IFKMS: Inverse Function-based Key Management Scheme for IoT networks

Mohammed Nafi[a,c,*], Mohamed-Lamine Messai[b], Samia Bouzefrane[c], Mawloud Omar[d]

[a] *Laboratoire d'Informatique Médicale, Faculté des Sciences Exactes, Université de Bejaia,06000 Bejaia, Algérie*
[b] *Univ Lyon, Univ Lyon 2, ERIC*
[c] *CEDRIC Lab, Conservatoire National des Arts et Métiers -CNAM, Paris, France*
[d] *IRISA, Université Bretagne Sud, Vannes, France*

Abstract

Due to the rapid growth of the Internet of Things (IoT), secure communication is becoming a significant concern. Nodes that compose such a dynamic network need to exchange sensitive and valuable data. The data must be kept safe against attacks. This protection requires the development of efficient key management protocols. However, this task is challenging because of the high resource constraints of most IoT devices in terms of storage, communication, processing, and energy capabilities. Some existing key management techniques have certain weaknesses since sensitive parameters are not protected during transmission, and cryptographic keys are stored in plain text and usually renewed at a fixed period of time. This paper proposes a new key management protocol aiming to secure communications before and after key establishment. Our scheme uses hash and one-one functions to achieve security during the key establishment process. The symmetrical character of the invertible functions is thus exploited to conceal critical data and pairwise keys stored in nodes' memories. Moreover, the key refresh period is variable, which can be adjusted according to the number of occurred attacks in the network. BAN (Burrows Abadi Needham) logic is employed to assess the correctness of the proposed scheme. The results show that our scheme operates correctly and does not have redundancies or security flaws. Furthermore, the security and performance analysis point out that the proposed scheme is resilient against well-known attacks, and efficient in terms of

*Corresponding author
Email address: mohammed.nafi@univ-bejaia.dz (Mohammed Nafi)

storage, communication, computation overhead, and energy consumption.

Keywords: Security, Key management, IoT, Inverse function, BAN logic
Variable refresh period, Hash function, One-one function

---

1. Introduction

Internet of Things (IoT) refers to a system that interconnects various types of autonomous and heterogeneous everyday objects. These devices are connected to the Internet and are provided with unique identifiers and communication capabilities. This allows them to exchange data with each other and with their environment over a network without any human intervention [1]. IoT belongs to the dynamic networks category since its composition is not fixed and frequently changes following the addition and deletion of nodes after deployment. The goal is to extend the network and replace faulty nodes [2]. These networks have gained a lot of interest due to their wide range of application fields, including health care, smart home, smart cities, environment, agriculture, etc.

Devices that compose such a dynamic network sometimes have to exchange sensitive data. For instance, in a healthcare scenario, the sensors collect and transmit patients' physiological data toward the server to be reachable and analyzed by doctors. So, two legitimate entities need to communicate securely. An adversary may try to intercept, alter or delete exchanged sensitive data. Securing such communication is a real challenge. Nodes mainly communicate over vulnerable wireless channels, subject to passive and active attacks. Devices have high resource constraints in storage, processing, communication, and energy capabilities. Heterogeneity in terms of resources and dynamic topology resulting from nodes' mobility and frequent changes in network composition, etc. To provide security, one or more cryptographic keys is required. However, key management is complex, including key establishment, distribution, revocation, renewal, and addition.

Several security solutions have been proposed in the literature to address the issue of key management in dynamic networks. However, most of them suffer from two main problems that must be fulfilled. The first one is that sensitive parameters such as nodes' and keys' identifiers are usually transmitted alone and in clear text during the keys discovery phase. This is a point of weakness that may be exploited by an adversary to easily attack the network by doing man-in-the-middle and denial of service (DoS) attacks. Thus, recipient nodes cannot detect changes that an attacker might make

during the transmission of these sensitive data over an insecure wireless communication channel. The second problem is that keys are always renewed at a specific moment. The key refresh period is fixed and does not adapt to the network's dynamic over time. An attacker only needs to compromise a node once to obtain this information. Consequently, he or she only has to strengthen his or her attack during its expiration in order to succeed.

