A Key Management Protocol for Secure Device-to-Device Communication in the Internet of Things

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

▶ **To cite this version:**

Mohamed Ali Kandi, Hicham Lakhlef, Abdelmadjid Bouabdallah, Yacine Challal. A Key Management Protocol for Secure Device-to-Device Communication in the Internet of Things. IEEE Global Communications Conference (GLOBECOM 2019), Dec 2019, Waikoloa, United States. pp.1-6, 10.1109/globecom38437.2019.9013595 . hal-03280892

HAL Id: hal-03280892
https://hal.science/hal-03280892

Submitted on 7 Jul 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Key Management Protocol for Secure Device-to-Device Communication in the Internet of Things

Mohamed Ali Kandi[1], Hicham Lakhlef[1], Abdelmadjid Bouabdallah[1] and Yacine Challal[1,2]
[1]Sorbonne Universités, Université de Technologie de Compiègne, CNRS, UMR7253 Heudiasyc-CS 60319-60203 Compiègne Cedex, France
[2]Laboratoire de Méthodes de Conception de Systèmes, École nationale Supérieure d'Informatique, Algiers, Algeria
[2]Centre de Recherche sur l'Information Scientifique et Technique, Algiers, Algeria
Email: $\{mohamed-ali.kandi, hicham.lakhlef, madjid.bouabdallah, yacine.challal\}$ @$hds.utc.fr$

***Abstract*—The Internet of Things (IoT) is a network made up of a large number of devices which are able to automatically communicate in a Peer-to-Peer manner. The aim is to provide various services for the benefit of society. One of the main challenges facing the IoT is how to secure this Device-to-Device communication. Among all the security issues, the Key Management is one of the most difficult. This is mainly due to the fact that most of these devices have limited resources in terms of storage, calculation, communication and energy. Although different approaches have been proposed to deal with this problem, each of them presents its own limitations and weaknesses. In this paper, we propose a novel Key Management protocol for Device-to-Device communication in the Internet of Things. Compared to the existing Peer-to-Peer schemes, our solution provides the best compromise between the IoT requirements: resilience, connectivity, efficiency, scalability and flexibility. To achieve this balance, the network members are uniformly distributed into logical sets. A device shares then a distinct pairwise key with each member of its set and a unique pairwise set key with the members of each of the other sets. We then prove that our solution is resilient as the capture of a member compromises a negligible part of a large network. Moreover, we show that our scheme has a good network connectivity. It is then efficient as it does not require additional calculation or communication costs on the network members. We also demonstrate that our protocol is scalable as storage cost on the network members does not significantly increase when the network gets larger. We finally show that our solution is flexible.**

***Index Terms*—Internet of Things, Device-to-Device communication, Security, Peer-to-Peer Key Management.**

I. Introduction

The number of devices connected to the 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 the Internet but about the Internet of Things (IoT). This emerging technology gives rise to revolutionary applications such as health care, environment monitoring, smart homes, smart cities, autonomous vehicles...etc. The IoT devices are able to automatically communicate to each other in a Peer-to-Peer manner. The Vehicle-to-Vehicle communication, for example, allows vehicles to exchange information about their speed and position to avoid crashes, ease traffic congestion and improve the environment. One of the main challenges facing the IoT is how to secure this Device-to-Device communication. Among all the security issues, the Key Management is one of the most difficult. This is mainly due to the fact that most of these devices have limited resources in terms of storage, calculation, communication and energy [1].

The Key Management (KM) is the core of secure communication. Its main role is to establish secure links between the network members [11]. To achieve this, the KM provides them with secret cryptographic keys that are used to encrypt and decrypt the exchanged data. According to the encryption technique used, the KM approaches can be classified into two categories: symmetric and asymmetric. Symmetric schemes involve the use of the same key for encryption and decryption, while asymmetric approaches use two different keys. Asymmetric approaches usually imply intensive computing, which makes them impractical on the IoT constrained devices [15].

