An Efficient Multi-Group Key Management Protocol for Internet of Things

Mohamed Ali Kandi, Hicham Lakhlef, Abdelmadjid Bouabdallah, Yacine Challal

An Efficient Multi-Group Key Management Protocol for Internet of Things

Mohamed Ali Kandi[1], Hicham Lakhlef[1], Abdelmadjid Bouabdallah[1] and Yacine Challal[2]
[1]Heudiasyc UMR CNRS 7253, Université de Technologie de Compiègne, Compiègne, France
[2]Laboratoire de Méthodes de Conception de Systèmes, École nationale Supérieure d'Informatique, Algiers, Algeria
Email: $\{mohamed-ali.kandi, hicham.lakhlef, madjid.bouabdallah, yacine.challal\}$ @$hds.utc.fr$

***Abstract*—Internet of Things (IoT) is a network made up of a large number of devices which are able to automatically communicate to computer systems, people and each other providing various services for the benefit of society. One of the main challenges facing the IoT is how to secure communication between these devices. Among all the issues, the Group Key Management is one of the most difficult. Although different approaches have been proposed to solve it, most of them use the same security parameters to secure all communications. Thus, if several services are provided by the network, communications within a service will be accessible to all network members even those which did not subscribe to it. Moreover, the compromise of a member will jeopardize all services. In this paper, we propose a highly scalable Multi-Group Key Management protocol for IoT which ensures the forward and backward secrecy, efficiently recovers from collusion attacks and guarantees the secure coexistence of several services in a single network. To achieve this, our protocol manages several groups with independent security parameters.**

***Index Terms*—Internet of things, service, security, Group Key Management, forward and backward secrecy, collusion attack.**

I. Introduction

The number of devices connected to Internet is constantly increasing since its appearance. Now that this number far exceeds that of people in the world, we are no longer talking about Internet but about Internet of Things. This emerging technology gives rise to revolutionary applications such as health care, environment monitoring, smart homes, smart cities...etc. The IoT devices, commonly called smart objects, are able to automatically communicate to computer systems, people and each other. The aim is to provide various services for the benefit of society. One of the main challenges facing the IoT is how to secure communication between these objects.

The Group Key Management (GKM) is the core of secure communication. Its main role is to establish secure links between the members of a group. To achieve this, the GKM provides them with a secret cryptographic key that is used to encrypt the data exchanged [19]. Nevertheless, when a member leaves the group, it must no longer be able to decipher the future communications (forward secrecy). Also, if a node joins the group, it must not be able to decipher the previous ones (backward secrecy). Backward and forward secrecy are usually guaranteed by rekeying. Thus, when a node joins or leaves the group, the secret key is revoked and a new one is distributed to the remaining members. However, multiple compromised nodes can cooperate to regain access to the secret key. Such an attack is referred to as collusion attack [16].

The GKM is a difficult issue especially for networks of constrained devices. Although different approaches have been proposed to solve it, most of them use the same parameters to secure all communications. Thus, if several services are provided by the network, communications within a service will be accessible to all network members even those which did not subscribe to it. Moreover, the compromise of a member will jeopardize all services. To address this problem, the hierarchical group access control has been proposed in [14] and [20]. However, this scheme cannot achieve a high performance when no hierarchy exists among services. A solution based on a Master Key Encryption has then been introduced in [9]. Nonetheless, using an asymmetric approach, this protocol is not well suited for networks of highly constraint devices. In this paper, we propose a highly scalable Multi-Group Key Management ($MGKM$) protocol for IoT which ensures the forward and backward secrecy, efficiently recovers from collusion attacks and guarantees the secure coexistence of several services in the network. To achieve this, our protocol manages several groups with independent security parameters.

The remainder of this paper is organized as follows: related works are discussed in Section II. We detail then our solution in Section III. Section IV presents the security analysis of our protocol. In section V, we evaluate the performance of our solution. Finally, we conclude in Section VI.

II. Related Works

According to the encryption technique used, the GKM schemes can be classified into three categories: symmetric, asymmetric and hybrid [19]. Symmetric approaches involve the use of the same key for encryption and decryption, while asymmetric ones use two different keys.

Generally, symmetric schemes require less computation time and are more suitable for limited resources devices [18]. However, most of them suffer from high communication and memory overhead, are not scalable and are not resilient against compromise [11]. Symmetric approaches are usually based on Logical Key Hierarchy (LKH) [5, 13], Exclusion Basis Systems (EBS) [4, 7], polynomials [6], matrices [17]...etc.

On the other hand, asymmetric protocols are more secure and scalable. However, they usually require intensive computing, which makes them impractical on constrained devices. Despite this, some asymmetric schemes was proposed even for wireless sensor networks. Most of them implemented Elliptic

Curve Cryptography (ECC) [1, 10, 12], CertificateLess Public Key Cryptography (CL-PKC) [8, 11], ID-Based Encryption (IBE) [3]...etc. Some works [2] proposed then hybrid schemes that combine both techniques (symmetric and asymmetric) to take advantages of each and overcome its disadvantages.