This paper proposes a new key management scheme, termed IFKMS (Inverse Function-based Key Management Scheme for IoT networks), which corrects the previously highlighted shortcomings of some existing solutions. The main objective is to support both resilience and efficiency. In fact, the proposed scheme mainly uses one-one functions to conceal the stored keys and the sensitive data sent over an insecure channel before the key establishment phase. The critical parameters are thus never sent alone, but coupled with other information, which gives the recipient the ability to detect any message alteration. Moreover, our scheme uses a variable refresh period, which increases resilience because not only it adds difficulties to the adversary to find out the value of the refresh period, but also the current keys are frequently refreshed in the presence of attacks. It also improves efficiency since nodes save their energy in the absence of attacks.

The main features of the proposed key management scheme are enumerated in the following points:

- The proposal better secures communications between different legitimate entities not only after pairwise keys establishment, but also during this process;
- The sensitive data are never sent in plain text but always transmitted in a secure manner. Indeed, the data is hidden by using both one-one and hash functions or encrypted with the established pairwise keys;
- The established pairwise keys are protected using invertible functions associated with nodes before their storage;
- The refresh period is not fixed and can be dynamically adjusted according to the number of occurred attacks. In fact, this period increases if no attacks are detected and decreases otherwise. As a result, our scheme supports both efficiency and resilience;
    - Efficiency in terms of energy consumption is guaranteed because in the absence of attacks, the constrained devices must wait for a long time before renewing their secret keys;

  - Resilience is also ensured because the secret keys are frequently refreshed in the presence of attacks;

- The key revocation phase does not require any exchange of messages between current members, which considerably reduces costs in terms of communication and energy consumption;

- The proposed scheme allows node addition and revocation after the initial deployment phase.

The proposed scheme can be used in many areas, including the Medical Internet of Things, where wearable devices and sensors continuously collect patient physiological data, and securely transmit it to the server so that doctors can access and process it in real-time. This allows them to intervene in time and thus improve patients care.

The remainder of this paper is organized as follows. Section 2 recalls some basic mathematical concepts used in this paper. An overview of some related works is given in Section 3. Our proposed key-management scheme is described in detail in Section 4. Section 5 discusses the scheme's formal and informal security analysis. Section 6 highlights the performance evaluation of our scheme compared with recent existing schemes according to storage, communication, computation costs, and energy consumption. Finally, Section 7 concludes the paper and gives some future directions.

2. Mathematical preliminaries

This section briefly gives the basic concepts of one-one and inverse functions used in our scheme for security purposes.

2.1. One-one function

A one-one function f is a function where each element x in the domain D_f corresponds exactly to one element, called image $y = f(x)$ in the range or co-domain R_f and vice versa. This means, no two different elements in the domain of f have the same image in the range, and no element y in the range is the image of more than one element x in the domain. A one-one function f can therefore be expressed as :
if $f(x1) = f(x2) \implies x1 = x2$, or equivalently,
if $x1 \neq x2 \implies f(x1) \neq f(x2)$.
It is important to note that only one-one functions have an inverse that is also a one-one function.

2.2. Inverse of one-one function

The inverse of a one-one function f with domain D_f and range R_f, denoted f^{-1}, is the function with domain R_f and range D_f, which is defined by $f^{-1}(y) = x$ for any $y = f(x)$ in the range of f. In other words, the inverse function f^{-1} of f is defined as: $\{(y, x)|(x, y) \in f\}$, that means f^{-1} is obtained by interchanging the coordinates in each ordered pair of f.