Although symmetric schemes are more suitable for the IoT constrained devices, they rarely meets all its requirements: resilience (Rsl), connectivity (Cnt), efficiency (Efc), scalability (Scl) and flexibility (Flx). A protocol is resilient if the capture of a node does not jeopardize the communications of the other ones. When network connectivity is low, some neighboring communicators may not share a common key and relay on intermediate nodes to establish a secure link. Efficiency implies minimal use of node resources in terms of storage, calculation and communication. If increasing the network size does not significantly degrade its performance, the protocol is scalable. Finally, a flexible solution operates well regardless of nodes location and supports their dynamic deployment [6].

In this paper, we propose a novel KM protocol for Device-to-Device communication in the IoT. Compared to the existing Peer-to-Peer schemes, it provides the best compromise between the IoT requirements mentioned above. To achieve this balance, the network members are uniformly distributed into logical sets. A device shares then a distinct pairwise key with each member of its set and a unique pairwise set key with the members of each of the other sets. We then prove that our solution is resilient as the capture of a member compromises a negligible part of a large network. Moreover, we show that our scheme has a good connectivity. It is then efficient as it does not require additional calculation or communication on the devices. We also prove that our protocol is scalable as storage cost does not significantly increase when the network gets larger. We finally show that our solution is flexible.

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 our work in Section VI.

II. Related Works

In this work, we focus on the symmetric schemes since they require less resources than the asymmetric ones. This makes them more suitable for the IoT constrained devices [15]. Symmetric schemes can in turn be classified into three sub-categories: deterministic, pure probabilistic and deployment knowledge based schemes.

A. Deterministic schemes

The basic deterministic scheme (Pairwise Key Protocol) consists of storing a distinct pairwise key in each pair of nodes before their deployment. Since such a technique [8] requires a lot of storage, other approaches were proposed: Polynomial-based protocols [2, 4], Matrix-based schemes [9, 14]...etc. The deterministic approaches have the advantage of being resilient against node capture. Moreover, they are usually efficient and have a good connectivity. However, they suffer from poor scalability (Figure 1a). Indeed, Pairwise Key schemes require that a node stores as many keys as there are members in the network. On the other hand, the larger is the network, the more vulnerable the Polynomial and Matrix-based approaches are to compromise. Finally, most of these schemes lack flexibility as they are based on key pre-distribution.

B. Pure probabilistic schemes

The first pure probabilistic scheme was introduced in [10]. It consists of using a large pool of keys and to randomly distribute some of them (a key ring) to each network member. Two neighboring nodes can then communicate only if they share a common key. Otherwise, they relay on intermediate nodes to establish secure links. Other methods were proposed to enhance the resilience. Using the Q-composite [5] scheme, nodes can communicate only if they share Q keys. Also, polynomial pool based schemes [13, 18] use a pool of polynomials instead of keys. Probabilistic schemes are resilient and more scalable than the deterministic ones. Nevertheless, they suffer from poor flexibility, efficiency and connectivity (Figure 1b). Indeed, they are usually based on key pre-distribution. Moreover, intermediate nodes may be necessary to establish secure links. This requires additional calculation and communication and thereby more energy consumption [17]. Some works tried to enhance the connectivity using the unital design theory [3], system of equations [17]...etc. However, as long as they are probabilistic, the connectivity is rarely total.

C. Deployment knowledge based schemes

These schemes are neither deterministic nor purely probabilistic. They are based on the location of nodes to maximize the connectivity. Thus, to increase the probability of sharing keys, nodes are distributed into regional zones. Key rings are then assigned to them so that neighboring nodes share a maximum of keys. Like the other approaches, the deployment knowledge based schemes can use pairwise keys [7], polynomials [12] or matrices [16]. These approaches are resilient, scalable and provide a better network connectivity than the pure probabilistic schemes. However, they are not flexible and are more suitable for static networks (Figure 1c).

