A decentralized blockchain-based key management protocol for heterogeneous and dynamic IoT devices

Mohamed Ali Kandi, Djamel Eddine Kouicem, Messaoud Samir Doudou, Hicham Lakhlef, Abdelmadjid Bouabdallah, Yacine Challal

A Decentralized Blockchain-based Key Management Protocol for Heterogeneous and Dynamic IoT Devices

Mohamed Ali Kandi, Djamel Eddine Kouicem, Messaoud Doudou, Hicham Lakhlef, Abdelmadjid Bouabdallah and Yacine Challal

Abstract—Secure communication is one of the main challenges that are slowing down the development of the Internet of Things (IoT). Key Management (KM) is particularly a difficult security issue, mainly because of the lack of resources of the IoT devices. Most of the existing solutions do not consider the heterogeneous and dynamic nature of the IoT. They do not regard the difference in capability of its components and impose equal loads on them. Moreover, they store keys in device memories before deployment, which makes adding devices difficult afterwards. We propose a novel decentralized blockchain-based KM protocol for the IoT. Our solution balances the loads between nodes according to their capabilities. We prove that this makes it efficient and scalable. Furthermore, our solution securely rekey the network upon a change. To decentralize the KM, we use the blockchain technology and smart contracts. We show that the system continues to operate when an entity fails and that the compromise of an entity does not jeopardize the whole network. We also prove that our solution fulfills the IoT requirements in terms of security and performance. Finally, we propose an implementation on IoT platforms to validate our theoretical analysis and simulation results.

Index Terms—Internet of Things, Security, Key Management, Blockchain, Smart contract.

1 Introduction

The Internet of Things (IoT) consists of extending connectivity beyond standard devices (computers, tablets, smartphones... etc.) to all everyday objects. This gives rise to a network made up of a large number of heterogeneous devices, which are able to automatically communicate and collaborate with each other. The IoT makes possible the development of a large number of applications that have the potential to improve our lives. Smart homes, for example, involve using smart devices to ensure comfort, convenience and energy efficiency to the homeowners. Autonomous vehicles are able to automatically exchange data to maintain traffic flow and avoid crashes.

Most of the IoT devices are nevertheless limited by their small physical size and suffer from a lack of resources in terms of storage, computation, communication and energy [31]. For these reasons, many challenges are slowing down the development of this emerging technology. Securing communication is considered as one of the most difficult and the most important [19]. Cryptography is a powerful tool used to provide security services. It is based on secret parameters, called keys, that are usually known only to authorized entities. The proper management of keys is essential to guarantee the effectiveness of cryptography.

- *MA. Kandi is with IRIT, Université Paul Sabatier, Toulouse, France. Email: mohamed-ali.kandi@irit.fr*
- *DE. Kouicem, M. Doudou, H. Lakhlef, A. Bouabdallah are with Sorbonne Université, Université de Technologie de Compiègne, HEUDIASYC UMR 7253 CS 60319 60203 Compiègne Cedex France. Email: {djamel-eddine.kouicem,messaoud-doudou,hicham.lakhlef, madjid.bouabdallah}@hds.utc.fr*
- *Y. Challal is with Laboratoire de Méthodes de Conception de Systèmes, École nationale Supérieure d'Informatique, Algiers, Algeria. Email: y_challal@esi.dz*

A Key Management (KM) system includes all the operations involving the handling of cryptographic keys: generation, storage, distribution and replacement [63]. Most of the existing KM solutions are proposed for traditional Internet or Wireless Sensor Networks (WSNs). They rarely consider heterogeneous and dynamic networks such as the IoT. The solutions used in traditional Internet are usually based on asymmetric cryptography. This implies intensive computing, which makes them impractical on the IoT constrained devices [69]. The solutions proposed for WSNs are indeed lighter [57, 72, 74] as they generally use symmetric cryptography. However, they consider that the network is homogeneous, unlike IoT, and do not balance the loads between the nodes. They do not take into account the difference in capability of devices and impose the same overheads on all of them. Thus, while a negligible part of the resources of powerful devices is used by the protocol, those of the constrained ones maybe barely or even not enough. This exhausts the resources of the constrained devices, which can significantly degrade the network performance and shorten its lifetime. It may also happen that some constrained nodes cannot support the overheads at all, while others can handle much more.

Furthermore, most of the existing solutions store the keys in the device memories before deployment. This key pre-distribution is more suitable for static networks, whose members do not change frequently. It is however difficult to add new members after deployment in the case of dynamic networks such as the IoT. On the other hand, the other solutions that dynamically update the keys upon a change in the network rely on a centralized entity [56]. This entity becomes a single point of failure and the main target of attacks. If it fails, the entire system stops operating, and when it is attacked, the whole network is compromised.

To address these issues, we propose a novel decentralized blockchain-based KM protocol for heterogeneous and dynamic networks. Our solution considers the heterogeneity of the IoT devices and balances the loads between them according to their capabilities. By using a bit more of the resources of powerful devices, our protocol becomes lighter for the constrained ones. This improves the network performance and increase its lifetime. Our solution is therefore both efficient and highly scalable in the IoT heterogeneous networks. It also dynamically and securely updates the keys upon a network change. To overcome the disadvantages of centralized approaches, we propose a decentralized architecture based on blockchain. This technology is used as a trustworthy and secure ledger, managed by multiple participants, to ensure the traceability of the update of keys. Thus, the failure of a participant will not prevent the system from working and its capture will not jeopardize the security of the whole network. The main contributions of our work can be summarized in the following major points:

- We present a state of the art of the existing KM schemes and classify them according to several parameters.
- We introduce a novel decentralized blockchain-based KM protocol for heterogeneous and dynamic networks:
  - We propose algorithms that balance the loads between the heterogeneous devices according to their capabilities. We also opted for hybrid encryption, which consists of combining symmetric (for constrained devices) and asymmetric (for powerful devices) encryption to take advantages of each and overcome its disadvantages.
  - We propose a decentralization based on blockchain and smart contracts to automatically and securely distribute new keys to the members upon a network change. We also design a lightweight consensus algorithm that takes into account the capability of the blockchain participants for block validation.
- We analyze and compare our solution with existing schemes. We prove that our solution avoids the single point of failure problem and fulfills the IoT requirements in terms of security and performance (Table 1).
- We propose an implementation on real IoT platforms to validate our theoretical and simulation results.

The remainder of this paper is organized as follows. In section 2, we discuss the related works. In section 3, we recall the basic concepts of blockchain technology. In section 4, we detail our solution. In section 5, we present the security analysis. In section 6, we evaluate the performance of our solution. In section 7, we conclude our work.

2 Related Works

The main role of a KM system is to establish secure links between nodes by providing them with secret cryptographic keys. These keys are used, along with cryptographic algorithms, to secure communication. Although different KM protocols have been proposed, each of them presents its own limitations. None of them meets all the IoT requirements in terms of security and performance. These requirements are summarized in Table 1. They depend on the mode of communication of the IoT devices: device-to-device and group communication.In device-to-device communication (such as Vehicle-to-Vehicle communication), a device addresses a specific other one in a peer-to-peer manner. In group communication (such as Vehicle-to-Everything communication), a device communicates with multiple or all the nodes at the same time. These devices usually participate in the same service and thereby have a common interest.

2.1 Classification criteria

To properly characterize the existing KM protocols and to clearly identify the remaining challenges to be overcome, we classify them according to different criteria: key cryptography, key type, distribution method and load balancing. We discuss each category and explain its weaknesses.

2.1.1 Key cryptography

The KM protocols can be classified into symmetric and asymmetric schemes. Symmetric approaches [5, 7, 10, 11, 15, 16, 21, 25–28, 33, 39, 44, 51, 56, 57, 60–63, 72, 74] involve the use of the same key for ciphering and deciphering the exchanged messages. They usually require affordable computing capability and reasonable computing time. However, their effectiveness depends on the secrecy of these shared keys. The management and distribution of these keys generally require an amount of storage and communication growing with the size of the network. Symmetric approaches are efficient, but not scalable. Asymmetric approaches [1, 3, 9, 35, 38, 45, 46, 50, 52, 65, 66] use two different keys: a public key which may be disseminated widely and a private key which is known only by the owner. One is always calculated from the other so that if the first is used for encryption, the second is used for decryption. Using these algorithms, no secret key exchange is required and a device only needs to store its own keys. However, their effectiveness depends on the difficulty of guessing the private key from the public one [58]. These algorithms are then based on computing power. Asymmetric approaches are scalable, but not efficient.

Communication mode	Requirement	Description
Device-to-Device	Resilience	Capturing devices must have a minimal impact on the network security
	Connectivity	Probability of sharing keys between neighboring devices must be maximum
	Mobility	Moving devices must share keys with their new neighbours
	Flexibility	Devices must be able to join or leave the network at any time
Group	Backward secrecy	New devices must not have access to the old keys
	Forward secrecy	Old members must no have access to the future keys
	Collusion resistance	Unauthorized devices must not have access to the keys if they cooperate
Both modes	Efficiency	Resource usage of devices must be minimal
	Scalability	Increasing the network size must not degrade performance

TABLE 1: Key Management requirements for the IoT.

2.1.2 Type of Keys

The KM protocols can be classified into network and pairwise key schemes. Network key approaches entail using the same key by all members. They are scalable as they require little storage. However, they are not suitable for device-to-device communication as they lack resilience. Indeed, each device can decrypt the communication of the other ones. The best known schemes are: Logical Key Hierarchy (LKH) and Exclusion Basic System (EBS). LKH schemes [68] consist of using a tree structure to reduce the communication cost during the process of rekeying. EBS schemes [15] are based on combinatorial optimization to choose a compromise between the number of keys stored on devices and that of messages exchanged during the rekeying process. Pairwise key approaches consist of using several keys instead of one. They are resilient as capturing a member does not jeopardize the others. However, they are not well suitable for group communication as the same message must be encrypted and sent several times. Pairwise key approaches can be deterministic or probabilistic. Deterministic schemes [5, 11, 57] establish a direct secure link between each pair of devices. They guarantee a total connectivity coverage, but are not scalable. Indeed, each device stores as many keys as there are members in the network. Probabilistic schemes [7, 10, 70, 72, 74] require less storage, but do not guarantee a total connectivity coverage. Intermediate devices are necessary to establish secure links, which requires additional computation and communication [72].

2.1.3 Load balancing

The KM solutions can be classified into homogeneous and heterogeneous schemes. Most of the existing solutions are homogeneous as they impose the same costs on all the nodes. While a negligible amount of resources is sufficient for some, others will not have enough. This exhausts the resources of the constrained devices, which can significantly degrade network performance and shorten its lifetime. Homogeneous schemes lack efficiency and scalability in heterogeneous networks. Heterogeneous schemes balance the costs between the nodes according to their capabilities. By using a bit more of the resources of powerful devices, the constrained ones are more likely to support the costs. Some protocols [1, 2, 29, 46] divide the network into two classes only (powerful and constrained devices), while others [26, 27] automatically adapt to the network state.