2.3. Properties of one-one function

The main properties of an invertible function f are listed as follows:

- The domain of f equals to the range of $f^{-1}(D_f = R_{f^{-1}})$ and the range of f equals to the domain of $f^{-1}(R_f = D_{f^{-1}})$.
- Each function reverses what the other function does. That means, $f^{-1}(f(x)) = x$ for all x in the domain of f and $f(f^{-1}(x)) = x$ for all x in the domain of f^{-1}.
- The graph of an invertible function and the graph of its inverse are symmetric with respect to the line $y = x$. That means, (x, y) is on the graph of f if and only if (y, x) is on the graph of f^{-1}.
- Strong-collision resistance: It is impossible to find two different input values that give the same output or image.

3. Related work

This section reviews some related works presented in the literature to address key management issue in dynamic networks. These key establishment schemes can be classified into two main categories, namely probabilistic and deterministic. This paper mainly focuses on key pool-based, polynomial pool-based, and hybrid schemes. The last technique consists on the combination of both key and polynomial pool-based schemes.

In [2], the authors proposed a random key pre-distribution scheme (RKP) for sensor networks, called the basic scheme. Before deployment, each sensor is assigned a subset of keys, called key ring, randomly chosen from a large key pool. Every node broadcasts the list of keys' identifiers belonging to its key ring in the key discovery phase. Any pair of sensors share at least one key with a certain probability. If they do not find a common key, a path key is established by relying on an intermediate node, which has a common key with both sensors. The main drawback of this scheme is its resilience to node capture attack. Since the keys are chosen randomly from the same key pool,

more than one pair of nodes may share the same key. Hence, compromising a node affects its links and some other links in the network. Thus, when the number of compromised nodes increases, the fraction of affected links quickly increases. In addition, whenever the key identifiers are sent in clear text during key discovery, an attacker could use them to launch attacks.

A key management scheme based on symmetric polynomial is presented by Blundo et al. [3]. The idea is that a server selects a symmetric bivariate polynomial $P(x, y)$ of degree t with coefficients randomly chosen over F_q, where q is a prime number large enough $q > n$. The server gives each user i the polynomial $P(i, y)$, which is obtained by evaluating the initial polynomial $P(x, y)$ at point $x = i$. The scheme enables a pair of sensors i and j to establish a pairwise key by evaluating $P(i, j)$ and $P(j, i)$ respectively. The scheme is t-secure since it can resist to t captured nodes. Moreover, it does not require any communication during pairwise keys establishment. However, when the network size increases, an adversary may easily compromise more than t sensors, which may compromise the whole network.

Liu et al. [4], proposed a framework for polynomial pool-based key predistribution that enables two sensors to establish a pairwise key between them. This scheme is similar to the probabilistic key distribution or key pool [2] when the degree of all the polynomials is equal to zero, and it is equivalent to the polynomial-based key pre-distribution protocol [3] when the size of the polynomial pool is equal to one. The scheme provides better security when compared to the previous ones. It is more resilient to a node capture attack since an attacker has to find out more than t shares of the same polynomial to be able to disclose pairwise keys of non-compromised nodes. In addition, unlike the previous scheme, the same key is never shared by multiple couples of sensors. However, some sensitive data like polynomial IDs are sent in clear text during polynomial share discovery.

An efficient multi-party key management scheme (EKM) for securing group communication in resources constrained networks is proposed by Mahmood et al. [5]. The network is first organized into clusters. After receiving an encrypted identifier from each member, cluster heads perform XOR operations on the subset of randomly picked hash values of these identifiers to generate the polynomial. The latter is then securely sent to members. The scheme preserves the anonymity of nodes because their identifiers are not transmitted in clear text. When generating the polynomial, it also reduces computation cost by performing light XOR operations instead of expensive multiplication. Moreover, EKM allows node addition and migration. However, its communication overhead remains high, and the polynomial needs to be refreshed after each node addition or deletion. Furthermore, too many

operations are still performed by cluster heads, which may become a point of weakness for the network.