Regardless of the type of encryption used, only few researches considered the possibility of coexistence of several services in a single network. The authors in [14] and [20] proposed $MGKM$ schemes that achieve hierarchical group access control. However, the protocols cannot achieve a high performance when no hierarchy exists among services. The authors in [9] proposed then a new scheme called the Master Key Encryption Based $MGKM$. Nevertheless, the protocol is based on an asymmetric approach and is therefore not well suited for networks of highly constraint devices.

The symmetric protocol GREP [16] exploits the history of node joining to establish a total ordering among nodes. The aim is to make the rekeying process scalable and efficient. However, although nodes are organized into subgroups, their security parameters are not independent. All subgroups must then be rekeyed when a node gets compromised. On this basis, we propose a highly scalable $MGKM$ protocol for IoT which ensures the forward and backward secrecy, efficiently recovers from collusion attacks and guarantees the secure coexistence of several services in a single network.

III. Our solution

The $MGKM$ we propose uses two layers. The upper layer manages multiple groups and assigns nodes to them according to the services to which they subscribe. On the other hand, the lower layer distributes the nodes of each group into logical subgroups in order to reduce the protocol overheads on them. The network is then divided into several groups, each of which is also partitioned into several subgroups (Figure 1). By doing this, the security parameters of services will be independent and the protocol is lighter for the network nodes.

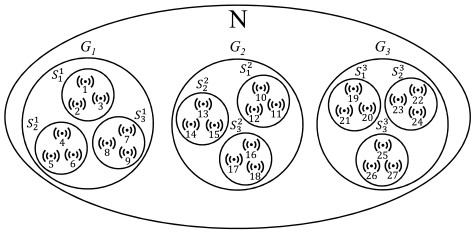

Fig. 1: Network partitioning according to services.

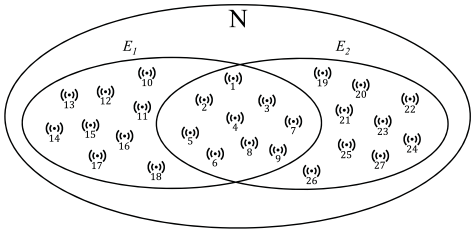

Fig. 2: Network partitioning achieved by the protocol.

A. Services and groups management

An IoT service is a transaction between two entities: a provider and a consumer. The former measures the state of the latter or initiates actions which will cause a change to it [15]. The provider is usually an object while the consumer can be a human, the environment or an other smart device. The main role of the GKM is to establish secure communications between the objects. A smart device can then participate to a service as a provider, a consumer or both. It may also participate to different services, at the same time, and subscribe or unsubscribe from services at any time. The IoT can then be seen as a set of overlapping classes each gathering nodes which collaborate to provide a service and others that benefit from it (Figure 2). As these classes are overlapping, a group of the protocol cannot be associated to a service. Indeed, the independence of the group security parameters will then lose its meaning and the compromise of a node can jeopardize several groups. We propose then the creation of a group for each possible combination of services. A combination A_i of k_i services, of a finite set E of e services, is a subset of k_i elements of E. The number of combinations, nc, is equal to:

$$nc = \sum_{k=1}^{e} C_e^k = 2^e - 1 \tag{1}$$

The network N is then partitioned into groups. Each group G_i is associated with an ID, gid_{G_i}, which is unique within N. It contains then the nodes participating in the services of the combination A_i associated to it. When an actual member subscribes or unsubscribes from services, it migrates from a group to another according to its new combination of services. The number of groups can reach nc (Formula 1) if there are nodes participating in every possible combination of services. On the other hand, it cannot exceed the number of network nodes, n, because empty groups are not allowed. The maximum number of groups, max_g, is therefore equal to:

$$max_g = Min(2^e - 1, n) \tag{2}$$

Groups are created and removed as and when required and the probability of having only one node in each group is low. Their number can then be much smaller than max_g. In Figure 2, two services E_1 and E_2 coexist in N. Three combinations are then possible: $A_1 = \{E_1, E_2\}$, $A_2 = \{E_1\}$ and $A_3 = \{E_2\}$. Each group G_i contains the nodes participating in the combination of services A_i associated to it (Figure 1).

B. Subgroups management

To reduce the protocol costs for nodes, each group G_i is in turn partitioned into a set of logical subgroups. A subgroup S_j^i is associated with an ID, $sid_{S_j^i}$, which is unique within G_i and reflects its subgroups' total order. Given two subgroups $S_{j_1}^i$ and $S_{j_2}^i$, $sid_{S_{j_1}^i} < sid_{S_{j_2}^i}$ if and only if $S_{j_1}^i$ was created before $S_{j_2}^i$. Thereby, $S_{j_1}^i$ is considered as an elder kindred of $S_{j_2}^i$ whereas the latter is seen as a junior kindred of the former. Each subgroup S_j^i is also associated to two subgroup tokens: a forward, $st_{S_j^i}^F$, and a backward one, $st_{S_j^i}^B$. In Figure 1, each group is partitioned into three subgroups.