2.1.4 Distribution method

The KM protocols can be classified into pre-distribution and post-distribution schemes. Pre-distribution approaches [5, 7, 10, 11, 57, 72, 74] involve the storage of keys in the device memories before deployment. These methods are not suitable for dynamic networks as they lack flexibility. It is difficult to add new members to the network afterwards or revoke those that get compromised. Post-distribution approaches [15, 16, 21, 24–28, 33, 56, 60, 61, 68] rely on a trusted authority to dynamically provide keys to the nodes and to update them upon a network change. This method poses a risk of unavailability and has a low level of resilience. The centralized entity may indeed become a single point of failure and the main target of attacks.

2.2 Proposed classification

To identify the challenges that related works still face in the IoT, we propose a classification of these protocols. Thus, we use the following notation to refer to a class of Key Management solutions: ${}^{typ}_{cry}\mathcal{KM}^{loa}_{dis}$. The abbreviation "$cry$" refers to the key cryptography. It can take two values "sym" for symmetric protocols and "asy" for the asymmetric ones. The abbreviation "typ" indicates whether pairwise keys, "pai", or a network wide key, "net", is used. The abbreviation "dis" can be replaced by "pre" for the protocols that are based on pre-distribution and by "pos" for post-distribution schemes. The abbreviation "loa" specifies whether the load balancing adopted by the scheme is homogeneous, "hom", or heterogeneous, "het". Finally, the notation "hyb" is used, to replace any of the above-mentioned notations, when a scheme is based on a hybridization of two categories.

In Table 2, we introduce a classification of the existing solutions based on the criteria and the notations presented above. We discuss each of the existing classes and present its weaknesses. We therefore see that none of them meet all the IoT requirements. For this reason, we propose a new solution belonging to the class ${}^{hyb}_{hyb}\mathcal{KM}^{het}_{pos}$ and we show that it is more suitable for the IoT. To overcome the drawbacks of a centralized approach, we also propose a decentralization of our protocol based on the blockchain technology and smart contracts.

3 Background of Blockchain

A blockchain is a decentralized and secure storage technology. Its name derives from the fact that it is composed of a chain of blocks, each storing a set of transactions (which is the storage unit) and the cryptographic hash of the previous block. Every participant within a blockchain network takes part in the management of data. The whole system is therefore controlled by the members of a peer-to-peer network. Since these collaborative parties do not necessarily trust each other, the blockchain offers mechanisms allowing them to reach common consensus [67]. Consensus algorithms guarantee that the data cannot be altered without the agreement of most of the participants. Examples of well know consensus algorithms are Proof-of-Work (PoW) [41], Proof-of-Stake (PoS) [59] and Practical Byzantine Fault Tolerance Algorithm (PBFT) [8]).

The blockchain architectures can be classified into three categories [73]: public, consortium and private. In a public architecture (e.g. Bitcoin and Ethereum), the data is accessible and can be managed by anyone who wants to join. A large number of participants makes data more secure, but the operation of validation usually requires a lot of resources. In a private architecture (e.g. Ripple and Tendermint), the data is accessible and can be managed only by authorized users from a specific organization. With a limited number of participants, it is less difficult to alter the data, but the management is more efficient. In a consortium architecture (e.g. Quorum and Hyperledger), the data is accessible and can be managed by authorized users from several organizations. A blockchain has four main features [37, 64]: decentralization, immutability, traceability and security.

Decentralization consists of distributing computation and storage over multiple entities. This solves the single point of failure problem and makes compromising the system difficult. Immutability means that the stored data are permanent and unalterable. This brings more trust between parties and more data integrity. Traceability allows to track the origin of each transaction. This helps to prevent from tampering with the records. Finally, the blockchain uses cryptography to secure data.

Smart contracts are autonomous and irrevocable computer programs stored in the blockchain [12]. They can be automatically run by its participants to execute the settlement of a contract between organizations, people or objects. Although the idea was introduced in 1994, by Nick Szabo, it has not been put into practice until the appearance of the blockchain. It is precisely this technology that has eliminated the need for a trusted third-party.

KM and Blockchain: The term blockchain first appeared in Nakamoto's Bitcoin paper describing a new decentralized cryptocurrency [41]. It started then to be used in various applications. Recently, researchers began to take interest in using it to decentralize the KM. The authors of [32, 33] proposed a blockchain-based KM system to secure group communication in intelligent transportation systems. In [37], a blockchain was used to decentralize the KM for Hierarchical Access Control in the IoT. The authors of [4, 34] proposed blockchain-based authentication schemes for providing secure communication in VANETs. However, most of the existing works do not consider the device-to-device communication and use the proof-of-work [41] consensus algorithm. Our solution secures both device-to-device and group communication. It is also based on a version of proof-of-stake [59] that takes into account the capability of the blockchain participants. Proof-of-stake is known to be far less energy-intensive than proof-of-work [49].

Category	Reference	Discussion
${}^{pai}_{sym}\mathcal{KM}^{hom}_{pre}$	[7, 11, 57, 70, 72, 74]	[+] This category is efficient for secure device-to-device communication in static networks, whose members do not change frequently. [−] This category neither considers group communication nor the dynamic and heterogeneous nature of the IoT.
${}^{pai}_{sym}\mathcal{KM}^{hom}_{post}$	[5]	[+] These categories are efficient for secure device-to-device communication in dynamic networks. [−] These categories do not consider group communication and lack scalabilty in heterogeneous networks containing limited-resource devices.
${}^{pai}_{sym}\mathcal{KM}^{hom}_{hyb}$	[10, 51]	
${}^{pai}_{sym}\mathcal{KM}^{het}_{pre}$	[36]	[+] This category is efficient and scalable for secure device-to-device communication in heterogeneous networks. [−] This category does not consider neither group communication nor the dynamic nature of the IoT.
${}^{pai}_{sym}\mathcal{KM}^{het}_{hyb}$	[13]	[+] This category is efficient and scalable for secure device-to-device communication in heterogeneous and dynamic networks. [−] This category does not consider group communication.
${}^{net}_{sym}\mathcal{KM}^{hom}_{hyb}$	[39, 61, 75]	[+] These categories are efficient for secure group communication in dynamic networks. [−] These categories do not consider the device-to-device communication and lack scalabilty in heterogeneous networks containing limited-resource devices.
${}^{net}_{sym}\mathcal{KM}^{hom}_{post}$	[15, 16, 56, 60, 62, 68]	
${}^{net}_{sym}\mathcal{KM}^{het}_{post}$	[21, 33]	[+] This category is efficient and scalable for secure group communication in heterogeneous and dynamic networks. [−] This category does not consider the device-to-device communication.
${}^{hyb}_{sym}\mathcal{KM}^{hom}_{pre}$	[22]	[+] This category is efficient for secure device-to-device and group communication in static networks. [−] This category does not consider the heterogeneous and dynamic nature of the IoT.
${}^{hyb}_{sym}\mathcal{KM}^{hom}_{hyb}$	[44, 74]	[+] This category is efficient for secure device-to-device, and group communication in dynamic networks. [−] This category lacks scalabilty in heterogeneous networks containing limited-resource devices.
${}^{pai}_{asy}\mathcal{KM}^{hom}_{post}$	[9, 35, 45, 50, 52, 65, 66]	[+] This category is scalable for secure device-to-device communication in dynamic networks. [−] This category does not consider neither group communication nor the heterogeneous nature of the IoT. Also, being based on asymmetric encryption, it is not suitable for the IoT constrained devices.
${}^{pai}_{asy}\mathcal{KM}^{het}_{hyb}$	[46]	[+] This category is scalable for secure device-to-device communication in dynamic and heterogeneous networks. [−] This category does not consider group communication. Also, being based on asymmetric encryption, it is not suitable for the IoT constrained devices.
${}^{pai}_{hyb}\mathcal{KM}^{hom}_{pre}$	[55]	[+] This category is efficient for secure device-to-device communication in static networks. [−] This category neither considers group communication nor the dynamic and heterogeneous nature of the IoT.
${}^{pai}_{hyb}\mathcal{KM}^{het}_{post}$	[38]	[+] This category is efficient for secure device-to-device communication in heterogeneous and dynamic networks. [−] This category does not consider group communication.

TABLE 2: Classification of existing solutions.

4 OUR SOLUTION

Our solution takes into account the heterogeneity of the IoT devices and balances the loads between them according to their capabilities. It also automatically and securely provides new keys to the devices upon a network change. To securely decentralize the KM, our solution is based on the blockchain technology and smart contracts. Thus, our solution is organized into two layers (Figure 1). The node layer organizes the nodes into logical sets and provides them with cryptographic keys. The blockchain layer manages the blockchain and its participants to overcome the disadvantages of centralized approaches. Before we detail these two layers, we start by classifying our solution according to the criteria presented in the related works section.

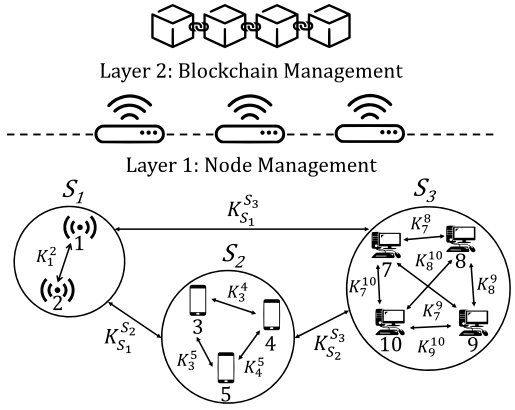


Fig. 1: Architecture of our solution.

Our solution automatically provides keys to the nodes and updates them upon a network change, meaning that is a post-distribution approach. It is heterogeneous as it balances the loads between the devices according to their capabilities. Regarding the type of keys used, our solution combines pairwise and network key schemes to secure the two communication modes of the IoT: device-to-device and group communication. Finally, our solution is a hybridization of symmetric and asymmetric encryption to take advantages of each and overcome its disadvantages. Symmetric encryption is mainly used in layer 1, while asymmetric encryption is only used in layer 2. The main notations, that are used in multiple sections of the paper, are summarized in Table 3.

Notation	Definition
u, v	Examples of nodes
c_u	The capability of u in number of keys
n	The number of nodes in the network
S, T	Examples of node sets
m_S, m_T	The size of the sets S and T
mc_S, mc_T	The minimum capability of the sets S and T
p	The number of sets in the network
BP	A Blockchain Participant
r	The number of BPs in the network
cp	The consensus period
ct	Maximum temporary transactions
RM	A Rekeying Message
KDF	A Key Derivation Function
RNG	A Random Number Generator

TABLE 3: Summary of notations.

4.1 Layer 1: Node Management

The first layer of our solution manages the nodes. It organizes them into logical sets and provides them with cryptographic keys. These keys can be classified into two categories: Data Encryption Keys ($DEKs$) and Key Encryption Keys ($KEKs$). The $DEKs$ are used by nodes to encrypt the data exchanged between them. They include the network key (used to secure group communication) and the pairwise keys (used to secure device-to-device communication). The $KEKs$ are used to secure the communication between the KM and the nodes in order to protect the $DEKs$. The aim is to ensure the backward and forward secrecy (for group communication) and to secure the distribution of keys (for device-to-device communication).