The authors in Kumar et al. [6] presented a k-degree multivariate polynomial-based session key computation protocol for securing communication in a dynamic group called NISKC. The scheme is non-interactive because the session key is computed at the group members' side without exchanging them. Initially, a trusted server generates polynomial shares for every member who wants to participate in group communication. The private values of group members constitute the polynomial variables. The polynomial shares are then securely distributed to participants. The latter have to insert their values into the polynomial to be able to compute a common session key. The scheme allows new members to join and current ones to leave the group during the communication process. The scheme also reduces the communication and computation overhead during the generation of the symmetric session key. However, the polynomial must be regenerated every time there is a change in group membership.

The authors in [7] proposed a key establishment scheme for wireless sensor networks based on both polynomial and probabilistic key pre-distribution schemes. Before network deployment, the server generates a polynomial and key pools. Each polynomial and key are assigned with a distinct identifier. The server randomly assigns a set of polynomials shares and keys with their corresponding identifiers for each sensor. Two nodes can establish a communication key if they have at least one shared key, a common polynomial, or the pre-loaded keys of one node are computed using the polynomial shares selected for the other node. This scheme provides better resilience against node capture attacks than the previous works. However, it requires a large amount of memory to store keys and polynomial shares. In addition, the more polynomials and keys are stored in each node, the greater the connectivity and the lower the security. Thus, a trade-off between storage, connectivity, and resilience to node capture attacks must be established.

Lu et al. proposed in [8] a distributed key management framework in heterogeneous wireless sensor networks. The proposed framework is based on a random key pre-distribution scheme and a polynomial key pre-distribution scheme. The network is divided into several classes, where each class contains sensors having the same resource capabilities. The lower class holds the least powerful nodes, and the higher one includes larger resource nodes in terms of communication, processing, and energy capabilities. Like previous key establishment schemes, this framework also consists of three phases: initialization, direct key, and path key setup. In the first step, the server considers the heterogeneity features of sensor nodes when distributing them

polynomial shares. Direct and path keys are then established during the second and third steps, respectively. Classifying sensors based on their communication ranges, processing capability, and energy levels helps designing the network. However, the authors do not present how these classes are formed and how classes evolve through the network lifetime when the energy levels of sensors decrease.

In Manasrah et al. [9], a key management scheme in wireless sensor networks (WSN) is proposed. The scheme combines polynomial pool-based and matrix-based schemes and uses the block LU decomposition algorithm presented in [10]. This approach consists of three phases, namely polynomial pool generation, polynomials pre-distribution, and key generation. In the first phase, a large pool of bivariate symmetric t-degree polynomials is generated by the base station. In the second phase, an n by n matrix is constructed from a set of polynomials chosen from the pool, where n is the size of the network. An LU decomposition of the obtained matrix is then performed. Each sensor i is pre-loaded with a row_i of the matrix L and the corresponding $column_i$ of the matrix U, as well as the univariate polynomial $P(i, y)$. In the last phase, two sensors generate a common key after authenticating each other by exchanging their identifiers and the rows of their L matrices. In this scheme, each sensor needs to store $(t + 1)$ coefficients of the polynomial and $N/2$ elements of the row of the matrix L and the same number of the column of the matrix U. However, this approach did not address new node addition, key renewal, and revocation that are important in WSNs.

Ashwag Albakri et al. [11] proposed a new polynomial-based key distribution scheme in hierarchical WSN. The proposed scheme aims to reduce the impact of sensor capture attacks. The scheme includes three phases that are token generation, key establishment, and revocation. Three types of polynomials are used: trivariate, bivariate, and univariate, which are held respectively by the sink, cluster heads, and sensors. Sensors' tokens are generated using a trivariate polynomial $F(x, y, z)$ and a prime number p. The parameters x, y, and z are of degree $(k - 1)$, $(t - 1)$, and $(h - 1)$, respectively. Two types of keys are also employed: unicast and broadcast keys. The scheme requires low communication overhead, and its resistance to node capture attack is mainly related to the thresholds k, t, and h. The scheme withstands the capture of $(k-1)$ cluster heads (CH). In other words, if more than k CH are captured, an attacker can reconstruct the trivariate polynomial and thus compromise the whole network. Moreover, if at least t sensors of the same cluster are captured, an attacker can recover the bivariate polynomial and thus compromise all the clusters. However, the addition

of new nodes to the network is not considered in this scheme.