C. Nodes management

Each node u is first assigned to a subgroup S_j^i of the group G_i. The group is chosen according to the combination of services A_i in which u participates. On the other hand, the problem of the choice of the subgroup is treated by us in an ongoing work. The node u is then associated with an ID, nid_u, which is unique within S_j^i and reflects, its members' total order. Given two nodes u and v, $nid_u < nid_v$ if and only if u has joined S_j^i before v. Thus, u is considered as an elder cognate of v whereas v is seen as a junior cognate of u.

Also, u is associated to two node tokens: a forward, t_u^F, and a backward one, t_u^B. The node does not know neither its tokens (t_u^F and t_u^B) nor those of S_j^i ($st_{S_j^i}^F$ and $st_{S_j^i}^B$). However, it stores the forward and backward node tokens associated to its elder and junior cognates, respectively, and the forward and backward subgroup tokens of the elder and junior kindred of S_j^i, respectively. Moreover, u holds a secret node key, K_u, and shares a subgroup key, $K_{S_j^i}$, with its cognates and a group key, K_{G_i}, with all the nodes of G_i. Finally, u stores a service key, K_{E_k}, for each service E_k in which it participates. Note that unlike other keys, those of services do not intervene in the rekeying process and are only used to encrypt communication. The table in Figure 3 shows the keys and tokens held by the nodes of the group G_1 of the example presented in Figure 1.

Node ID	Node key	Subgroup key	Service keys	Group key	Node tokens		Subgroup tokens	
1	K_1	$K_{S_1^1}$	K_{E_1}, K_{E_2}	K_{G_1}	-	t_2^F, t_3^F	-	$st_{S_2^1}^F, st_{S_3^1}^F$
2	K_2				t_1^B	t_3^F		
3	K_3				t_1^B, t_2^B	-		
4	K_4	$K_{S_2^1}$			-	t_5^F, t_6^F	$st_{S_1^1}^B$	$st_{S_3^1}^F$
5	K_5				t_4^B	t_6^F		
6	K_6				t_4^B, t_5^B	-		
7	K_7	$K_{S_3^1}$			-	t_8^F, t_9^F	$st_{S_1^1}^B, st_{S_2^1}^B$	-
8	K_8				t_7^B	t_9^F		
9	K_9				t_7^B, t_8^B	-		

Fig. 3: Example of keys and tokens held by nodes.

In the following, we use the notations t_M, K_R, KEK, H and KDF to refer to a master node token, a refresh key, a key encryption key, a one-way hash function and a pseudo-random key derivation function, respectively. Also, $\{m\}K$ means the message m is encrypted using K and $MGKM \rightarrow R :< m >$ denotes that the $MGKM$ sends $< m >$ to the node(s) R.

1) Rekeying upon joining: When a node u joins the network, the $MGKM$ starts by assigning it to a given subgroup S_j^i of the group G_i. Due to space constraints, we assume that the subgroup S_j^i already exists. It determines then nid_u, K_u and t_u^B. To ensure the backward secrecy, the $MGKM$ randomly generates K_R and t_M. Then, using the KDF, it computes t_u^F and the new group and subgroup keys, $K_{G_i}^+$ and $K_{S_j^i}^+$ (Formulas 3 to 5). The KDF is also used to update the key K_{E_k} of each service E_k to which u subscribes (Formula 6). Finally, the $MGKM$ discards t_M and K_R and broadcasts the messages $JM1$ to $JM3$. The message $JM1$ is sent to the nodes of S_j^i and is encrypted using its actual key $K_{S_j^i}$. The message $JM2$ is intended to those belonging to the other subgroups of G_i and is then encrypted using its actual key K_{G_i}. For each group G_s ($G_s \neq G_i$), which shares at least a service with G_i ($A_s \cap A_i \neq \emptyset$), the message $JM3$ is sent encrypted using K_{G_s}.

$$t_u^F = KDF(t_M||K_R) \quad (3) \qquad K_{G_i}^+ = KDF(K_{G_i}||K_R) \quad (4)$$

$$K_{S_j^i}^+ = KDF(K_{S_j^i}||K_R) \quad (5) \qquad K_{E_k}^+ = KDF(K_{E_k}||K_R) \quad (6)$$

$$JM1 : MGKM \rightarrow S_j^i :< nid_u, \{t_M, K_R\}\, K_{S_j^i} >$$

$$JM2 : MGKM \rightarrow G_i :< \{K_R\}\, K_{G_i} >$$

$$JM3 : MGKM \rightarrow G_s :< \{A_s \cap A_i, K_R\}\, K_{G_s} >$$

Upon receiving $JM1$, a node in S_j^i uses $K_{S_j^i}$ to retrieve t_M and K_R, computes t_u^F, $K_{G_i}^+$ and $K_{S_j^i}^+$ and updates its service keys (Formulas 3 to 6). Upon receiving $JM2$, a node in S^i ($S^i \in G_i$ and $S^i \neq S_j^i$) uses K_{G_i} to retrieve K_R, computes $K_{G_i}^+$ and updates its service keys (Formulas 4 and 6). Upon receiving $JM3$, a node in G_s uses K_{G_s} to retrieve K_R and updates the shared service keys (Formula 6). Finally, the $MGKM$ provides u, via a secure channel, with K_u, $K_{G_i}^+$, $K_{S_j^i}^+$, the service keys, the backward tokens of its cognates and the backward and forward ones of the kindreds of S_j^i.