Let us consider a node u that belongs to a set S. The keys it holds are summarized in Table 4. We denote by p the number of sets in the network and by m_S the number of nodes in S. In this case, u manages $m_S - 1$ pairwise node keys, $p - 1$ pairwise set keys, one network key, one node key and one set key. The number of keys u has to handle is therefore proportional to the sum $p + m_S$. On this basis, we propose set management algorithms to balance the loads between the heterogeneous devices according to their capabilities. Unlike homogeneous protocols, where the sets are uniformly organized, our solution determines the size of a set S (m_S) based on the capability (mc_S) of its weakest member. Thus, each set must verify the following inequality:

$$\forall S, mc_S \geq p + m_S \quad (1)$$

Key Type	Notation	Description
Data Encryption Key (DEK)	$K_{u,v}$	This is a pairwise node key used by u to secure the device-to-device communication with v ($v \in S$). A node has as many of these keys as there are members in its set
	$K^{S,T}$	This is a pairwise set key used by u to secure the device-to-device communication with the members of T ($T \neq S$). A node has as many of these keys as there are sets in the network
	K_G	This is a network key used by u to secure the group communication. It is known by all the nodes
Key Encryption Key (KEK)	K_u	This is a node key used by u to secure the communication with the KM. It is known only by u and the KM
	K^S	This is a set key used to secure the communication with the KM. It replaces the node key when the same message is sent to all the set members (for more efficiency). It is known only by the members of S
	K_R	This is a refresh key used for key update. It is not stored in the node memory

TABLE 4: Classification of cryptographic keys.

4.1.1 Set Management

The set management consists of organizing nodes into logical sets while minimizing the number of keys they store. The aim is to improve the protocol scalability without significant loss of resilience, efficiency or network connectivity. To satisfy the inequality 1, a minimum capability mc_S is attributed to each set S when created. We then assign at most $mc_S - p$ nodes to the set.

A node u, that can handle c_u keys, is assigned to S only if mc_S is the nearest value less than c_u. In this case, u will manage $p + m_S$ keys, in the worst case. Since $c_u \geq mc_S$ and $mc_S \geq p + m_S$ then $c_u \geq p + m_S$. In other words, u will always be able to support the costs. Moreover, thanks to this assignment strategy, the loads are well balanced between the nodes according to their capabilities. Considering the fact that a set cannot be empty, any node should be able to store at least p keys. For this reason, the value of p must be minimized.

Since the value of p or m_S increases after the assignment, it may happen that some nodes will not be able to handle their keys anymore. In this case, their sets are split into two sets with the same minimum capability. Their size being divided in two, the members of these sets will again be able to handle the keys. Note that a set cannot be split if it contains only one node. In this case, the set is removed and its member is revoked.

Regarding the choice of the set minimum capabilities, the difficulty lies in the fact that sets are created and removed as and when required and that the abilities of nodes are not known a priori. We tried different increasing sequences and found that the loads are well balanced and p is minimized if the sequence grows exponentially. We then selected powers of two as an example of sequence. The network may be partitioned so that a minimum capability is the double of the preceding one.

The set management is based on two algorithms: Assignment and Reorder Algorithms. First, the Assignment Algorithm is run when nodes join the network and assigns them to the right sets. It creates new ones when it is necessary and may split others so that the inequality 1 remains always satisfied. On the other hand, the Reorder Algorithm is executed after a node leaves the network to reduce the number of sets. It then removes those that become empty and merges others to the possible extent. Figure 2 shows an example of a network partitioned using powers of two. The inequality 1 is satisfied for all the sets and the value of p is minimal.

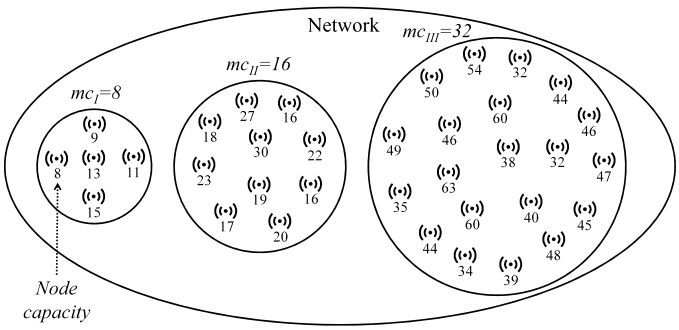


Fig. 2: Example of a network partitioned into three sets.

4.1.1.1. *Assignment Algorithm*

The Assignment Algorithm (Algorithm 1) is run when a node u joins the network. The algorithm takes as input c_u, the number of keys u can handle, and assigns it to a set S according to the input value. To achieve this, the algorithm manipulates a list of sets, ls, of size p. Each of its items contains the ID of a set, its minimum capability, its size and the IDs of its members.

Algorithm 1: Assignment Algorithm

Input : c_u = capability of the node u
Round down c_u to the nearest power of two mc_u;
Find in ls a set S so that $mc_S = mc_u$;
if *no set is found* **then**
  Create a new one S;
end
Assign u to S;
Update ls;
while $\exists$ T *for which* $mc_t < p + m_t$ **do**
  Split T;
end

Set creation:

Creating a new set S consists of assigning it a unique ID, sid_S, a key, K^S, and a pairwise set key for each set T of the network. Each of these pairwise set keys, $K^{S,T}$, is encrypted using the key of the set associated to it and sent to its members (message CM).

$$CM : KM \rightarrow T :< \left\{sid_S, K^{S,T}\right\}_{K^T} >$$

Set splitting:

Splitting S consists first of creating a new set T ($mc_T = mc_S$). The $\frac{m_S}{2}$ last nodes that have joined S are then moved to T. We denote by S^+ the set S after being splitted and by f the first node of S to join T, i.e. $\forall u \in S^+, nid_u < nid_f$ and $\forall v \in T, nid_v \geq nid_f$.

The algorithm determines first sid_T. Next, to ensure the forward secrecy, it randomly generates two refresh keys, K_{R_1} and K_{R_2}. Then, using the KDF, it computes K^{S^+} and K^T (Formulas 2 and 3). After that, all the pairwise keys associated to two nodes which no longer belong to the same set are removed. Also, for each set U (including S), a pairwise set key $K^{T,U}$ is created.

$$K^{S^+} = KDF(K^S||K_{R_1}) \quad (2)$$

$$K^T = KDF(K^S||K_{R_2}) \quad (3)$$

Furthermore, the algorithm sends the message $SM1$ to each node $u \in S$ ($nid_u < nid_f$). The message is encrypted by means of the node secret key and contains K_{R_1}. It also sends the message $SM2$ to each node $v \in T$ ($nid_v \geq nid_f$) encrypted using the node secret key. $SM2$ contains K_{R_2}. Finally, the message $SM3$ is sent to each set U ($U \neq S$ and $U \neq T$). It is encrypted by means of the set key and contains $K^{T,U}$.

$$SM1 : KM \rightarrow u :< \{uid_f, K_{R_1})\}\, K_u >$$

$$SM2 : KM \rightarrow v :< \{uid_f, K_{R_2}\}\, K_v >$$

$$SM3 : KM \rightarrow U :< \left\{K^{T,U}\right\}\, K^U >$$

4.1.1.2. Reorder Algorithm

The Reorder Algorithm (Algorithm 2) is run, after a node leaves the network, to reduce the number of sets. The algorithm takes as input a percentage of merging, pcm, and removes or merges sets if it is possible. Note that the more the value of pcm increases, the more the threshold, thr, below which a set must be merged increases. Thus, this parameter allows us to choose the best compromise between the merging's cost and the value of p.

Algorithm 2: Reorder Algorithm

```
Input : pcm = percentage of merging
foreach set S that a node has left do
    if m_S = 0 then Remove S ;
    else
        thr ← pcm.(mc_S − p);
        if m_S < thr then
            Find T such as m_T < thr and
              mc_T = mc_S;
            if a set T is found in ls then
                Merge S and T;
            end
        end
    end
end
```

Set removal:

Removing a set S consists of deleting its ID, sid_S, its key, K^S, and all the pairwise set keys associated to it. The message RM, containing the ID of the set, is then sent to each remaining set so that its members can remove the pairwise set key they share with the nodes of S.

$$RM : KM \rightarrow T :< \{sid_S\}_{K^T} >$$

Set merging:

Merging S and T consists of three steps. A new set is first created (following the steps presented above). Next, the members of S and T are moved to the new set. New pairwise keys are then generated for every pair of nodes u, v ($u \in S$ and $v \in T$) and sent to them (Messages $MM1$ and $MM2$). These messages are encrypted by means of the node keys. They contain the new cognate *ID* and the pairwise key associated to it. Finally, the two sets S and T are removed (following the steps presented above).

$$MM1 : KM \rightarrow u :< \{nid_v, K_{u,v}\}_{K_u} >$$
$$MM2 : KM \rightarrow v :< \{nid_u, K_{u,v}\}_{K_v} >$$

4.1.2 Rekeying upon joining

When a node u joins the network, the KM starts by running the Assignment Algorithm to select a set S that matches its capability. It determines then the node ID and uses the RNG to generate a secret code and the keys associated to it (Formulas 4 and 5). To ensure backward secrecy, the KM also randomly generates the K_R and uses it with the KDF to update the keys that u will share with other members of the network (Formulas 6, 7 and 8).

$$K_{u,v} = RNG(), \forall v \in S \quad (4) \qquad K_u = RNG() \quad (5)$$
$$K^{S,T^+} = KDF(K^{S,T}||K_R), \forall T \in N \quad (6)$$
$$K_G^+ = KDF(K_G||K_R) \quad (7) \qquad K^{S^+} = KDF(K^S||K_R) \quad (8)$$

After that the keys are generated and updated, they are distributed by the KM on the appropriate nodes (Figure 3). Thus, it sends to each node v of the set S the unicast message $RM1$ encrypted by means of the node secret key, K_v. The message contains the ID of the joining node (nid_u) and the pairwise node key, $K_{u,v}$, associated to it. The KM also broadcasts for each set T (including S) the message $RM2$ encrypted using K_T, the current set key of T. The message contains the ID of S (sid_s) and K_R. Finally, the KM provides u, using a key agreement method, 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.

$$RM1 : KM \rightarrow v :< \{nid_u, K_{u,v}\}_{K_v} > (\forall v \in S)$$
$$RM2 : KM \rightarrow T :< \{sid_S, K_R\}_{K_T} > (\forall T \in N)$$

When a network node receives the rekeying messages, it decrypts them using its node and set keys. It then stores the new keys and updates those that it will share with the joining node (Formulas 6, 7 and 8).

4.1.3 Rekeying upon leaving

When a node u leaves a set S or is evicted because it gets compromised, the KM runs the Reorder Algorithm if a set removal or merging is possible. Next, the KM removes the node's key and all the pairwise keys associated to it. The same steps as for the node joining are then followed to update the keys known by u (Formulas 6, 7 and 8).

Next, the KM distributes the new keys to the appropriate nodes (Figure 3). Thus, it sends, to each node v of the set S, the unicast message $RM3$ 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 $RM4$ to provide its members with K_R. The message $RM4$ is encrypted using K_T, the current set key of T.

$$RM3 : KM \rightarrow v :< \{nid_u, K_R\}_{K_v} > (\forall v \in S, v \neq u)$$
$$RM4 : KM \rightarrow T :< \{sid_S, K_R\}_{K_T} > (\forall T \in N, T \neq S)$$

When a network node receives a rekeying message, it decrypts it using its node or set key. It then removes the keys associated to the leaving node and updates those that it shared with it (Formulas 6, 7 and 8).