A distributed and dynamic key management scheme for homogeneous wireless sensor networks has been introduced in [12]. This scheme uses three types of keys: network key shared by all the sensors, cluster key known only to the cluster's members, and pairwise key established by a couple of nodes. Every node is pre-loaded with a recursive function with the first term. It is also assigned with m keys picked from m selected key chains. Two nodes can securely communicate if they share a common key or establish a pairwise key using the nonconvergent numerical sequence. The scheme is scalable and efficient in terms of communication overhead. It also allows node addition and key refresh. However, the storage overhead remains high.

A new set of hybrid hierarchical key distribution protocols in the context of smart cities, called H2KD are proposed in [13]. The scheme uses both elliptic curve and symmetric cryptography. It is composed of four phases, which are key pre-deployment, key initialization, key establishment and key updating. Required parameters are first distributed to the nodes during the first and second phases. Master, public/private and session keys are then established at the third phase. New keys are generated in the last phase when a mobile node changes its location. The scheme supports mobility, heterogeneity, scalability and resilience. However, some messages are transmitted in clear text during key updating phase for instance. As a result, an adversary could easily alter them in order to disturb the key establishment process.

Recently, a new key establishment scheme based on Rabin cryptosystem is proposed in [14]. The Rabin cryptosystem is an asymmetric encryption algorithm which relies on integer factorization problem. The proposed scheme is considered in a smart home IoT application where a trustworthy third party initializes the system and registers all devices in the smart home. The third party generates keys and a hash function is used to register devices. Then, different steps for establishing session keys are described to secure communication. The proposed scheme lacks an energy evaluation for the use of an asymmetric cryptography algorithm. The presence in the network of a trusted third entity does not apply to all IoT network applications more precisely where the devices are deployed in hostile places. In addition, the proposed scheme is vulnerable to integer factorization attacks.

More recently, authors in [15] presented a key distribution method based on elliptic curve cryptography (ECC) in IoT networks. An improved version of the well known protocol, called LEACH [16] has also been proposed. The network is organized into clusters. The proposed method consists of four phases. First, the number of clusters is identified and cluster-heads are

selected after authentication of all nodes. Next, IoT nodes use the ECC method to establish and distribute keys. Then, routing tables are generated. Finally, the re-keying method is described when the topology of the clusters changes. A drawback of this method is that the authors do not present how the number of clusters is identified. In addition, the use of ECC is still an energy consuming process which can shorten the lifetime of nodes. Moreover, the re-keying step requires an exchange of messages between IoT nodes, but the authors do not show its impact on the resources of nodes.

4. IFKMS: Inverse Function-based Key Management Scheme

This section presents our solution for key management in IoT networks by focusing mainly on the system models, the notations used in this paper, and the different phases of the proposed scheme.

4.1. System models

This subsection presents the two models adopted in the proposed scheme: the network and threat models.

4.1.1. Network model

The proposed key management approach uses the network model presented in [17], which is illustrated in Fig. 1 and includes the following three components:

- *Server node (S)*: This performs several heavy operations, such as the computation of the inverse functions corresponding to the selected invertible functions. Moreover, we assume that the server is equipped with an Intrusion Detection System (IDS) and can detect attacks.
- *Gateway node (G)*: This acts as an intermediary between the server and the highly constrained nodes since the latter are not within the communication range of the server.
- *Constrained node (N)*: these nodes carry out operations that are as light as possible to preserve their resources.

