):16–30, 2000.
7. Gianluca Dini and Ida Maria Savino. S2rp: a secure and scalable rekeying protocol for wireless sensor networks. In *2006 IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, pages 457–466. IEEE, 2006.
8. Ju-Hyung Son, Jun-Sik Lee, and Seung-Woo Seo. Topological key hierarchy for energy-efficient group key management in wireless sensor networks. *Wireless personal communications*, 52(2):359–382, 2010.
9. Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. Leap+ efficient security mechanisms for large-scale distributed sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(4):500–528, 2006.
10. Adrian Perrig, Ran Canetti, J Doug Tygar, and Dawn Song. The tesla broadcast authentication protocol. *Rsa Cryptobytes*, 5(2):2–13, 2002.
11. Wenbin Yao, Si Han, and Xiaoyong Li. Lkh++ based group key management scheme for wireless sensor network. *Wireless Personal Communications*, 83(4):3057–3073, 2015.
12. Nico Ferrari, Teklay Gebremichael, Ulf Jennehag, and Mikael Gidlund. Lightweight group-key establishment protocol for iot devices: Implementation and performance analyses. In *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*, pages 31–37. IEEE, 2018.

13. Lien Harn and Changlu Lin. Authenticated group key transfer protocol based on secret sharing. *IEEE transactions on computers*, 59(6):842–846, 2010.
14. Chia-Yin Lee, Zhi-Hui Wang, Lein Harn, and Chin-Chen Chang. Secure key transfer protocol based on secret sharing for group communications. *IEICE TRANSACTIONS on Information and Systems*, 94(11):2069–2076, 2011.
15. Pawani Porambage, An Braeken, Corinna Schmitt, Andrei Gurtov, Mika Ylianttila, and Burkhard Stiller. Group key establishment for enabling secure multicast communication in wireless sensor networks deployed for iot applications. *IEEE Access*, 3:1503–1511, 2015.
16. Abubakar Sadiq Sani, Dong Yuan, Phee Lep Yeoh, Wei Bao, Shiping Chen, and Branka Vucetic. A lightweight security and privacy-enhancing key establishment for internet of things applications. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.
17. Fatemeh Kazemeyni, Einar Broch Johnsen, Olaf Owe, and Ilangko Balasingham. Group selection by nodes in wireless sensor networks using coalitional game theory. In *2011 16th IEEE International Conference on Engineering of Complex Computer Systems*, pages 253–262. IEEE, 2011.
18. Trupti Mayee Behera, Sushanta Kumar Mohapatra, Umesh Chandra Samal, Mohammad S Khan, Mahmoud Daneshmand, and Amir H Gandomi. Residual energy-based cluster-head selection in wsns for iot application. *IEEE Internet of Things Journal*, 6(3):5132–5139, 2019.
19. Fahimeh Hamzeloei and Mohammad Khalily Dermany. A topsis based cluster head selection for wireless sensor network. *Procedia Computer Science*, 98:8–15, 2016.
20. Kristian Bjoernsen. Koblitz curves and its practical uses in bitcoin security. *order (ε (GF (2k)*, 2(1):7, 2009.
21. Theofanis P Raptis, Andrea Passarella, and Marco Conti. Performance analysis of latency-aware data management in industrial iot networks. *Sensors*, 18(8):2611, 2018.
22. Bruno Blanchet, Ben Smyth, Vincent Cheval, and Marc Sylvestre. Proverif 2.00: automatic cryptographic protocol verifier, user manual and tutorial. *Version from*, pages 05–16, 2018.
23. D. Dolev and A. C. Yao. On the Security of Public Key Protocols. In *Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*, SFCS '81, pages 350–357, Washington, DC, USA, 1981. IEEE Computer Society.
24. Mark D Ryan and Ben Smyth. Applied pi calculus. In *Formal Models and Techniques for Analyzing Security Protocols*, pages 112–142. Ios Press, 2011.