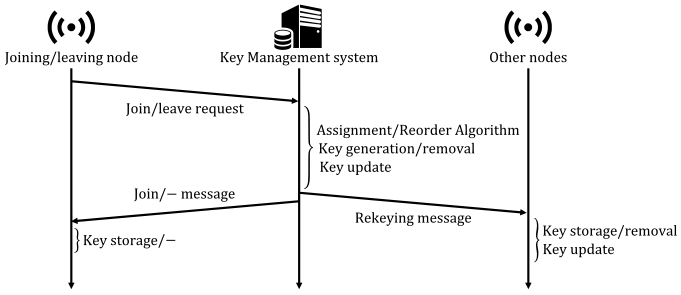


Fig. 3: Rekeying upon a network change.

4.2 Layer 2: Blockchain Management

The main purpose of this layer is to decentralize the KM using the blockchain technology and smart contracts. Note that any type of architecture can be used (since the secret keys are not stored in the blockchain), but a private or a consortium blockchain remains preferable in an application such as the KM. We introduce into the network IoT gateways (or BPs for Blockchain Participants) that generate, validate and store transactions upon a network change.

The BPs act as intermediaries between the nodes and the blockchain (Figure 1). The aim is to not involve any additional cost on nodes, except those imposed by Layer 1. When a node wishes to join the network, it sends a request to a BP. If the transaction corresponding to this request is validated by the other BPs and is correctly added to the blockchain, the node is attached to the gateway that initiates the joining process. It will remain attached to it until the node moves, leaves the network or when the BP fails or gets compromised (see section 4.2.3). Meanwhile, the BP manages (generates, stores and updates) the keys associated to the node. The BP also sends to the node the rekeying messages, so that it can update its keys.In case a new shared key needs to be created (e.g., the group key or a set key), the BPs can directly and securely exchange it.

A blockchain transaction is the storage unit that corresponds to a specific event, which is a rekeying operation in our case. Any BP that executes in order the operations stored in the blockchain should have the same organization in sets. As shown in Figure 4, a transaction contains the following information:

- The rekeying operation (join, leave, evict...);
- The node ID;
- The node capability;
- The ID of the set of the node;
- The encrypted cryptographic hash of the node's secret code (see section 4.2.3);
- The refresh key used to update the keys.

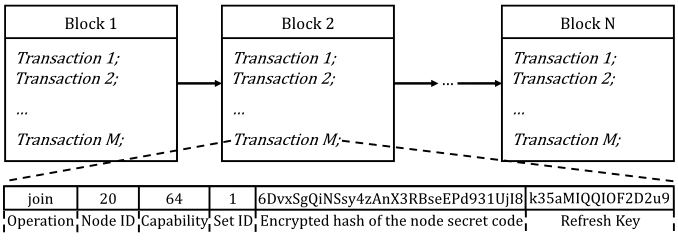


Fig. 4: Example of a blockchain transaction.

4.2.1 Transaction management upon network change

When a BP receives a join/leave request, it first uses layer 1 for set and node management. The algorithms and operations performed (e.g. Assignment and Reorder Algorithms) are transformed into smart contracts. The aim is to ensure that, for a certain join/leave request, any BP should obtain the same result. Before distributing the keys, the layer 1 calls the layer 2 to generate, validate and store a transaction in the blockchain. If the transaction corresponding to the current rekeying operation is correctly stored in the blockchain, the layer 2 informs the layer 1. The BP can then distribute the generated keys on the appropriate nodes after ciphering them using the $KEKs$ (Figure 5).

(1) Transaction generation: When the layer 2 receives the information from the first layer about a rekeying operation, it starts by generating the corresponding transactions. A single transaction is sufficient for most rekeying operations. However, more than one transaction may be generated in some cases. Indeed, in the case of a set merging, we need to generate transactions for node leaving, set removal and set creation. The layer 2 then stores these transactions in its memory pool of temporary transactions and broadcasts them to all the BPs. Note that the communication between the BPs is ciphered using asymmetric encryption.

(2) Transaction verification: When a BP receives a transaction, it uses the smart contracts to verify its correctness. In the case of a node joining, the BP runs again the assignment algorithm to confirm that the node was assigned to the right set. It also checks if the node ID and the hash of the secret code have not already been used for another network node. On the other hand, if a node leaves the network or is evicted, the BP checks if the leaving node is actually a network member. It also verifies if there is a match between the node ID, the set ID and the cryptographic hash of the node's secret code. If the BP judges that the transaction is correct, it adds it to its memory pool.

(3) Transaction validation: After a certain period of time (cp) or when the size of the memory pool reaches a certain threshold (ct), the BPs run a consensus algorithm (see next section). The aim is to achieve a consensus between them on whether the content of the memory pool can be included to the blockchain. If all the BPs agree that a block of transactions can be added to the blockchain and when it is correctly stored, these transactions become valid. The layer 1 is therefore informed so that it can distribute the generated keys on the appropriate nodes.

4.2.2 Consensus Algorithm

We introduce a lightweight consensus algorithm for transaction validation. This is a proof-of-stake version that takes into account the capability of the blockchain participants, i.e. the BP's capability determines its chance to validate the next block. In order not to favor the more powerful BPs, more parameters are included into the selection process: confidence and age. The level of confidence is initialized to a certain value, increased over time and decreased when an incorrect transaction from a BP is detected. The age of a BP represents the time elapsed between the last time it validated a block and a given time. This gives chance to the weak BPs to participate in the validation process when their age is high enough.

Our Consensus Algorithm (Algorithm 3) can be executed either periodically (the period is cp) or when the number of transaction in the memory pool reaches a certain threshold (ct). For each block, a validator (the BP that forges the block) is randomly elected by including a weighting according to the capabilities of BPs and the other abovementioned parameters (the higher the capability, the confidence and the age of a BP, the higher its chance to be elected). To achieve this, all BPs generate random numbers and exchange them with each other. Each of them combines all these random values by adding them, for example, or by applying another common mathematical function. The resulting value, being common to all, can be used by BPs

to perform a weighted random draw and thereby elect the same validator. This validator groups the transactions contained in its memory pool to forge the new block, signs it then broadcasts it. The other unelected BPs wait for the new block and check its content once received. All the BPs, including the validator, store the block in their copy of the blockchain and remove the validated transactions from their memory pool. Finally, they update the validator parameters. They reset its age and modify its level of confidence depending on whether an error is detected or not.

Algorithm 3: Consensus Algorithm

```
Generate a random number and broadcast it;
Wait for the random numbers of the other BPs;
Combine the received numbers into one value;
Use this shared value to elect a validator;
if the BP is the validator then
    Forge the new block;
    Sign the block;
    Broadcast the block;
else
    Wait for the block from the validator;
    Check the block and its transactions;
end
Store the block in the blockchain;
Remove validated transactions from memory pool;
Update the validator parameters;
```

4.2.3 Blockchain interest

The blockchain is used as a trustworthy ledger that ensures the traceability of the updating of keys. Thus, in addition to securely distribute the KM and ensure consistency between the different BPs, the use of blockchain offers functionalities that a centralized solution cannot provide.

System availability: When a BP fails or when it is a target of malicious attacks (such as DoS attacks), the nodes attached to it become orphans. Each of them sends then a rejoin request to an other BP. When a BP receives a rejoin request, it agrees with the sender on new KEKs so they can securely communicate. After that, the node sends the hash of its secret code to be able to get authenticated. The BP consults the blockchain and checks if the hash received corresponds to that of the node. The node in question is the only one able to generate the hash of its secret code. Moreover, the hashes stored in the blockchain are encrypted using a key known only by the authorized BPs. Therefore, the BP can conclude that the node sending the request is legitimate. If it is the case, the node is then attached to this gateway without having to add new transactions to the blockchain (Figure 6). This makes the rejoin operation more efficient. More importantly, the failure of a BP does not prevent the system from working. Therefore, our solution solves the single point of failure problem, especially as no BP knows all the keys.

Node mobility: As when a BP fails, a node can use its secret code to get authenticated with another BP if its actual BP is no longer in range. The node then sends a rejoin request to a BP which is within reach. When the BP receives the rejoin request, it sends to the node new KEKs to secure their communication. The node sends then the hash of its secret code to get authenticated. The BP consults the blockchain and checks if the hash received corresponds to that of the node.If it is the case, the node is attached to this gateway without adding new transactions to the blockchain (Figure 6). This makes the mobility operation more efficient. Our protocol is therefore well suitable for dynamic networks.

Node sleeping: To save energy, a node can sleep if it does not have a work in progress. During sleeping, the node turns off its radio and will not receive the rekeying messages. Note that these messages contain the refresh keys that allow the nodes to update their keys. Thus, the sleeping node will not have the opportunity to update its keys. However, when it wakes up, it will need the new keys to be able to securely communicate with the other nodes. It will then send to its BP a rekey request containing the last refresh key it received. Since all the refresh keys are stored in the blockchain, the BP can retrieve and send to the node the refresh keys it missed. It will then be able to update its keys without having to add new transactions to the blockchain.

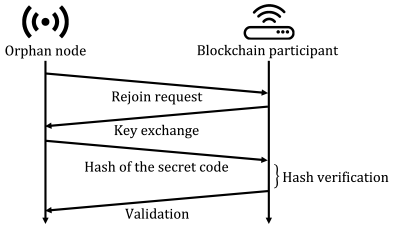


Fig. 6: Rejoin exchange.

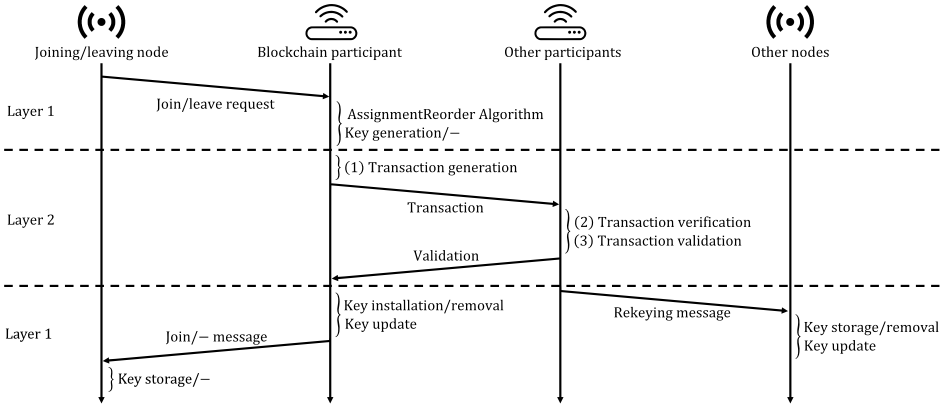


Fig. 5: Decentralized rekeying upon a network change using a blockchain.

5 Security analysis

In this section, we analyze the security of our solution and prove that it secures the two modes of IoT communication.

5.1 Threat model

A malicious device can be inside or outside the network [6] and may jeopardize the security of both modes of communication (Figure 7). An outsider node can store the messages exchanged between the nodes (group communication) and decipher them when it joins the network. An evicted member can also pose a threat to the network, if it is still able to decipher the future communications. If a node or a BP inside the network is captured, it may try to decrypt the device-to-device communication of the other nodes. We assume that the blockchain is tamper proof (protected against P2P attacks such as eclipse or hijacking attacks). An attacker can not alter its content unless it has a capability that exceeds 51% of the overall network capacity, which is practically unlikely [18].