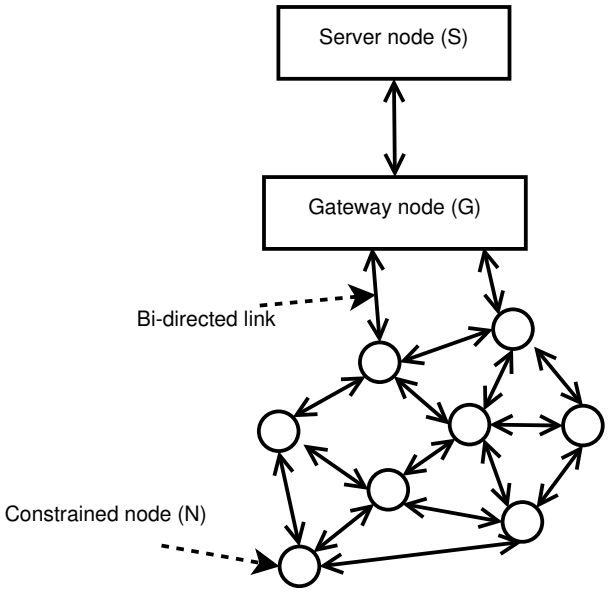


Fig. 1 Network model

4.1.2. Threat model

We use the widely known Dolev-Yao threat model [18], where each participant transmits messages over an unsecured medium. According to this model, an adversary may:

– eavesdrop on all the messages sent in an insecure way;

– alter, delete or replay intercepted messages;

– capture any node and have access to its memory.

4.2. Notations

The notations with their corresponding descriptions used in this paper are illustrated in Table 1.

4.3. Phases of the proposed scheme

The proposed scheme includes five important phases, namely (1) initialization, (2) key establishment, (3) key refresh, (4) node addition, and (5) key revocation. These phases are detailed below.

Notation	Description
i	Identifier of the node i
f_i	Non-usual invertible function assigned to node i
D_f, R_f	Domain and range of the function f respectively
f_i^{-1}	Inverse function of f_i
ri	Random number generated by node i
$xi \Vert xj$	Concatenation of xi with xj
Tr	Refresh period
T_{min}	Minimum value of refresh period
T_{max}	Maximum value of refresh period
K_{ij}	Shared key between nodes i and j
$\{M\}k$	Message M encrypted with the key k
α	Security parameter

Table 1 Notations

4.3.1. Initialization phase

The setup phase is carried out for the purpose of initializing IoT devices with involved key materials prior to network deployment. First, the server node randomly generates a pool of n invertible functions f over a definition domain D_f, and computes the pool of their corresponding inverse functions, denoted f^{-1}, where n represents the network size. After that, for each node i, the server assigns a one-one function f_i with its corresponding inverse function f_i^{-1}. Initially, each device i that compose the network is pre-loaded with one 3-tuple (i, f_i, f_i^{-1}) and $(n-1)$ 2-tuples (j, f_j), with $j = 1, ..., n$ and $j \neq i$. The nodes are also pre-distibuted with a common hash function $h()$, and the refresh period Tr along with its minimum T_{min} and maximum T_{max} values. Note that the refresh period is set to the medium value between the two last ones, that is $Tr = (T_{min} + T_{max})/2$.

4.3.2. Key establishment phase

The present phase includes two steps: *neighbor discovery* and *pairwise key generation*, which are described in the following.

a) *Neighbor discovery*

This step is performed for neighborhood discovery purposes. Once the network deployment is established, each node i proceeds to discover the adjacent devices within its communication range. This is carried out by broadcasting its identifier i along with its masked identifier obtained with its one-one function f_i^{-1}. A node i maintains a neighbors vector νi, wherein

it stores the received identifiers from the authentic and direct neighbors. At the end of this step, each node i keeps in its memory only its 3-tuple (i, f_i, f_i^{-1}) and the 2-tuples (j, f_j), where $j \in \nu i$, and erases all the remaining 2-tuples.