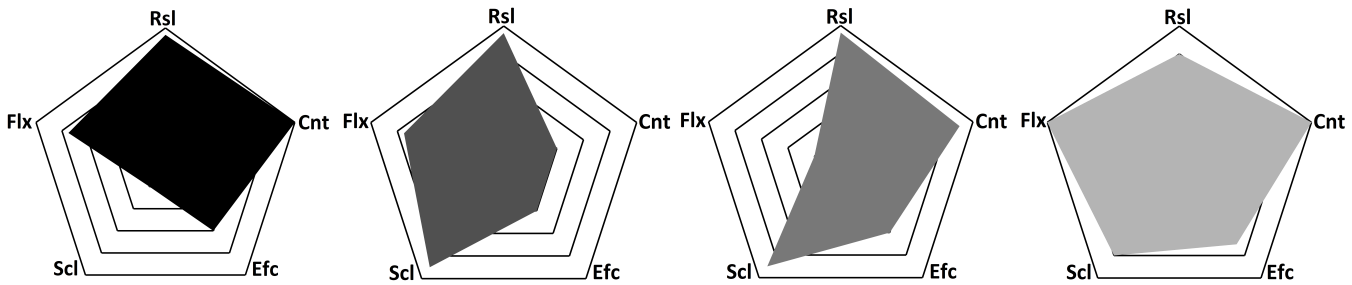
III. Our solution

Our literature review shows that none of the existing solutions meets all the IoT requirements. To improve the scalability of deterministic schemes without loss of efficiency or connectivity, as it is the case with probabilistic schemes, our solution uniformly distributes the network members into logical sets. A node shares then a distinct pairwise key with each member of its set and a unique pairwise set key with the members of each of the other sets. The scalability of the protocol is then improved as nodes store fewer keys. Although the members of a set share the same pairwise set key, we prove that our solution remains resilient against node capture. Unlike deployment knowledge schemes, our protocol operates well regardless of the position of nodes. Moreover, as keys are dynamically distributed to the network members when nodes join or leave the network, our solution is flexible (Figure 1d).

Since some keys are shared by several nodes, the KM must ensure that they are known only by the current members. Thus, when a node joins or leaves the network, these keys are revoked and new ones are distributed to the remaining ones. This rekeying ensures that a joining node will not have access to the old keys (backward secrecy) and a leaving member will no longer know the future ones (forward secrecy).

A. Overview

The network members are distributed into logical sets to improve the protocol scalability. To each set S is associated a unique ID, sid_S. It is important to note that these grouping is logical and transparent to the application layer. Although nodes belonging to the same set are considered as cognates, they can be physically far from one another. Each of them, u, is also associated with a unique ID, nid_u.



(a) Deterministic schemes. (b) Pure probabilistic schemes. (c) Deployment knowledge. (d) Our solution.

Fig. 1: Symmetric Peer-to-Peer Key Management approaches.

The keys managed by our solution can be classified into two types: Data Encryption Keys ($DEKs$) and Key Encryption Keys ($KEKs$). The $DEKs$ are symmetric pairwise keys that are used by nodes to encrypt the data exchanged between them. The KEK are used to secure the communications between the KM and the nodes in order to protect the $DEKs$ and thereby ensure the backward and forward secrecy. A node u, belonging to a set S, holds the following keys:

- A pairwise node key, K_u^v, for each of its cognates v. It is a DEK allowing the two nodes to communicate safely.
- A pairwise set key, K_S^T, for each set T ($T \neq S$). This is a DEK that allows the members of S to communicate with those of T safely. This key is used, instead of the pairwise node keys, to reduce the storage overhead on nodes and thereby improve the protocol scalability.
- A node key, K_u, which is a KEK known only to u. It allows the KM to communicate with the node safely.
- A set key, K_S, which is a KEK known to the members of S only. This key is used, instead of the node keys, when the KM sends the same message to all the members of S. The latter will then be encrypted once for more efficiency.

Figure 2 and Table I show an example of distribution of the nodes of a network N and the keys they know. Hereafter, the keys K_u^v and K_v^u are the same and can be used interchangeably. The same goes for the keys K_S^T and K_T^S. We also use the notations K_R and KDF to refer to a refresh key and a pseudo-random key derivation function, respectively. Finally, $KM \rightarrow R :< \{M\}_K >$ means that the KM sends the message $\{M\}$, encrypted using the key K, to the node(s) R.