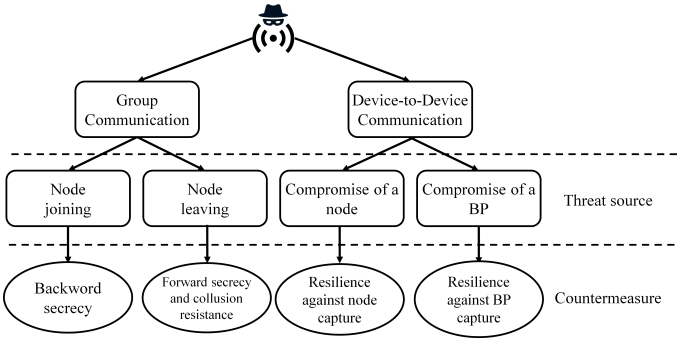


Fig. 7: Threat model and countermeasures

5.2 Backward secrecy

We prove that a joining node cannot access the current or previous shared (network, set and pairwise set) keys.

Proposition 1: Backward secrecy is guaranteed as the joining node never gets knowledge of the old keys.

Proof: Let us consider a node u that joins a set S. The KM starts by updating the keys mentioned above. Before u can actually join the network, the KM rekeys all current members of the network, by means of messages $RM1$ and $RM2$. These messages are encrypted using the node and set keys, respectively. Since none of these keys are known by u, it is excluded from the process of rekeying.

5.3 Forward secrecy and collusion resistance

We prove that leaving nodes cannot access the new shared (network, set and pairwise set) keys or any future incarnation of them, even if they cooperate.

Proposition 2: Our solution guarantees the forward secrecy and resists to collusion after nodes leaving, since they do not have access to the new security material.

Proof: Let us consider a node u that leaves a set S. The KM rekeys the nodes by means of the messages $RM3$ and $RM4$, respectively. The former is encrypted using the node keys and the latter 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. Furthermore, as these keys are independent of each other, u can not collude with other evicted nodes to decipher the rekeying messages.

5.4 Resilience against node capture

Although our solution is heterogeneous, a homogeneous distribution allows us to evaluate resilience against node capture with no significant lack of generality. The n nodes $(n > 1)$ of the network are then uniformly distributed in p sets of m members each, i.e. $p = m = \sqrt{n}$

5.4.1 Theoretical analysis

We study the rate of compromised links after node capture.

Lemma 1: A node can decrypt a number of links D:

$$D = n - 1 + (\sqrt{n} - 1)(n - \sqrt{n}) = (\sqrt{n} - 1)(n + 1) \quad (9)$$

Proof: A node can decrypt the communications linking it to the $n - 1$ other nodes 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:

$$P = \frac{D}{T} = \frac{2(n+1)}{(\sqrt{n}+1)n} \to 0, \text{ as } n \to \infty \quad (10)$$

Proof: From lemma 1 and the fact that the total number of links in a network of n nodes is equal to $T = C_n^2 = \frac{n(n-1)}{2}$, we obtain this percentage.

Proposition 4: The capture of the whole network cannot succeed unless all the nodes are compromised.

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

5.4.2 Comparison

We compare our solution to the deterministic scheme presented in [11]. Providing a perfect resilience, none of the other solutions can do better. Figure 8 shows that, using our solution, the rate of compromised links due to node capture is negligible for large networks like 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.

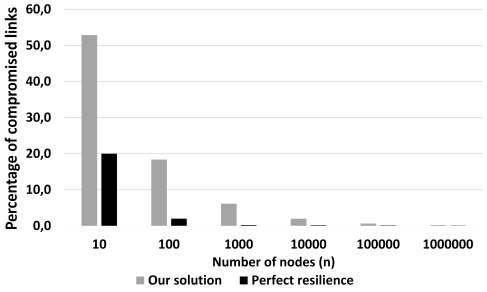


Fig. 8: Resilience against node capture.

5.5 Resilience against *BP* capture

We assume that nodes are uniformly distributed on a number of BPs equal to r (i.e. $\frac{n}{r}$ nodes are attached to each BP). This allows us to evaluate resilience against BP capture without a significant lack of generality.

5.5.1 Theoretical analysis

We study the rate of compromised links after BP capture.

Proposition 5: The percentage of links that a compromised BP can decipher is equal to:

$$P = \frac{D}{T} = \frac{2nr - n - r}{(n-1)r^2} \rightarrow \frac{2r-1}{r^2}, \text{ as } n \rightarrow \infty \qquad (11)$$

Proof: A BP is responsible for the generation of the keys associated to the nodes attached to it. Therefore, if it gets compromised, it will be able to decipher the $\frac{n}{2r}(\frac{n}{r} - 1)$ links between them. It will also be able to decipher the communications between its $\frac{n}{r}$ nodes and the $n - \frac{n}{r}$ other members of the network. It can then decrypt a total number of links equal to $D = \frac{n}{2r}(\frac{n}{r} - 1) + \frac{n}{r}(n - \frac{n}{r})$.

Proposition 6: The capture of the whole network cannot succeed unless all the BPs are compromised.

Proof: As shown in proposition 4, deciphering all the communications requires the knowledge of all the pairwise keys. This is possible only if all the BPs are captured.

5.5.2 Comparison

In previous works [25, 26], we assumed that the KM itself is secure and that only the nodes can be compromised. In this paper, we propose a decentralization based on the blockchain as in practice the central entity can be captured. Thanks to the blockchain features, the KM is securely decentralized so that the compromise of a BP has no effect on the others. Thus, compared to the solution based on a centralized entity [25, 26], which once captured the whole network is compromised, only a part (Proposition 5) is captured using this decentralized version (Figure 9). We showed that the rate of compromised links is inversely proportional to the number of BPs. In other words, the more we increase the number of BPs, the more resilient is our solution. In the next section, we analyze the effect of this parameter on the network performance to help the reader to choose the best compromise between resilience and performance.

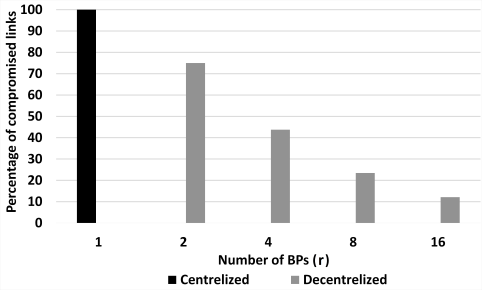


Fig. 9: Resilience against BP capture.

6 Application and Performance evaluation

To give a concrete overview of the performance of our solution, we consider the example of a smart city. This choice is motivated by the fact that a smart city contains a huge number of heterogeneous devices (servers, computers, smartphones, gateways, sensors...etc) spread across the city to provide various services to the benefit of society (healthcare, intelligent transportation system...etc.). These devices can use the two communication modes of the IoT (device-to-device and group communication).

6.1 Theoretical analysis

We study the overheads of our solution on nodes and BPs.

Property 1: The storage cost on the members of a set S is proportional to $p + m_S$. It is of the order of $\frac{n}{r}$ on the BPs.

Proof: A node of a set S stores $m_S - 1$ pairwise node keys, $p - 1$ pairwise set keys, one network key, one node key and one set key. The storage overhead on nodes is therefore proportional to the sum $p + m_S$. If we choose not to store the pairwise node keys (which are used for device-to-device communication between the nodes) in the BP memories, the largest number of keys to store will then be that of the node keys of the members attached to it. The BP will then store a total number of keys proportional to $\frac{n}{r}$.

Property 2: The calculation cost of an operation related to a set S is proportional to $p + m_S$ on nodes and BPs.

Proof: Regardless of the rekeying operation performed (e.g. node joining S or node leaving S), a node can, in the worst case, update all the keys it knows. The calculation cost on nodes is then proportional to the storage, which has been proven to be of the order of $p + m_S$. Furthermore, a BP updates the keys which are or will be known by the node in question. The calculation overhead on the BP is therefore also proportional to the storage cost on nodes. Regarding the Assignment and Reorder Algorithms, they loop through the list of sets in the worst case, which is of size p.

Property 3: The communication cost is $O(1)$ on nodes. The communication overhead of an operation related to a set S is proportional to $p + m_S$ on the BPs.

Proof: Regardless of the rekeying operation performed, a node receives a constant number of messages. The communication cost on nodes is therefore independent of all parameters $(i.e. O(1))$. Also, a BP sends a unicast message to each of the m_S members of S and broadcasts a message for each of the other $p - 1$ sets, in the worst case. A BP sends then a number of messages proportional to $p + m_S$.

6.2 Experiments

For our experiments, we used a laptop and IoT motes. The laptop is an Intel Core i7 with 4GB RAM. The motes (based on Contiki) are of 5 types: 22 Exp5438 [17], 22 MicaZ [40], 12 Openmotes [43], 22 TelosB [54] and 22 Z1 [71]. To join the network, the devices send requests to the BPs. The BPs process the requests and send rekeying messages to nodes. Given the limited number of physical motes available to us, some are Cooja motes. For scaling up even more, a simulator (written in Python) sends requests for the tests in which the number of devices exceeds 100 (Figure 10). The experiments are carried out following these assumptions:

- We use AES-128 for encryption involving nodes and AES-256 for BPs. Indeed, some of the nodes support only AES-128. We use ECC [30] for the key exchange between BPs and for signing the blocks. We plan to propose a new lightweight key exchange method in future works. For now, we store a temporary key in the nodes before deployment to secure the first exchange.
- We rely on storage to assess the capabilities of nodes. Indeed, the communication for nodes is $O(1)$ and the calculation depends on storage. The keys (128 bits long) are stored in the Flash memory of the nodes.

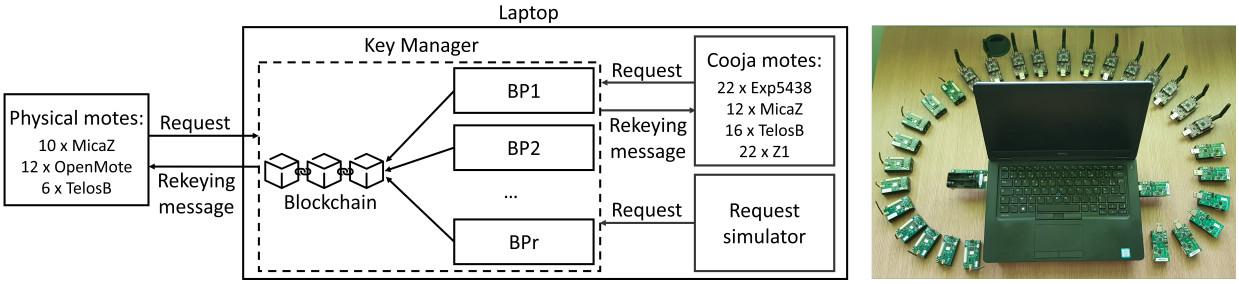


Fig. 10: Experimental platform.

- The simulated requests are uniformly distributed on the BPs and their capabilities follow a uniform distribution between 64 and a maximum value noted C_MAX.