b) *Pairwise key generation*

When a node i needs to communicate securely, for the first time, with its direct neighbors j, it first generates a random number ri. It uses its one-one function f_i^{-1} to hide this sensitive data by calculating its image. After that, this node sends a *request* composed of two parts: the first one includes its identifier and the masked information, which is the image of both its identifier i and the secret value ri concatenated together. The second part contains the hash value obtained after applying the hash function to the previous concatenated values before being masked.

$$\mathsf{Node}i \rightarrow \mathsf{Node}j : \mathsf{Request}, i, f_i^{-1}(i \,\|\, ri), h(i \,\|\, ri) \tag{1}$$

Upon receiving this request, the recipient node j uses the appropriate inverse function f_i to extract the initial values hidden by the sender, namely its identifier and the random number. After that, the node checks whether that request is not tampered with by an adversary during its transmission. If the verification is successful, then the latter generates a new random number rj, and responds with a *reply* message that also consists of two parts: the first one is composed of its identifier concatenated with the image of its identifier j, the nonce rj, and the successor of the received random number $ri+1$ all concatenated together. The second part includes the digest of these three last pieces of information concatenated together.

$$\mathsf{Node}j \rightarrow \mathsf{Node}i : \mathsf{Reply}, j, f_j^{-1}(j \,\|\, rj \,\|\, ri + 1), h(j \,\|\, rj \,\|\, ri + 1) \tag{2}$$

After the reception of the *reply*, the node i checks whether it is authentic and has not been altered by an attacker. If the verification succeeds then both nodes i and j proceed to compute the common pairwise key k_{ij} as in the following Eq. 3.

$$K_{ij} = h(ri \,\|\, rj), \quad \mathit{with} \quad ri < rj \tag{3}$$

The node i replies with a message that mainly contains the successor of the received random number encrypted with the established pairwise key K_{ij} afterward.

$$Nodei \rightarrow Nodej : \{ri||rj + 1\}K_{ij} \quad (4)$$

Finally, the node j receives and decrypts the cipher message using the established pairwise key K_{ij}. The key generation phase is depicted in Fig. 2.

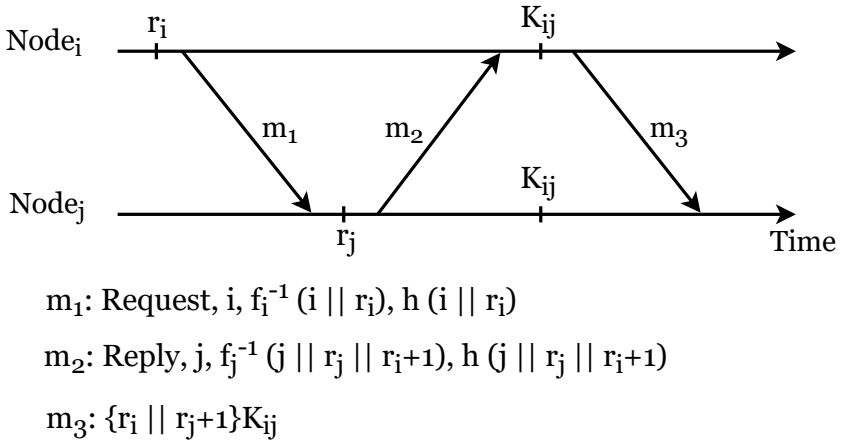


Fig. 2 Pairwise key generation phase

It is important to note that keys are subsequently concatenated with the secret values and stored in nodes' memories after they have been concealed with their appropriate one-one functions. As a result, this avoids storing keys in plain text, which reduces the impact of capture node attacks. As an example, the node i stores $f_i(K_{ij}||r_i)$, with r_i is the random value.