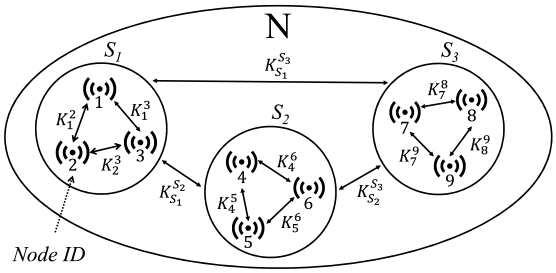


Fig. 2: Example of a distribution of nodes.

Node ID	Node key	Pairwise node keys	Set key	Pairwise set keys
1	K_1	K_1^2, K_1^3	K_{S_1}	$K_{S_1}^{S_2}, K_{S_1}^{S_3}$
2	K_2	K_2^1, K_2^3		
3	K_3	K_3^1, K_3^2		
4	K_4	K_4^5, K_4^6	K_{S_2}	$K_{S_2}^{S_1}, K_{S_2}^{S_3}$
5	K_5	K_5^4, K_5^6		
6	K_6	K_6^4, K_6^5		
7	K_7	K_7^8, K_7^9	K_{S_3}	$K_{S_3}^{S_1}, K_{S_3}^{S_2}$
8	K_8	K_8^7, K_8^9		
9	K_9	K_9^7, K_9^8		

TABLE I: Example of keys held by nodes.

B. Rekeying upon joining

Let us consider a node u joining the network. The node is first assigned to a set S following the steps described in section III-D. The KM generates then some new keys and updates some of the previously existing ones. The aim of this update is to ensure the backward secrecy. Indeed, if these keys are not updated and if the joining node u has stored the messages previously exchanged, it will be able to decipher some of them. Next, the KM provides some nodes with the new keys and sends to others the elements allowing them to update some of the keys they hold. The process of rekeying upon joining consists of the four following steps.

1) Key generation: The first step in the rekeying process consists of determining the secret key, K_u, of the joining node u. After that, the KM generates a pairwise node key, K_u^v, for each node v of the set S. The KM also determines the unique node ID, nid_u, associated to u.

2) Key update: The KM starts by randomly generating K_R. Then using it and the KDF, the KM updates the set key of S and the pairwise set keys known by its members (Formulas 1 and 2, respectively). As previously said, the aim of this update is to guarantee the backward secrecy.

$$K_S^+ = KDF(K_S||K_R) \quad (1)$$

$$K_S^{T+} = KDF(K_S^T||K_R), \forall T \in N \quad (2)$$

3) Key distribution: After the keys generation and update are completed, the KM distributes these new keys to the appropriate nodes. Thus, it sends to each node v of the set S the unicast message $JM1$ encrypted by means of the node secret key, K_v. The message contains the ID of the joining node and the pairwise node key, K_u^v, associated to it. The KM also broadcasts for each set T (including S) the message $JM2$ encrypted using K_T, the current set key of T. The message contains the ID of the set S and K_R. Finally, the KM provides u, via a pre-existing secure channel, with its secret key, the new set key, the pairwise node keys to share with its cognates and all the new pairwise set keys associated to S. After the keys distribution, the KM discards K_R.

$$JM1 : KM \rightarrow v :< \{nid_u, K_u^v\}_{K_v} > (\forall v \in S)$$

$$JM2 : KM \rightarrow T :< \{sid_S, K_R\}_{K_T} > (\forall T \in N)$$

4) Key installation: When a member of the set S, v, receives the messages $JM1$ and $JM2$, it first decrypts them using its secret and set keys, respectively. Then, it installs K_u^v as the pairwise key to use for encrypting the communications with the joining node u. The node v also uses K_R and the KDF to update the set key and all the pairwise set keys it knows (Formulas 1 and 2, respectively). After that, v discards K_R. On the other hand, when a node w, not belonging to S, receives $JM2$, it first decrypts the message, using the current set key, and retrieves K_R. Then, using the KDF, it updates the pairwise set key it shares with the members of S (Formula 2). Once done, w discards K_R.

C. Rekeying upon leaving

A node u can leave the network or be evicted when it get compromised. In both cases, the keys it knows must be revoked. The KM removes then some of them and updates some others. The aim of this update is to ensure the forward secrecy. Indeed, if these keys are not updated, the leaving node will be able to decipher some of the future communications. Next, the KM provides the network members with the elements allowing them to remove the keys that should be removed and to update those that must be updated. The process of rekeying upon leaving consists of the four following steps.