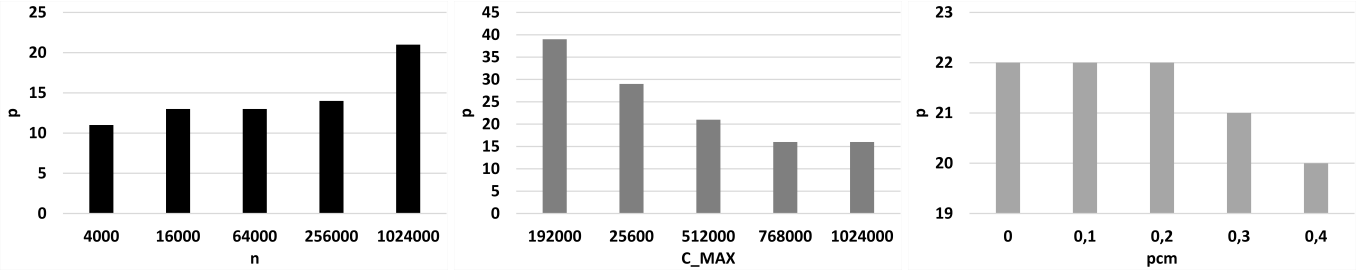
6.2.1 Number of sets

We deduce from the theoretical study that our solution's overheads mainly depend on the number of sets. Thus, we start by analyzing the evolution of p according to the algorithms' inputs. The aim is to present an overview on how to minimize this value. We study the effect of three parameters on the value of p: the number of nodes (n), the nodes maximum capability (C_MAX) and the percentage of merging (pcm). Each time we set two parameters to default values and vary the third one. The default values of the three parameters are 1024000, 0.4 and 256000, respectively. Note that only the request simulator is used in this part of the experiments.

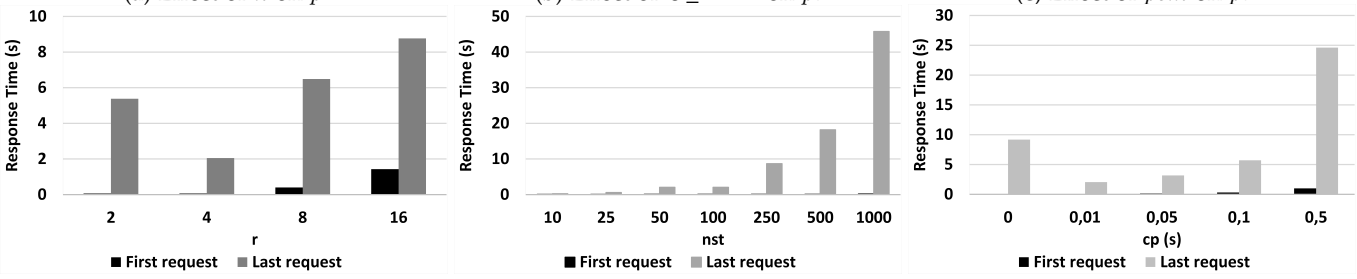
Starting with n, the results (Figure 11a) show that, regardless of the network size the number of sets remains reasonable. Even when n exceeds one million, the value of p does not exceed a few dozen. This makes our solution scalable since the constrained nodes manage a reasonable number of keys. Next, we analyze the effect of C_MAX, the maximum capability of the simulated requests. The results (Figure 11b) show that the more powerful the nodes, the smaller the value of p. This is because powerful devices are able to manage more keys and can be assigned to larger sets. Note that the larger the sets, the more their number decreases. Therefore, the constrained nodes are more likely to support the overheads. Even when the maximum capability is small, the value of p remains reasonable for a network containing over a million nodes. Finally, we study the effect of pcm. The results (Figure 11c) show that the greater the percentage of merging, the smaller the value of p. Therefore, the merging operation actually reduces the number of sets and makes our solution lighter for the constrained devices. Most of the costs imposed by set merging are on the BPs and have no significant influence on the nodes performance.

6.2.2 Overheads on the BPs

After studying the evolution of the number of sets, we analyze the overheads of our solution on the BPs (the blockchain overhead). We assume that, unlike nodes, the BPs have enough storage and focus on their response time. It is the time separating the reception of a join or leave request from the sending of a response to the node. We analyze the effect of three parameters on the response time (of the first and the last of simultaneous requests): the number of BPs (r), the number of simultaneous requests (nst) and the consensus period (cp). Each time we set two parameters to default values and we vary the third. The default values are 8, 100 and $100ms$, respectively. The size



(a) Effect of n on p (b) Effect of C_MAX on p. (c) Effect of pcm on p.



(d) Effect of r on time response. (e) Effect of nst on time response. (f) Effect of cp on time response.

Fig. 11: Experimental results.

of the memory pool (ct) is set to the number of BPs (r). Note that the request simulator as well as the physical and Cooja motes are used in this part of the experiments.

Starting with r, the results (Figure 11d) show that the more the number of BPs increases, the more the processing time of one request rises. This can be explained by the fact that there is more communication between BPs. However, if more than one request are received at the same time, we notice a decrease in the response time (of the last request) before it starts going up again. This is because several BPs can process different requests at the same time. However, after a certain threshold (8 BPs with our means), the time lost due to communications covers the time saved thanks to parallelism. Next, we analyze the effect of nst. The results (Figure 11e) show that the more the system receives simultaneous requests, the more the response time increases, especially for the last node. Note that with our means, more than 1000 requests are processed per minute. Finally, we study the effect of cp (Figure 11f). The best results are obtained when the period to forge new blocks is neither too short nor too long ($10ms$ with our means). If it is too short, there will be a lot of unnecessary message exchanges, while the memory pool is empty. Conversely, when this period is too long, the processing time of a request increases.

6.2.3 Overheads on nodes

Now, we evaluate the performance of our solution on the nodes. We used the 100 Cooja and physical motes in this part of the experiments. They are all within the reach of the BPs (routing is therefore not necessary) and belong to the same network. After all the nodes joined the network, we obtained the set distribution shown in Figure 12. Note that the sets 4 and 5 result from the split of the sets 2 and 1, respectively. Furthermore, we used PowerTrace for the execution time and the energy consumption. It is a Contiki built-in tool that reports the resource utilization of a node (with an accuracy of 94% [14]) and prints the statistics to the console. However, This tool is not supported by MicaZ. Therefore, we only present the storage cost on these motes.

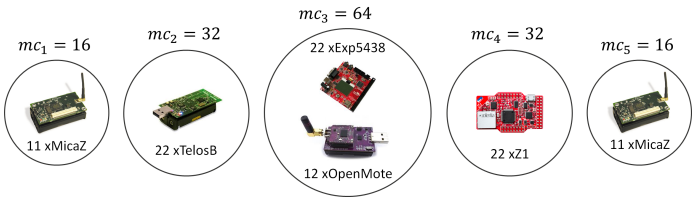


Fig. 12: Network partitioning.

Storage overhead on nodes: The program, the values of the initialized variables and the cryptographic keys are stored in the flash memory, while the data are saved in RAM. The choice of storing the keys in the flash memory was motivated by the fact that some of the motes (especially MicaZ) have a very limited RAM. The number of keys stored in nodes varies depending on weather our heterogeneous seting (proportional to the node capacities) is used or not (i.e. nodes hold n keys). The percentage of memory occupation is presented in Table 5. The results show that, although the network is neither too large nor too heterogeneous, our heterogeneous seting reduces the storage on nodes.

Node	Flash		RAM
	Heterogeneous	None	
Exp5438	9.76%	10.14%	52.75%
MicaZ	19%	19.9%	91.70%
OpenMote	5.46%	5.65%	21.05%
TelosB	63.26%	66.03%	47.15%
Z1	34.7%	35.96%	37.72%

TABLE 5: Storage overhead on nodes.

The occupation of the MicaZ RAM may seem important. However, the basic communication program, which consists of periodically sending and receiving unicast messages (without broadcast and AES encryption), occupies 67% of the RAM. In other words, a large part of the memory is used by other processes. The part of the memory used by our protocol remains reasonable, especially considering the very limited MicaZ RAM. In [47], the network is composed of 15 TelosB motes only and yet 96.3% (against 63.26% using our solution) of their flash memory and 74.92% (against 47.15% using our solution) of their RAM are occupied. Our solution requires then less storage on TelosB.

Execution time on nodes: The evaluation of the computing cost of our solution concerns two operations a node may perform: key installation and key update. Key installation corresponds to the operation by which a new node, which has just join the network, processes the Key Manager's messages and stores the keys assigned to it in its memory. Key update is the operation by which a network node updates its keys upon a network change. For this operation, we consider both cases where a set is split or not. The execution time of any of these operations corresponds to the time separating the reception of a message from a BP and the end of its processing. The execution times on nodes are presented in Table 6. The results were obtained using Powertrace. This tool is not supported on MicaZ. Therefore, we only present the execution time on the other motes.

Node	Key installation	Key update	Key update/split
Exp5438	57.98 ms	69.18 ms	139.34 ms
OpenMote	101.86 ms	103.46 ms	211.46 ms
TelosB	85.41 ms	89.02 ms	177.28 ms
Z1	60.33 ms	76.29 ms	167.23 ms

TABLE 6: Execution time on nodes.

Energy consumption by nodes: For energy consummation, we consider the same operations as in the previous section (key installation and key update with or without set split). The energy consumed by a node includes calculations and communications. We then used Powertrace to evaluate the energy consumption of our solution per second. The results are presented in Table 7.

Node	Key installation	Key update	Key update/split
Exp5438	3.2 mJ	3.95 mJ	8.41 mJ
OpenMote	4.38 mJ	4.52 mJ	10.03 mJ
TelosB	2.32 mJ	4.22 mJ	9.83 mJ
Z1	3.7 mJ	4.38 mJ	9.46 mJ

TABLE 7: Energy consumption by nodes.

In Table 8, we present an estimate of the lifespan of 2xAA batteries using our solution. The lifespan is calculated, for each type of node, according to the average value of the quantity of energy it consumes. The results are obtained assuming that the nodes are constantly receiving a rekeying message per second.

Node	Battery life
Exp5438	101 days
OpenMote	69 days
TelosB	74 days
Z1	76 days

TABLE 8: Battery life.

Although some authors (such as those of [7, 38, 42]) evaluate the energy consummation of their Key Management protocols, they are based on theoretical models (an example can be found in [20]) and cannot be compared to our experimental results. In the absence of similar works presenting the results of an implementation on the same IoT motes we used, our results can be compared to those presented in [53]. The authors used Powertrace to estimate the energy consumption of TelosB motes when sending ipv6 packets (while we used the lightweight Rime Stack). Since communication is the operation that consumes the most energy, these results can be compared to the energy consumption of our solution, even if they do not consider the overheads related to the Key Management.

6.3 Comparison

We compare our solution to the existing protocols proposed for device-to-device (PKS [11], Trade [48], UKP [7], Kronecker [57]) and group communication (MGKMP [28] and GREP [56]). We also compare our consensus algorithm to Tendermint, which is based on PBFT [23].

6.3.1 Efficiency and scalability

We take as example a TelosB node, which can theoretically store up to 1536 keys of 256 bits (ignoring the other node's memory requirements). For the node to support the storage cost of our solution, it must be able to store at least p keys. The percentage of storage must then be greater than $P_o = \frac{p}{1536}$. The authors of GREP, MGKMP, Trade and Kronecker show that the storage of their solutions is proportional to $O(\sqrt{n})$. The memory rate required to store these $\sqrt{n}$ keys is $P_r = \frac{\sqrt{n}}{1536}$. Finally, the nodes storage cost of PKS is of the order of n. The percentage of memory required is $P_n = \frac{n}{1536}$.