2) Rekeying upon leaving: When a node u leaves a subgroup S_j^i of the group G_i, the cryptographic material it holds get compromised and must be revoked. By construction, four tokens remain secret (t_u^F, t_u^B, $st_{S_j^i}^F$ and $st_{S_j^i}^B$) and are then used to rekey the network. Due to space constraints, we do not consider the case when S_j^i becomes empty. Thus, to ensure the forward secrecy, the $MGKM$ starts by generating K_R. Then, using the KDF, it computes $K_{G_i}^+$, $K_{S_j^i}^+$, four $KEKs$ (Formulas 4, 5 and 7 to 10) and updates the service keys u held (Formula 6). Next, the $MGKM$ removes the node key and tokens of u and uses K_R and H to update the tokens it knows (Formulas 11 and 12). Finally, the $MGKM$ broadcasts the messages $LM1$ to $LM3$ and discards K_R and the $KEKs$. The message $LM1$ is sent to the nodes of S_j^i and is encrypted using the $KEKs$ generated from the tokens of u (K_F and K_B). The message $LM2$ is intended to those belonging to the other subgroups of G_i and is then encrypted using the $KEKs$ generated from the tokens of S_j^i (K_F^S and K_B^S). For each group G_s ($G_s \neq G_i$) which shares at least a service with G_i ($A_s \cap A_i \neq \emptyset$), $LM3$ is sent encrypted by means of K_{G_s}.

$$K_F = KDF(t_u^F) \quad (7) \qquad K_B = KDF(t_u^B) \quad (8)$$

$$K_F^S = KDF(st_{S_j^i}^F) \quad (9) \qquad K_B^S = KDF(st_{S_j^i}^B) \quad (10)$$

$$t^+ \leftarrow H(t||K_R) \quad (11) \qquad st^+ \leftarrow H(st||K_R) \quad (12)$$

$$LM1 : MGKM \rightarrow S_j^i :< nid_u, \{K_R\}\, K_F, \{K_R\}\, K_B >$$

$$LM2 : MGKM \rightarrow G_i :< sid_{S_j^i}, \{K_R\}\, K_F^S, \{K_R\}\, K_B^S >$$

$$LM3 : MGKM \rightarrow G_s :< \{A_s \cap A_i, K_R\}\, K_{G_s} >$$

Upon receiving $LM1$, a node v in S_j^i ($v \neq u$) computes either K_F, if $nid_v < nid_u$, or K_B otherwise (Formulas 7 or 8) and retrieves K_R. Then, it removes either t_u^B or t_u^F, computes $K_{G_i}^+$ and $K_{S_j^i}^+$ and updates the service keys and tokens u knows (Formulas 4 to 6, 11 and 12). Upon receiving $LM2$, a node in S^i ($S^i \in G_i$ and $S^i \neq S_j^i$) computes either K_F^S, if $sid_{S^i} < sid_{S_j^i}$, or K_B^S otherwise (Formula 9 or 10) and retrieves K_R. Then, it computes $K_{G_i}^+$ and updates the service keys and the subgroup tokens u knows (Formulas 4, 6 and 12). Upon receiving $LM3$, a node in G_s uses K_{G_s} to retrieve K_R and updates the shared service keys (Formula 6).

3) Recovering from collusion attack: In case of collusion attack, multiple compromised nodes share their information to regain access to the group and service keys. A subgroup, a group or a service is compromised if it contains at least one compromised node. To recover from it, the $MGKM$ starts by generating K_R, removing the tokens of the evicted nodes and updating those they know (Formulas 11 and 12). Then, it computes a new key for each compromised group or service (Formulas 4 and 6). Regarding subgroups, three cases arise.

a) Rekeying a compromised subgroup S_j^i: The $MGKM$ determines first its eldest (u_e) and youngest (u_y) compromised members. By construction, the tokens $t_{u_e}^F$ and $t_{u_y}^B$ remain secret for the evicted nodes and are used to generate two $KEKs$ (Formulas 7 and 8). The $MGKM$ utilizes then these keys to encrypt the message $RM1$ and broadcasts it. However, non evicted nodes, that are junior cognates of u_e and elder cognates of u_y, hold only compromised tokens. Thus, for each of them, the $MGKM$ sends the unicast message $RM2$ encrypted by the node key. If a non compromised node in S_j^i receives $RM1$, it calculates one of the $KEKs$ and uses it to retrieve K_R. On the other hand, if the node receives $RM2$, it directly uses its secret key to retrieve K_R . In both cases, the node utilizes K_R to compute $K_{G_i}^+$ as well as $K_{S_j^i}^+$ and to update the service keys it knows (Formulas 4 to 6). Finally, it removes the tokens of its evicted cognates and updates those they held (Formula 11).