1) Key removal: The KM starts by removing the secret key, K_u, of the leaving node as well as its ID, nid_u. Next, the KM deletes all its pairwise keys, K_u^v ($v \in S$, $v \neq u$).

2) Key update: The KM starts by randomly generating K_R. Then using it and the KDF, the KM updates the set key of S and all the pairwise set keys known by its members (Formulas 1 and 2, respectively). As previously said, the aim of this update is to guarantee the forward secrecy.

3) Key distribution: After the keys removal and update are completed, the KM distributes the new keys to the appropriate nodes. Thus, it sends, to each node v of the set S, the unicast message $LM1$ encrypted by means of the node key, K_v. The message contains the ID of the leaving node and K_R. The KM also broadcasts, for each set T ($T \neq S$), the message $LM2$ to provide its members with K_R. The message $LM2$ is encrypted using K_T, the current set key of T. The message $LM2$ is not sent to the members of S because the leaving node u knows the set key K_S. The refresh key is therefore sent to the other members of S via the unicast message $LM1$ instead. After the keys distribution, the KM discards K_R.

$$LM1 : KM \rightarrow v :< \{nid_u, K_R\}_{K_v} > (\forall v \in S, v \neq u)$$
$$LM2 : KM \rightarrow T :< \{sid_S, K_R\}_{K_T} > (\forall T \in N, T \neq S)$$

4) Key installation: When a member of S, v, receives $LM1$, it first decrypts the message, using its secret key K_v, and retrieves K_R. Then, it removes the pairwise key K_u^v, which was used for encrypting the communications with the leaving member u. The node v also uses the KDF to update the set key and all the pairwise set keys it knows (Formulas 1 and 2, respectively). Once done, the node v discards K_R. On the other hand, when a node w, not belonging to the set S receives $LM2$, it first decrypts the message, using the current set key, and retrieves K_R. Then, using it and the KDF, the node updates the pairwise set key it shares with the members of the set S (Formula 2). Finally, the node w discards K_R.

D. Set management

The set management consists of distributing nodes on sets while minimizing the number of keys they store. The aim is to improve the protocol scalability without significant loss of resilience, efficiency nor network connectivity. In the following, we use the notations n and p to refer to the number of nodes and sets in the network, respectively. We also denote the number of the members of a set S by m_s.

A node stores one secret key, $m_s - 1$ pairwise node keys, one set key and $p - 1$ pairwise set keys. Storage on nodes is therefore proportional to $p + m_s$. The problem consists then of creating sets and assigning nodes to them so as to satisfy:

$$\forall S, min\,(p + m_s) \tag{3}$$
$$\sum_{s=1}^{p} m_s = n \tag{4}$$

To have the same number of keys stored on each network member, we opted for a uniform distribution (i.e. $\forall S, m_s = m$). By replacing 4 in 3 and studying the monotony of the resulting function ($f(p) = p + \frac{n}{p}$), we can easily show that storage is minimized when $p = m = \sqrt{n}$. The set management aims then to uniformly distributes the n nodes of the network into $\sqrt{n}$ sets of $\sqrt{n}$ members each (Figure 2).

The Assignment Algorithm (Algorithm 1) is run when nodes join the network and assigns them to the right sets. It takes as input n, the current number of network members, and assigns the joining node to a set according to the input value. The algorithm manipulates then a list of sets, ls, of size p. Each of its items contains the ID of a set and its size.

Algorithm 1: Assignment Algorithm

```
Input : n = the number of network members
Search in ls a set S such that m_s < sqrt(n);
if no set is found then
    Create a new set S;
end
Assign the joining node to S;
Update ls;
```

The Reorder Algorithm (Algorithm 2) is run, after a node leaving, to reduce the number of sets. It takes as input the size of the network, n, the percentage of merging, pcm, and tries to remove or merge sets when it is possible. Due to space constraints, we do not discuss the set merging in this paper.