We compare the variation of these percentages according to n (Figure 13a). The results show that our solution requires less storage on a TelosB than the other protocols. Indeed, regardless of network size, the value of P_o is smaller than P_r and P_n. More importantly, if the network contains 1536 nodes, the total memory of the TelosB will be used to store all the keys of PKS, while 1% of its storage capability is enough if our solution is used. This is because storage cost is well balanced between the nodes according to their capabilities. Thus, by using a bit more of the resources of powerful devices, our solution becomes much lighter for the constrained ones. It can then operate on much larger heterogeneous networks.

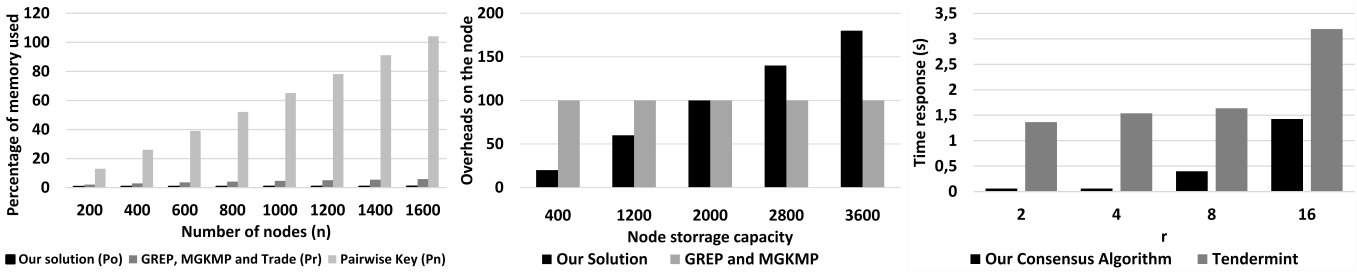
6.3.2 Connectivity, mobility and flexibility

While the probability that two neighboring nodes share a common key does not exceed 0.25 in Trade and is approximately lower bounded by 0.632 in UKP, it is always equal to 1 using our solution. Moving devices will also always share keys with their new neighbours. Indeed, each pair of communicators share a pairwise node or set key and can establish a direct secure link. Our solution provides then good connectivity and mobility. Note that when connectivity and mobility are low, some communicators rely on intermediate nodes to establish a secure link. This reduces the efficiency of the protocol.

Some schemes [10, 70] are based on the deployment knowledge to maximize the network connectivity. The application of this method is nevertheless restrictive if the deployment knowledge is not possible. It is therefore clear that the KM we propose is more flexible as the division into sets is logical and operates well regardless of the position of nodes. Furthermore, regardless of their category, most of the existing schemes for device-to-device communication are based on key pre-distribution. These schemes suffer from poor flexibility as it is hard to add new nodes to the network. Our solution, on the other hand, supports the dynamic deployment of nodes thanks to its rekeying mechanism. Indeed, we previously showed that nodes can join and leave the network at any time without jeopardizing the security of the network. Our solution is therefore more flexible.

6.3.3 Heterogeneity

Unlike most of the existing schemes, our solution balances the loads between the heterogeneous devices of the network according to their capabilities. To illustrate this difference, we consider the protocols MGKMP and GREP. The authors show that their calculation and storage costs are proportional to $O(\sqrt{n})$ for all the nodes, while they are both



(a) Efficiency and scalability.

(b) Heterogeneity.

(c) Consensus algorithm.

Fig. 13: Comparison results.

proportional to the nodes' storage capability, using our solution. We consider then a network of 10000 nodes and analyze the variation of the calculation and storage cost according to the node's capability (number of keys it can handle), for the three protocols. Note that the percentage of storage capability that we choose using our solution is 5% (i.e. only 5% of the real capability of the node is used). The results are plotted in Figures 13b.

We take as example two nodes u_1 and u_2 that can store 200 and 1800 keys, respectively. For both nodes, 5% of their memory is used by our solution, in the worst case. MGKMP and GREP, on the other hand, use 50% of the former and 5% of the latter. As the calculation overhead on node depends on the storage, these protocols quickly exhausts the resources of u_1, while u_2 has much more. More importantly, the nodes having a capability lower than 100 can not even store all the keys, while our solution uses 5% of their memory only. Thus, although the overheads imposed by MGKMP and GREP are lower than that of our solution for powerful devices (capability greater than 1000), they are much greater for the weak ones.

6.3.4 Consensus Algorithm

We mainly choose Tendermint for two reasons. First, it is a powerful blockchain engine based on the PBFT consensus algorithm [23]. Using Tendermint, solving a puzzle is not required to enter the next block. Therefore, compared to some of the most used consensus algorithms (e.g. PoW), PBFT reduces computation and thereby the energy consumption. Second, the application layer of Tendermint can be written in any programming language. This facilitates the integration of our solution (written in Python) to this environment. The other parameters that are not related to the consensus algorithms (such as the key size, the assignment algorithm and the material used) are the same in both cases (using our consensus algorithm and PBFT of Tendermint). The obtained results are plotted in Figure 13c. They show that regardless of the number of BPs, the process of a request is always faster using our solution.

7 Conclusion

In this paper, we proposed a novel decentralized blockchain-based Key Management protocol for heterogeneous and dynamic networks. Our solution balances the loads between the nodes according to their capabilities. We proved that this makes it efficient and highly scalable. Furthermore, our solution automatically and securely distribute new keys to the members upon a network change. To overcome the disadvantages of centralized approaches, we used the blockchain technology and smart contracts. We showed that the system will continue to operate even if an entity fails and that the compromise of an entity will not jeopardize the security of the whole network. Finally, we proposed an implementation on real IoT platforms to validate the theoretical analysis and the results of simulations. In future woks, we intend to propose a novel lightweight key agreement method to securely and efficiently distribute keys on the limited-constrained nodes.

Acknowledgments

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

References

[1] M. R. 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] S. Athmani, A. Bilami and D. E. Boubiche. "EDAK: An efficient dynamic authentication and key management mechanism for heterogeneous WSNs". In: *Future Generation Computer Systems* 92 (2019), pp. 789–799.

[3] J. Ayuso, L. Marin, A. Jara and A. F. G. Skarmeta. "Optimization of Public Key Cryptography (RSA and ECC) for 16-bits Devices based on 6LoWPAN". In: *1st International Workshop on the Security of the Internet of Things, Tokyo, Japan* (2010), pp. 1–8.

[4] M. Azees, P. Vijayakumar, L. J. Deborah, K. Marimuthu and M. S. Christo. "BBAAS: Blockchain-based anonymous authentication scheme for providing secure communication in VANETs". In: *Security and Communication Networks* 2021 (2021).

[5] 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.

[6] N. Baracaldo, B. Palanisamy and J. Joshi. "G-sir: an insider attack resilient geo-social access control framework". In: *IEEE Transactions on Dependable and Secure Computing* 16.1 (2017), pp. 84–98.

[7] 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.

[8] M. Castro, B. Liskov et al. "Practical Byzantine fault tolerance". In: *OSDI* 99.1999 (1999), pp. 173–186.

[9] K. Chatterjee, A. De and D. Gupta. "An improved ID-Based key management scheme in wireless sensor network". In: *International Conference in Swarm Intelligence* (2012), pp. 351–359.

[10] 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 (2015), pp. 494–502.

[11] 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.

[12] K. Christidis and M. Devetsikiotis. "Blockchains and smart contracts for the internet of things". In: *Ieee Access* 4 (2016), pp. 2292–2303.

[13] X. Du, Y. Xiao, M. Guizani and H.-H. Chen. "An effective key management scheme for heterogeneous sensor networks". In: *Ad Hoc Networks* 5.1 (2007), pp. 24–34.

[14] A. Dunkels, J. Eriksson, N. Finne and N. Tsiftes. "Powertrace: Network-level power profiling for low-power wireless networks". In: *SICS Technical Report* 5 (2011), pp. 11–25.

[15] M. Eltoweissy, M. Moharrum and R. Mukkamala. "Dynamic key management in sensor networks". In: *IEEE Communications magazine* 44.4 (2006), pp. 122–130.

[16] C. Esposito, M. Ficco, A. Castiglione, F. Palmieri and A. De Santis. "Distributed group key management for event notification confidentiality among sensors". In: *IEEE Transactions on Dependable and Secure Computing* 17.3 (2018), pp. 566–580.

[17] Exp5438. "MSP430F5438 Experimenter Board". In: *URL: https://www.ti.com/tool/MSP-EXP430F5438descriptionArea* (2022).

[18] J. T. George. "Proof of Stake: Consensus of the Future". In: *Introducing Blockchain Applications* (2022), pp. 107–123.

[19] E. I. W. Group and I. IoT. "IoT Developer Survey". In: *URL: https://iot.eclipse.org/community/resources/iot-surveys/* (2019).

[20] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan. "Energy-efficient communication protocol for wireless microsensor networks". In: *Proceedings of the 33rd annual Hawaii international conference on system sciences* (2000), pp. 10–20.

[21] F. Hendaoui, H. Eltaief and H. Youssef. "UAP: A unified authentication platform for IoT environment". In: *Computer Networks* 188 (2021), p. 107811.

[22] A. S. Hosen, G.-h. Cho et al. "A robust key management scheme based on node hierarchy for wireless sensor networks". In: *International conference on computational science and its applications* (2014), pp. 315–329.

[23] T. Inc. "Tendermint". In: *URL: https://docs.tendermint.com* (2022).

[24] M. A. Kandi, D. E. Kouicem, H. Lakhlef, A. Bouabdallah and Y. Challal. "A Blockchain-based Key Management Protocol for Secure Device-to-Device Communication in the Internet of Things". In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE. 2020, pp. 1868–1873.

[25] M. A. Kandi, H. Lakhlef, A. Bouabdallah and Y. Challal. "A Key Management Protocol for Secure Device-to-Device Communication in the Internet of Things". In: *IEEE Global Communications Conference (Globecom'19)* (2019), pp. 1–6.

[26] M. A. Kandi, H. Lakhlef, A. Bouabdallah and Y. Challal. "A versatile Key Management protocol for secure Group and Device-to-Device Communication in the Internet of Things". In: *Journal of Network and Computer Applications* 150 (2020), p. 102480.

[27] M. A. Kandi, H. Lakhlef, A. Bouabdallah and Y. Challal. "An Efficient Multi-Group Key Management Protocol for Heterogeneous IoT Devices". In: *IEEE Wireless Communications and Networking Conference (WCNC)* (2019), pp. 1–6.

[28] M. A. Kandi, H. Lakhlef, A. Bouabdallah and Y. Challal. "An Efficient Multi-Group Key Management Protocol for Internet of Things". In: *26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)* (2018), pp. 1–6.

[29] F. Kausar, S. Hussain, L. T. Yang and A. Masood. "Scalable and efficient key management for heterogeneous sensor networks". In: *The Journal of Supercomputing* 45.1 (2008), pp. 44–65.

[30] K. Keerthi, C. Rebeiro and A. Hazra. "An Algorithmic Approach to Formally Verify an ECC Library". In: *ACM Transactions on Design Automation of Electronic Systems* 23.5 (2018).