b) Rekeying non compromised subgroups of a compromised group G_i: The $MGKM$ determines first the eldest (S_e^i) and youngest (S_y^i) compromised subgroups of G_i. By construction, the tokens $st_{S_e^i}^F$ and $st_{S_y^i}^B$ remain secret for the evicted nodes and are used to generate two $KEKs$ (Formulas 9 and 10). The $MGKM$ utilizes then these keys to encrypt the message $RM3$ and broadcasts it. However, nodes of the non compromised subgroups, which are junior kindreds of S_e^i and elder kindreds of S_y^i, hold only compromised subgroup tokens. Thus, for each of them, the $MGKM$ broadcasts $RM4$ encrypted by means of the subgroup key. If a node of a non compromised subgroup receives $RM3$, it calculates one of the $KEKs$ and uses it to retrieve K_R. Otherwise, if the node receives $RM4$, it directly uses its subgroup key to retrieve K_R . In both cases, the node utilizes K_R to compute $K_{G_i}^+$ and update the service keys it knows (Formulas 4 and 6). Finally, it updates all the compromised subgroup tokens (Formula 12).

c) Rekeying non compromised subgroups with compromised services: Although a subgroup belongs to a non compromised group G_s, its members can share some services with the compromised ones. In this case, the $MGKM$ broadcasts the message $RM5$ encrypted by means of the group key, K_{G_s}. Upon receiving the message, a node in G_s retrieves K_R and updates its compromised service keys.

$$RM1 : MGKM \rightarrow S_j^i :< uid_{u_e}, uid_{u_y}, \{K_R\}\, K_F, \{K_R\}\, K_B >$$
$$RM2 : MGKM \rightarrow u :< uid_{u_e}, uid_{u_y}, \{K_R\}\, K_u >$$
$$RM3 : MGKM \rightarrow G_i :< sid_{S_e^i}, sid_{S_y^i}, \{K_R\}\, K_F^S, \{K_R\}\, K_B^S >$$
$$RM4 : MGKM \rightarrow S^i :< sid_{S_e^i}, sid_{S_y^i}, \{K_R\}\, K_{S^i} >$$
$$RM5 : MGKM \rightarrow G_s :< \{K_R\}\, K_{G_s} >$$

4) Rekying upon group changing: When an actual member u subscribes or unsubscribes from services, it is moved from a group G_p to another G_n. The choice of G_n depends on the new combination of services A_n in which u participates.

As u leaves the subgroup S_j^p of G_p, the cryptographic material it holds needs to be changed. By construction, the tokens t_u^F, t_u^B, $st_{S_j^p}^F$ and $st_{S_j^p}^B$ remain secret for u and are used to rekey G_p. To ensure the forward secrecy, the $MGKM$ generates K_R, uses the KDF to compute $K_{G_p}^+$, $K_{S_j^p}^+$ as well as four $KEKs$ (Formulas 4, 5 and 7 to 10) and updates the tokens u knows (Formulas 11 and 12). Also, as u joins the subgroup S_j^n of G_n, the $MGKM$ determines its new ID and backward token. To ensure the backward secrecy, the $MGKM$ uses the KDF to generate a new t_u^F, to compute $K_{G_n}^+$ and $K_{S_j^n}^+$ (Formulas 3 to 5) and to update the key of each service E_k to which u subscribes or unsubscribes (Formula 6). Finally, the $MGKM$ broadcasts the messages $EM1$ to $EM5$.

The message $EM1$ is sent to the nodes of S_j^p and is encrypted using the $KEKs$ generated from the tokens of u. On the other hand, $EM2$ is intended to those of the other subgroups of G_p and is encrypted using the $KEKs$ generated from the tokens of S_j^p. $EM3$ is sent to the nodes of S_j^n and $EM4$ to those of the other subgroups of G_n. They are then encrypted using $K_{S_j^n}$ and K_{G_n}, respectively. For each group G_s ($G_s \neq G_p$ and $G_s \neq G_n$) participating in a service E_k, the $MGKM$ broadcasts the message $EM5$ encrypted using K_{G_s}. Finally, a last message is used to provide u with its new cryptographic material and is encrypted by means of K_u.

$$EM1 : MGKM \rightarrow S_j^p :< nid_u, \{K_R\}\, K_F, \{K_R\}\, K_B >$$
$$EM2 : MGKM \rightarrow G_p :< sid_{S_j^p}, \{K_R\}\, K_F^S, \{K_R\}\, K_B^S >$$
$$EM3 : MGKM \rightarrow S_j^n :< nid_u, \{t_M, K_R\}\, K_{S_j^n} >$$
$$EM4 : MGKM \rightarrow G_n :< \{K_R\}\, K_{G_n} >$$
$$EM5 : MGKM \rightarrow G_s :< \{LE_s, K_R\}\, K_{G_s} >$$