Algorithm 2: Reorder Algorithm

```
Input : n = the number of network members
        pcm = percentage of merging
if m_s = 0 then Remove S ;
else
    if m_s < pcm.sqrt(n) then
        Find T such as m_t < pcm.sqrt(n);
        if a set T is found then Merge S and T ;
    end
end
Update ls;
```

IV. Security analysis

Since some keys are shared by several nodes, we start by showing that our solution fulfills the backward and forward secrecy requirements. We then prove that it provides a good level of resilience. We assume that the KM itself is secure and that only the network members can be compromised.

A. Backward and forward secrecy

The issue is to prove that a joining node cannot access the old keys and that a leaving node cannot access the new ones.

Proposition 1: Backward secrecy is guaranteed as the joining node never gets knowledge of the security material used before it joins the network.

Proof: Let us consider a node u that joins a set S of the network. The KM first updates the keys mentioned above. Then, before u can actually join the network, the KM rekeys all its current members by means of messages $JM1$ and $JM2$. The content of the former is encrypted using the nodes secret keys and that of the latter is protected by means of the set keys. Since none of these keys are known by u, the joining node is excluded from the process of rekeying.

Proposition 2: Forward secrecy is guaranteed as the leaving node does not have access to the new security material.

Proof: Let us consider a node u that leaves a set S of the network. The KM first rekeys the cognates of u, by means of the message $LM1$, then the members of the other sets, using $LM2$. The content of the former is encrypted using the nodes secret keys and that of the latter is protected by means of the set keys. Since none of these keys are known by u, the leaving node is excluded from the process of rekeying.

B. Resilience against node capture

According to [11], resilience is the measure of the impact of one captured node on the rest of the network. The issue is then to prove that, using our solution, this impact is negligible for large networks such as the IoT.

1) Theoretical analysis: We start by analyzing the rate of compromised links due to a node capture.

Lemma 1: A node can decrypt a number of links equal to:

$$D = n - 1 + (\sqrt{n} - 1)(n - \sqrt{n}) = (\sqrt{n} - 1)(n + 1) \quad (5)$$

Proof: A node can decrypt the communications linking it to the $n - 1$ other network members as well as the links between its $\sqrt{n} - 1$ cognates and the $n - \sqrt{n}$ other nodes.

Proposition 3: The percentage of links that a compromised node can decipher is equal to:

$$P = \frac{D}{T} = \frac{(\sqrt{n} - 1)(n + 1)}{\frac{n(n-1)}{2}} = \frac{2(n + 1)}{(\sqrt{n} + 1)n} \quad (6)$$

Proof: On the one hand, we have the lemma 1. On the other, the total number of links in a network of n nodes is equal to:

$$T = C_n^2 = \frac{n(n - 1)}{2} \quad (7)$$

Proposition 4: The compromise of the whole network occurs if and only if all the network members are captured.

Proof: Deciphering all the intra-subgroup communications requires the knowledge of all the pairwise node keys associated to it. This is only possible if all the subgroup members are captured. Also, deciphering all the inter-subgroup communications requires the knowledge of all the pairwise subgroup keys. This is only possible if at least a member of each subgroup is compromised. Deciphering all the communications is then possible if and only if all the network members are captured.

2) Comparison: According to [13], a perfect resilience can be reached if each pair of nodes share a distinct pairwise key. Thus, a captured node can only decipher the $n - 1$ communications linking it to the other network members. The percentage of compromised links is equal to $\frac{2(n-1)}{n(n-1)} = \frac{2}{n}$. Due to space constraints, we only compare the resilience of our solution to the Pairwise Key schemes. Providing a perfect resilience, none of the other solutions can do better. We consider then as example the work presented in [8]. We proved that, using our solution, the rate of compromised links due to a node capture is equal to P. Figure 3 shows that this value is negligible for large networks such as the IoT. It is even comparable to the percentage provided by the perfectly resilient scheme. We also showed that the compromise of the whole network requires the capture of all its members. Our solution provides then a good level of resilience.