[31] J. Y. Kim, W. Hu, H. Shafagh and S. Jha. "Seda: Secure over-the-air code dissemination protocol for the internet of things". In: *IEEE Transactions on Dependable and Secure Computing* 15.6 (2016), pp. 1041–1054.

[32] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. A. Ogah and Z. Sun. "Blockchain-based dynamic key management for heterogeneous intelligent transportation systems". In: *IEEE Internet of Things Journal* 4.6 (2017), pp. 1832–1843.

[33] A. Lei, C. Ogah, P. Asuquo, H. Cruickshank and Z. Sun. "A secure key management scheme for heterogeneous secure vehicular communication systems". In: *ZTE Communications* 14.S0 (2019), pp. 21–31.

[34] J. Liu, X. Li, Q. Jiang, M. S. Obaidat and P. Vijayakumar. "Bua: A blockchain-based unlinkable authentication in vanets". In: *IEEE International Conference on Communications (ICC)* (2020), pp. 1–6.

[35] Z. Liu, X. Huang, Z. Hu, M. K. Khan, H. Seo and L. Zhou. "On emerging family of elliptic curves to secure internet of things: ECC comes of age". In: *IEEE Transactions on Dependable and Secure Computing* 14.3 (2017), pp. 237–248.

[36] K. Lu, Y. Qian, M. Guizani and H.-H. Chen. "A framework for a distributed key management scheme in heterogeneous wireless sensor networks". In: *IEEE transactions on wireless communications* 7.2 (2008), pp. 639–647.

[37] M. Ma, G. Shi and F. Li. "Privacy-Oriented Blockchain-based Distributed Key Management Architecture for Hierarchical Access Control in the IoT Scenario". In: *IEEE Access* 7 (2019), pp. 34045–34059.

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

[39] G. Mehmood, M. S. Khan, A. Waheed, M. Zareei, M. Fayaz, T. Sadad, N. Kama and A. Azmi. "An Efficient and Secure Session Key Management Scheme in Wireless Sensor Network". In: *Complexity* 2021 (2021).

[40] MicaZ. "Micaz Wireless Measurement System". In: *URL: https://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf* (2022).

[41] S. Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: *Decentralized Business Review* (2008), p. 21260.

[42] M. Omar, I. Belalouache, S. Amrane and B. Abbache. "Efficient and energy-aware key management framework for dynamic sensor networks". In: *Computers & Electrical Engineering* 72 (2018), pp. 990–1005.

[43] Openmote. "OpenMote-cc2538". In: *URL: https://doc.riot-os.org/group_boards_openmote-cc2538.html* (2022).

[44] T. C. Priyadharshini and D. M. Geetha. "Efficient Key Management System Based Lightweight Devices in IoT". In: *Intelligent Automation and Soft Computing* 31.3 (2022), pp. 1793–1808.

[45] Z. Qin, X. Zhang, K. Feng, Q. Zhang and J. Huang. "An efficient identity-based key management scheme for wireless sensor networks using the bloom filter". In: *Sensors* 14.10 (2014), pp. 17937–17951.

[46] 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.

[47] K. Rizki. "Efficient Group Key Management for Internet of Things". In: *School of Electrical Engineering (EES)* 2016 (2016), pp. 1–115.

[48] 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 (2012), pp. 2224–2237.

[49] F. Saleh. "Blockchain without waste: Proof-of-stake". In: *The Review of financial studies* 34.3 (2021), pp. 1156–1190.

[50] S. 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.

[51] A. Singh, A. N. Tentu and V. C. Venkaiah. "A dynamic key management paradigm for secure wireless ad hoc network communications". In: *International Journal of Information and Computer Security* 14.3-4 (2021), pp. 380–402.

[52] S. Singh, A. Khan and T. Singh. "A New Key Management Scheme for Wireless Senm Networks using an Elliptic Curve". In: *Indian Journal of Science and Technology* 10.13 (2017), pp. 1–7.

[53] Sonhan. "Sample Data for powertrace using CM5000 motes". In: *URL: https://github.com/sonhan/contiki/tree/master/apps/powertrace-sonhan/sample-data* (2015).

[54] Telosb. "Telosb Mote Platform". In: *URL: https://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf* (2022).

[55] G. K. C. Thevar and G. Rohini. "Energy efficient geographical key management scheme for authentication in mobile wireless sensor networks". In: *Wireless Networks* 23.5 (2017), pp. 1479–1489.

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

[57] I.-C. Tsai, C.-M. Yu, H. Yokota and S.-Y. Kuo. "Key management in Internet of Things via Kronecker product". In: *IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)* (2017), pp. 118–124.

[58] Y.-M. Tseng, J.-L. Chen and S.-S. Huang. "A Lightweight Leakage-Resilient Identity-Based Mutual Authentication and Key Exchange Protocol for Resource-limited Devices". In: *Computer Networks* 196 (2021), p. 108246.

[59] P. Vasin. "Blackcoin's proof-of-stake protocol v2". In: *URL: https://blackcoin.org/blackcoin-pos-protocol-v2-whitepaper.pdf* 71 (2014).

[60] L. Veltri, S. Cirani, S. Busanelli and G. Ferrari. "A novel batch-based group key management protocol applied to the internet of things". In: *Ad Hoc Networks* 11.8 (2013), pp. 2724–2737.

[61] P. Vijayakumar, S. Bose and A. Kannan. "Rotation based secure multicast key management for batch rekeying operations". In: *Networking Science* 1.1-4 (2012), pp. 39–47.

[62] P. Vijayakumar, S. Bose and A. Kannan. "Chinese remainder theorem based centralised group key management for secure multicast communication". In: *IET information Security* 8.3 (2014), pp. 179–187.

[63] P. Vijayakumar, V. Chang, L. J. Deborah and B. S. R. Kshatriya. "Key management and key distribution for secure group communication in mobile and cloud network". In: *Future Generation Computer Systems* 84 (2018), pp. 123–125.

[64] W. Viriyasitavat and D. Hoonsopon. "Blockchain characteristics and consensus in modern business processes". In: *Journal of Industrial Information Integration* 13 (2019), pp. 32–39.

[65] C. Wan. "IBKES: Efficient Identity-Based Key Exchange with Scalability for Wireless Sensor Networks Using Algebraic Signature". In: *Adhoc & Sensor Wireless Networks* 39 (2017).

[66] J. Wang, H. Wang, X. A. Wang and Y. Cao. "An Authentication Key Agreement Scheme for Heterogeneous Sensor Network Based on Improved Counting Bloom Filter". In: *10th Interna-*

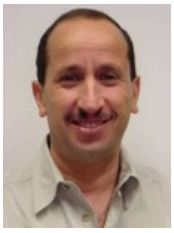
tional Conference on P2P, Parallel, Grid, Cloud and Internet Computing (2015), pp. 815–820.
[67] Y. Wen, F. Lu, Y. Liu and X. Huang. "Attacks and countermeasures on blockchains: A survey from layering perspective". In: *Computer Networks* 191 (2021), p. 107978.
[68] C. K. Wong, M. Gouda and S. S. Lam. "Secure group communications using key graphs". In: *IEEE/ACM transactions on networking* 8.1 (2000), pp. 16–30.
[69] M. S. Yousefpoor and H. Barati. "Dynamic key management algorithms in wireless sensor networks: A survey". In: *Computer Communications* 134 (2019), pp. 52–69.
[70] Z. Yu and Y. Guan. "A robust group-based key management scheme for wireless sensor networks". In: *IEEE Wireless Communications and Networking Conference* 4 (2005), pp. 1915–1920.
[71] Z1. "The Z1 mote". In: *URL: https://github.com/Zolertia/Resources/wiki/The-Z1-mote* (2022).
[72] 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.
[73] A. Zhang and X. Lin. "Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain". In: *Journal of medical systems* 42.8 (2018), p. 140.
[74] 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.
[75] Y. Zhang, X. Li, J. Liu, J. Yang and B. Cui. "A secure hierarchical key management scheme in wireless sensor network". In: *international journal of distributed sensor networks* 8.9 (2012), p. 547471.

Dr. Mohamed Ali Kandi is an associate professor at University Paul Sabatier (UPS), France. He obtained his Ph.D. degree from the University of Technology of Compiegne in 2020 (UMR CNRS 7253). He received a double degree (Master and Engineer) in Computer Science from the High National School of Computer Science of Algiers (ESI) in 2017. His main research interests are trusted computing schemes (especially the blockchain technology and smart contracts) to address the challenges (mainly security and privacy) facing the Internet of Things.

Dr. Djamel Eddine Kouicem obtained his Ph.D degree from the University of Technology of Compiegne (UMR CNRS 7253) in 2019. In September 2016, he gained a MSC diploma in Computer Science - Networking from Pierre & Marie Curie university (Paris 6) in France. In July 2015, he received Engineer Diploma in computer science from the High National School of Computer Science in Algiers. His research interests are in security and privacy in internet of things and Networking.

Messouad Doudou is currently working as a research Engineer at Itron R&D since 2020. He obtained his Ph.D. in 2016 from University of Science and Technology Houari-Boumedienne, USTHB. He was a full-time researcher at CERIST Research Center 2009-2017 and a postdoc at the University of Technology of Compiègne, Sorbonne Universities, France 2017-2019. He worked on the design and optimization of efficient low duty-cycle MAC protocols, involving low power, low latency, reliability, and QoS support in sensor networks. His research interest includes energy efficiency and QoS in wireless low-power networks, and data cleaning, processing, and analysis in IoT and Cyber-Physical systems.

Dr. Hicham Lakhlef is an associate professor at the University of Technology of Compiegne (UMR CNRS 7253). During the year 2015/2016 he was a temporary researcher in IRISA, University of Rennes 1 (UMR CNRS 6074). During the year 2014/2015 he was a temporary teaching assistant and researcher at the University of Franche-Comté/ FEMTO-ST institute (UMR CNRS 6174). He obtained his Ph.D degree from the University of Franche-Comté in 2014. He obtained his Master's degree from the University of Picardie Jules Verne in 2011. His research interests are in parallel and distributed algorithms, WSNs, clustering, optimization, routing, and IoT.



Pr. Abdelmadjid Bouabdallah received the Master (DEA) degree and Ph.D. from university of Paris-sud Orsay (France) respectively in 1988 and 1991. From 1992 to 1996, he was Assistant Professor at university of EvryVal-d'Essonne (France) and since 1996 he is Professor at University of Technology of Compiègne (UTC), where he is leading the Networking & Security research group and the Interaction & Cooperation research of the Excellence Research Center LABEX MS2T. His research Interest includes Internet QoS, security, unicast/multicast communication, Wireless Sensor Networks, and fault tolerance in wired/wireless networks. He conducted several large-scale research projects funded by Motorola Labs., Orange Labs., ANRRNRT, CNRS, and ANR-Carnot.

Dr. Yacine Challal is associate professor at Compiegne University of Technology (France). He is member of the networking and optimization research team at the UMR-CNRS-7253 Heudiasyc Lab. He got his Ph.D. and Master degrees respectively in 2005 and 2002 from Compiegne University of Technology. He got his engineering degree from National Institute of Informatics (Algiers) in 2001. His research interests include security in group communication, security in wireless mobile ad hoc networks and wireless sensor networks, and fault tolerance in distributed systems.