Upon receiving $EM1$, a node v in S_j^p ($v \neq u$) computes either K_F, if $nid_v < nid_u$, or K_B otherwise (Formula 7 or 8) and retrieves K_R. Then, it computes $K_{G_p}^+$ and $K_{S_j^p}^+$ (Formulas 4 and 5), updates the tokens u knows (Formulas 11 and 12) and removes either t_u^B or t_u^F. Moreover, v updates the keys of services from which u unsubscribed (Formula 6). Upon receiving $LM2$, a node in S^p ($S^p \in G_p$ and $S^p \neq S_j^p$) computes either K_F^S, if $sid_{S^p} < sid_{S_j^p}$, or K_B^S otherwise (Formula 9 or 10) and retrieves K_R. Then, it computes $K_{G_p}^+$ and updates the subgroup tokens u knows (Formulas 4 and 12). Also, the node updates the keys of services from which u unsubscribed (Formula 6). Upon receiving $EM3$, a node in S_j^n uses $K_{S_j^n}$ to retrieve t_M and K_R and computes t_u^F, $K_{G_n}^+$ and $K_{S_j^n}^+$ (Formulas 3 to 5). Also, the node updates the keys of services to which u subscribed (Formula 6). Upon receiving $EM4$, a node in S^n ($S^n \in G_n$ and $S^n \neq S_j^n$) uses K_{G_n} to retrieve K_R, computes $K_{G_n}^+$ (Formula 4) and updates the keys of services to which u subscribed (Formula 6). Finally, upon receiving $JM5$, a node in G_s uses K_{G_s} to retrieve K_R and the list of the shared services, LE_s, and updates their keys (Formula 6).

IV. Security analysis

Before a node u joins a subgroup S_j^i of a group G_i, the $MGKM$ rekeys both of them as well as all services involved. To achieve this, the $MGKM$ uses the messages $JM1$ to $JM3$ (if u joins the network) or $EM3$ to $EM5$ (when it changes group). The messages $JM1$ and $EM3$ allow the members of S_j^i to calculate the new subgroup, group and service keys. They are encrypted using the actual subgroup key, $K_{S_j^i}$. On the other hand, $JM2$ and $EM4$ enable the other nodes of G_i to calculate the new group and service keys. They are then encrypted using the actual group key, K_{G_i}. Finally, $JM3$ and $EM5$ allow the members of the other groups which participate to the same services as u to calculate the new service keys. They are encrypted by means of their group keys. At the end of the process of rekeying, the $MGKM$ provides u with the new keys, via a secure channel (if u joins the network) or using a message encrypted by means of K_u (if u changes group). The network is rekeyed and u gets the new keys without access to the old ones. The backward secrecy is then guaranteed.

After a node u leaves a subgroup S_j^i of a group G_i, the $MGKM$ rekeys both of them as well as all the services involved. To achieve this, the $MGKM$ uses the messages $LM1$ to $LM3$ (if u has left the network), the messages $EM1$, $EM2$ and $EM5$ (when it changes group) and the messages $RM1$ to $RM5$ (in case of collusion attack). The messages $LM1$, $EM1$, $RM1$ and $RM2$ allow the members of S_j^i to calculate the new subgroup, group and service keys. They are encrypted using their node keys or the $KEKs$ generated from the tokens of u. On the other hand, $LM2$, $EM2$, $RM3$ and $RM4$ enable the other nodes of G_i to calculate the new group and service keys. They are then encrypted using their subgroup keys or the $KEKs$ generated from the tokens of S_j^i. Finally, $LM3$, $EM5$ and $RM5$ allow the members of the other groups which participate to the same services as u to calculate the new service keys. They are encrypted by means of their group keys. By construction, u does not know any of the keys used to encrypt these messages and is then excluded from the process of rekeying. Since it cannot get access to the new keys or any future incarnation of them, the forward secrecy is guaranteed.

In our solution, the compromise of a node does not affect the services in which it does not participate. This is due, firstly, to the fact that the members of the same group participate to the same services and, secondly, because nodes belonging to different groups do not share any security parameter if they have no service in common. Let us consider two groups G_i and G_j associated to the combinations A_i and A_j, respectively. When a node u of the group G_i gets compromised, only the service keys of A_i are exposed. If G_i and G_j share some services ($A_i \cap A_j \neq \emptyset$), the keys of services to which the members of G_j participate but not those of G_i ($A_i \backslash A_j$) remain secret. Indeed, u does not know them. Furthermore, if the groups do not share services ($A_i \cap A_j = \emptyset$), any of the service keys of A_j gets compromised. In both cases, only the services in which u participates are compromised. Thus, the compromise of a service has no effect on the others.

V. Performance evaluation

In the following, we use the notations m_i^j, p_i and k_i to refer to the number of nodes in S_j^i and that of subgroups and services in G_i. In a collusion attack, we consider that c nodes are evicted. We assume that KDF and H require the same computing cost and that keys and tokens have the same size.

A. Overheads on the MGKM

In a network of n nodes, the $MGKM$ stores n node keys, n subgroup keys, max_g group keys, e service keys, $2.n$ node tokens and $2.n$ subgroup tokens, in the worst case. If we assume that $e \leq n$, the storage will be of the order of $O(n)$.