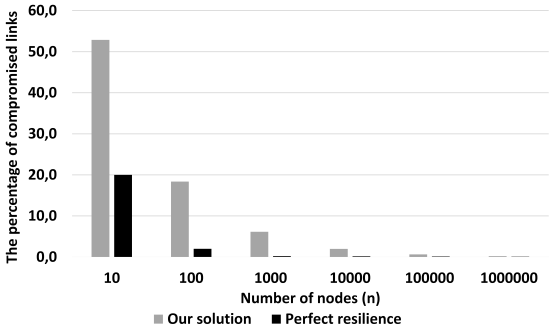


Fig. 3: Variation of the rate of captured links according to n.

V. Performance evaluation

Due to space constraints, we assume that the KM has plentiful of resources and focus on the protocol costs on nodes.

A. Theoretical analysis

We start by analyzing the protocol overheads on nodes.

Proposition 4: Storage and calculation costs on nodes are of the order of $O(\sqrt{n})$, while the communication is $O(1)$.

Proof: Using our solution, a node knows a secret key, $\sqrt{n} - 1$ pairwise node keys, a set key and $\sqrt{n} - 1$ pairwise set keys. It then stores in total $2.\sqrt{n}$ keys. Moreover, regardless of the rekeying operation performed (node joining, node leaving,...etc), a node receives a constant number of messages and calculates the hash of the $2.\sqrt{n}$ keys it knows.

B. Comparison

After showing that our solution provides a good level of resilience, let us prove that it meets the other IoT requirements.

1) Scalability: Although having a perfect resilience, the storage cost of the Pairwise Key schemes, in general, and the work presented in [8], in particular, is of the order of $O(n)$. The deterministic solution (Kronecker) presented in [14] and the probabilistic one (Trade) presented in [13] has a storage proportional to $O(\sqrt{n})$. For the other schemes (e.g. [3]), it is difficult to deduce the storage from the network size as it depends on other parameters. Our solution also has a storage proportional to $O(\sqrt{n})$.

Figure 4 shows that our solution stores fewer keys than the Pairwise Key schemes and can operate on larger networks of compromised nodes such as the IoT. It even provides a level of scalability comparable to the solutions presented in [13, 14].

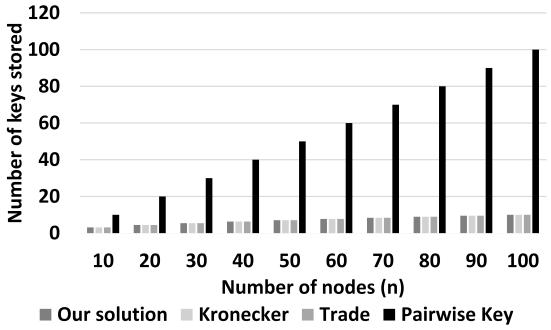


Fig. 4: Variation of nodes' storage overhead according to n.

2) Connectivity: Although being scalable, the probabilistic schemes, mentioned above, suffer from poor connectivity. The probability that two neighboring nodes share a common key does not exceed 0.25 in [13], while in [3] it is approximately lower bounded by 0.632. Using our solution, this probability is always equal to 1. Indeed, each pair of communicators share a pairwise node or subgroup key and can establish a direct secure link. It provides then a good connectivity.

3) Efficiency: Unlike most of the probabilistic schemes [3, 13], our solution has a good connectivity. Thus, it does not require additional calculation and communication costs to establish secure links. It also stores fewer keys on nodes than the Pairwise Key schemes [8] (Figure 4). Moreover, the communication and calculation costs are the same as in [14], while they are of the order of $O((log(n))^2)$ and $O(log(n))$ in [13]. Note that communication is the operation that consumes the most nodes' energy. On top of that, our solution is based on symmetric cryptography. It is therefore efficient.

4) Flexibility: Although deployment knowledge schemes [7, 12, 16] provides good connectivity, they are based on nodes' location. Our solution operates well regardless of the position of nodes and supports their dynamic deployment. We previously showed that nodes can join and leave the network at any time without jeopardizing its security. It is then more flexible and suitable for dynamic networks such as the IoT.

To sum up, considering the IoT requirements (resilience, connectivity, efficiency, scalability and flexibility) as a whole, our solution provides the best compromise between them (Figure 1).

VI. Conclusion

In this paper, we proposed a novel Key Management protocol for Device-to-Device communication in the IoT. Compared to the existing Peer-to-Peer schemes, our solution provides the best compromise between the IoT requirements: resilience, connectivity, efficiency, scalability and flexibility. To achieve this balance, the network members are uniformly distributed into logical sets. A node shares then a distinct pairwise key with each member of its set and a unique pairwise set key with the members of each of the other sets.

We proved that our solution is resilient as the capture of a member compromises a negligible part of a large network. We then showed that it has a good connectivity. It is then efficient as it does not require additional calculation or communication. We also demonstrated that our scheme is scalable as storage on nodes does not significantly increase when the network gets larger. We finally showed that it is flexible.

In future works, we intend to decentralize the protocol in order not to have a single point of failure.

Acknowledgments

This work was carried out and funded by INS2I STFOC project, Heudiasyc UMR CNRS 7253 and the Labex MS2T.

References

[1] F. A. Alaba, M. Othman, I. A. T. Hashem and F. Alotaibi. "Internet of Things security: A survey". In: *Journal of Network and Computer Applications* 88 (2017), pp. 10–28.

[2] E. Baburaj et al. "Polynomial and multivariate mapping-based triple-key approach for secure key distribution in wireless sensor networks". In: *Computers & Electrical Engineering* 59 (2017), pp. 274–290.

[3] W. Bechkit, Y. Challal, A. Bouabdallah and V. Tarokh. "A highly scalable key pre-distribution scheme for wireless sensor networks". In: *IEEE Transactions on Wireless Communications* 12.2 (2013), pp. 948–959.

[4] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. Yung. "Perfectly-secure key distribution for dynamic conferences". In: *Annual international cryptology conference*. Springer. 1992.

[5] H. Chan, A. Perrig and D. Song. "Random key predistribution schemes for sensor networks". In: *Symposium on Security and Privacy*. IEEE. 2003, pp. 197–213.

[6] O. Cheikhrouhou. "Secure group communication in wireless sensor networks: a survey". In: *Journal of Network and Computer Applications* 61 (2016), pp. 115–132.

[7] J. Choi, J. Bang, L. Kim, M. Ahn and T. Kwon. "Location-based key management strong against insider threats in wireless sensor networks". In: *IEEE Systems Journal* 11.2 (2017), pp. 494–502.

[8] T. Choi, H. B. Acharya and M. G. Gouda. "The best keying protocol for sensor networks". In: *Pervasive and Mobile Computing* 9.4 (2013), pp. 564–571.

[9] W. Du, J. Deng, Y.S. Han, P.K. Varshney, J. Katz and A. Khalili. "A pairwise key predistribution scheme for wireless sensor networks". In: *ACM Transactions on Information and System Security (TISSEC)* 8.2 (2005), pp. 228–258.

[10] L. Eschenauer and V. D. Gligor. "A key-management scheme for distributed sensor networks". In: *Proceedings of the 9th ACM conference on Computer and communications security*. ACM. 2002, pp. 41–47.

[11] X. He, M. Niedermeier and H. De Meer. "Dynamic key management in wireless sensor networks: A survey". In: *Journal of Network and Computer Applications* 36.2 (2013), pp. 611–622.

[12] D. Liu and P. Ning. "Improving key predistribution with deployment knowledge in static sensor networks". In: *ACM Transactions on Sensor Networks (TOSN)* 1.2 (2005), pp. 204–239.

[13] S. Ruj, A. Nayak and I. Stojmenovic. "Pairwise and triple key distribution in wireless sensor networks with applications". In: *IEEE Transactions on Computers* 62.11 (2013), pp. 2224–2237.

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

[15] M. S. Yousefpoor and H. Barati. "Dynamic key management algorithms in wireless sensor networks: A survey". In: *Computer Communications* (2018).

[16] Z. Yu and Y. Guan. "A robust group-based key management scheme for wireless sensor networks". In: *IEEE Wireless Communications and Networking Conference, 2005*. Vol. 4. IEEE. 2005, pp. 1915–1920.

[17] 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.

[18] J. Zhang, H. Li and J. Li. "Key establishment scheme for wireless sensor networks based on polynomial and random key predistribution scheme". In: *Ad Hoc Networks* 71 (2018), pp. 68–77.