When a node u joins the subgroup S_j^i of G_i, the $MGKM$ generates K_R, t_M, K_u, t_u^F, t_u^B and updates $K_{S_j^i}$, K_{G_i} and k_i service keys. It performs then $k_i + 7$ hash function executions. It also encrypts and sends messages $JM1$, $JM2$ and, for each group sharing services with G_i, the message $JM3$. As the number of groups reaches max_g, in the worst case, the $MGKM$ can encrypt and send up to $max_g + 1$ messages.

When u leaves the network, the $MGKM$ generates K_R, 4 $KEKs$ and updates $K_{S_j^i}$, K_{G_i}, k_i service keys, $m_j^i - 2$ node tokens and $p_i - 1$ subgroup tokens. It performs then $p_i + m_i^j + k_i + 4$ hash function executions. It also encrypts and sends the messages $LM1$, $LM2$ and, for each group which shares services with G_i, the message $LM3$. The $MGKM$ encrypts and sends then $max_g + 1$ messages in the worst case.

When u changes group, the $MGKM$ generates K_R, t_M and 4 $KEKs$, updates 2 group keys, 2 subgroup keys, $m_j^i - 2$ node tokens, $p_i - 1$ subgroup tokens and the k_u service keys to which the node subscribes or unsubscribes. It performs then $p_i + m_i^j + k_u + 7$ hash function executions. It also encrypts and sends the messages $EM1$ to $EM4$ and, for each group associated with a compromised service, the message $EM5$. The $MGKM$ encrypts and sends then $max_g + 3$ messages in the worst case.

The worst case for a recovery from collusion attack is when all nodes are rekeyed using the unicast message $RM2$ (all subgroups and all their node tokens are compromised). In this case, the $MGKM$ encrypts and sends $n - c$ messages. It also updates all subgroup, group and service keys. The communication and computing are then of the order of $O(n)$.

To sum up, the storage overheads for the $MGKM$ are of the order of $O(n)$. Also, the communication and computing costs are proportional to max_g (if a node joins, leaves or changes group) or of the order of $O(n)$ (in the case of collusion attack). Figure 4 illustrates the variation of max_g according to n and e. However, in our analysis, we consider the worst case whose probability of occurring is low. The costs are much lighter in the general case. Moreover, this is usually not a problem in practice, since the $MGKM$ has plentiful of resources. It is on the side of nodes that the protocol must be light.

B. Overheads on nodes

A node u, belonging to a subgroup S_j^i in G_i, stores K_u, $K_{S_j^i}$, K_{G_i}, k_i service keys, $m_i^j - 1$ node tokens and $p_i - 1$ subgroup tokens. It holds then $p_i + m_i^j + k_i + 1$ keys.

The worst case for u when a node v joins G_i (v joins the network or changes group) is that v is assigned to S_j^i. The node receives then one message, decrypts two keys, generates t_v^F and updates $K_{S_j^i}$, K_{G_i} and k_i service keys. It performs then 2 decryptions and $k_i + 3$ hash function executions.

The worst case for u when a node v leaves G_i (v leaves the network or changes group) is that they are cognates. The node u receives then one message, decrypts one key, calculates one KEK and updates $K_{S_j^i}$, K_{G_i}, k_i service keys, $m_i^j - 2$ node tokens and $p_i - 1$ subgroup tokens. It performs then one decryption and $p_i + m_i^j + k_i$ hash function executions.

When a recovery from collusion attack is performed, the worst case for a node u is that some of its cognates are evicted. The node receives then one message, decrypts one key and updates $K_{S_j^i}$, K_{G_i}, k_i service keys, $m_i^j - c - 1$ node tokens and $p_i - 1$ subgroup tokens. The node performs then one decryption and $p_i + m_i^j + k_i - c$ hash function executions.

To sum up, the communication for nodes is constant ($O(1)$) and the storage and calculation are proportional to $p_i + m_i^j + k_i$. If the number of services is negligible when compared to the size of the network and if the n nodes of the network are uniformly distributed (i.e. $\forall i$, $\forall j$, $p_i \simeq m_i^j \simeq \sqrt{n}$), the costs will be on average of the order of $O(\sqrt{n})$. Our solution is then efficient and scalable as, even if n increases, the protocol is affordable for constrained nodes.

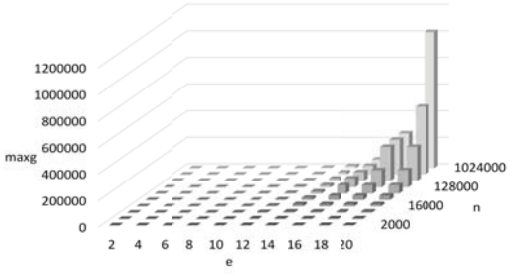

Fig. 4: Variation of max_g according to n and e.

VI. Conclusion

In this paper, we presented a highly scalable Multi-Group Key Management protocol for IoT which ensures the forward and backward security, efficiently recovers from collusion attacks and guarantees the secure coexistence of several services in the network. Our protocol manages several groups having independent security parameters. Each group is then associated to a combination of services. When a node joins the network, the protocol assigns it to the group associated to the combination of services to which it subscribes. When an actual member subscribes or unsubscribes from services, it migrates from a group to another according to the new combination of services. Our protocol efficiently rekeys groups, subgroups and services after a node joins or leaves the network, changes group or when a set of colluding nodes are evicted. In future works, we intend to decentralize the protocol. Cryptographic material will then be spread across more than one entity in order not to have a single point of failure and to make it more difficult to access or modify this secret material.

Acknowledgments

This work was carried out and funded by Heudiasyc UMR CNRS 7253 and the Labex MS2T.

References

[1] M. Alagheband and M.R. Aref. “Dynamic and secure key management model for hierarchical heterogeneous sensor networks”. In: *IET Information Security* 6.4 (2012), pp. 271–280.

[2] R. Azarderakhsh, A. Reyhani-Masoleh and Z. Abid. “A key management scheme for cluster based wireless sensor networks”. In: *Embedded and Ubiquitous Computing, 2008. EUC’08. IEEE/IFIP Int. Conf. on*. Vol. 2. IEEE. 2008, pp. 222–227.

[3] K. Chatterjee, A. De and D. Gupta. “An improved ID-Based key management scheme in wireless sensor network”. In: *Int. Conf. in Swarm Intelligence*. Springer. 2012, pp. 351–359.

[4] C. Chen, Z. Huang, Q. Wen and Y. Fan. “A novel dynamic key management scheme for wireless sensor networks”. In: *Broadband Network and Multimedia Technology (IC-BNMT), 2011 4th IEEE International Conference on*. IEEE. 2011, pp. 549–552.

[5] G. Dini and I.M. Savino. “S2rp: a secure and scalable rekeying protocol for wireless sensor networks”. In: *Mobile Adhoc and Sensor Systems, IEEE Int. Conf. on*. 2006, pp. 457–466.

[6] A. Diop, Y. Qi and Q. Wang. “Efficient group key management using symmetric key and threshold cryptography for cluster based wireless sensor networks”. In: *International Journal of Computer Network and Information Security* 6.8 (2014), p. 9.

[7] R. Divya and T. Thirumurugan. “A novel dynamic key management scheme based on hamming distance for wireless sensor networks”. In: *Computer, Communication and Electrical Technology (ICCCET), 2011 International Conference on*. IEEE. 2011, pp. 181–185.

[8] D. Mall, K. Konaté and A.K. Pathan. “ECL-EKM: An enhanced Certificateless Effective Key Management protocol for dynamic WSN”. In: *Networking, Systems and Security (NSysS), 2017 International Conference on*. IEEE. 2017, pp. 150–155.

[9] M. Park, Y. Park, H. Jeong and S. Seo. “Secure multiple multicast services in wireless networks”. In: *IEEE Transactions on Mobile Computing* (2012).

[10] S.M.M Rahman and K. El-Khatib. “Private key agreement and secure communication for heterogeneous sensor networks”. In: *Journal of Parallel and Distributed Computing* 70.8 (2010), pp. 858–870.

[11] S.H. Seo, J. Won, S. Sultana and E. Bertino. “Effective key management in dynamic wireless sensor networks”. In: *IEEE Transactions on Information Forensics and Security* 10.2 (2015), pp. 371–383.

[12] S.R. Singh, A.K. Khan and T.S. Singh. “A New Key Management Scheme for Wireless Senm Networks using an Elliptic Curve”. In: *Indian Journal of Science and Technology* 10.13 (2017).

[13] J. Son, J. Lee and S. Seo. “Topological key hierarchy for energy-efficient group key management in wireless sensor networks”. In: *Wireless personal communications* 52.2 (2010), p. 359.

[14] Y. Sun and K.R. Liu. “Hierarchical group access control for secure multicast communications”. In: *IEEE/ACM Transactions on Networking* 15.6 (2007), pp. 1514–1526.

[15] M. Thoma, S. Meyer, K. Sperner, S. Meissner and T. Braun. “On iot-services: Survey, classification and enterprise integration”. In: *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*. IEEE. 2012, pp. 257–260.

[16] M. Tiloca and G. Dini. “GREP: A group rekeying protocol based on member join history”. In: *Computers and Communication (ISCC), 2016 IEEE Symposium on*. IEEE. 2016, pp. 326–333.

[17] I. Tsai, C. Yu, H. Yokota and S. Kuo. “Key Management in Internet of Things via Kronecker Product”. In: *Dependable Computing, 2017 IEEE 22nd Pacific Rim International Symposium on*. Pp. 118–124.

[18] F. Zhan, N. Yao, Z. Gao and G. Tan. “A novel key generation method for wireless sensor networks based on system of equations”. In: *Journal of Network and Computer Applications* 82 (2017), pp. 114–127.

[19] J. Zhang and V. Varadharajan. “Wireless sensor network key management survey and taxonomy”. In: *Journal of Network and Computer Applications* 33.2 (2010), pp. 63–75.

[20] Q. Zhang and Y. Wang. “A centralized key management scheme for hierarchical access control”. In: *Global Telecommunications Conference, 2004. GLOBECOM’04. IEEE.* Vol. 4. 2004, pp. 2067–2071.