Illustrative example. The Fig. 3 gives an example of the key establishment process between the two nodes 1 and 2. The network is composed of three nodes. During the setup phase, each node i, i=1,2,3 is pre-loaded with one 3-tuple (i, f_i, f_i^{-1}) and two 2-tuples (j, f_j), with $i \neq j$. At the deployment phase, each node i broadcasts a message $i, f_i^{-1}(i)$ in order to discover its neighbors. For instance, the node 1 broadcasts $1, f_1^{-1}(1)$. Both nodes 2 and 3 receive that message. Therefore, they update their neighbor vector by adding the node 1. At the end of this phase, each node keeps in its memory only its one 3-tuple and the 2-tuples of its neighbors. For example, the node 2 has one neighbor 1. Thus, it has one 3-tuple $(2, f_2, f_2^{-1})$ and one 2-tuple $(1, f_1)$. Finally, to establish a pairwise key with the node 2, the node 1 sends a request and receives a reply. As a result, both nodes 1 and 2 generate the same pairwise key K_{12}.

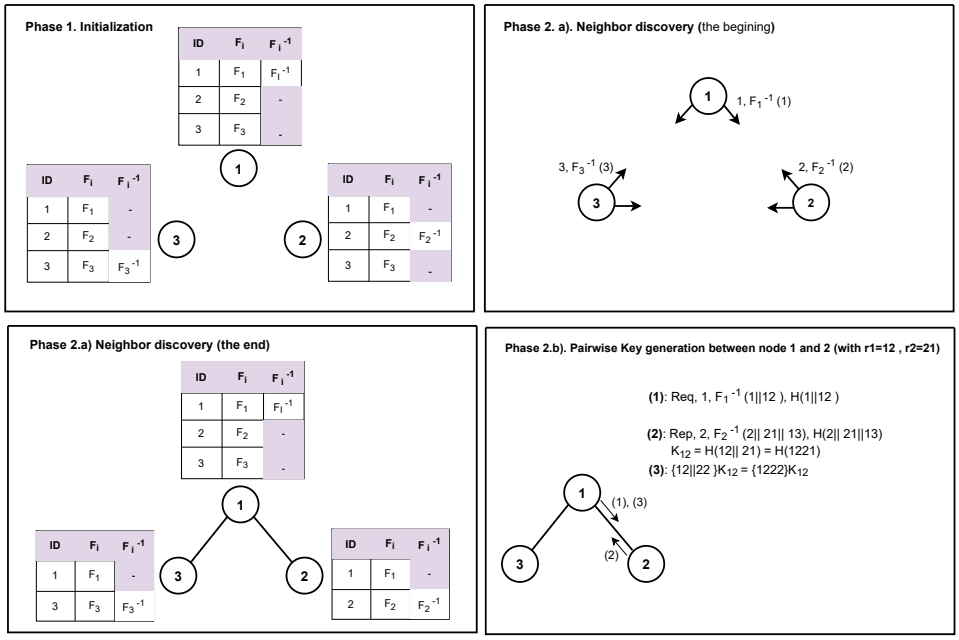


Fig. 3 Example of pairwise key generation phase

4.3.3. Key refresh phase

The keys must be refreshed periodically to counter some security attacks. When the same keys are used for a long time, an adversary would be able to discover them by a brute-force attack. In the proposed scheme, the nodes refresh the keys by generating new random numbers after each period Tr, and re-run the key generation phase. Furthermore, when the expiration time of $(2 * \mathsf{Tr})$ is reached, the server chooses a set of new invertible functions, computes their corresponding inverses, and transmits them to end devices through gateways in a secure manner. Therefore, every device replaces the old inverse functions with the appropriate new ones and performs the key generation phase. Note that the value of the timer changes continuously depending on the number of occurred attacks in the system. Whenever no attack happens during the renewal period Tr, the server will not send any refresh message to the network members. Hence, the latter wait for the expiration of their timer before renewing the keys and update the period Tr as follows: $\mathsf{Tr} = \mathsf{Tr} + \alpha$, if the timer has not yet reached its maximum threshold T_{max}, where α is the security parameter. The attack window is thus increased. Otherwise, if one or more attacks have been detected during this period, the members receive a refresh message from the server

via the gateway node to which they are attached. This message is encrypted with the appropriate pairwise keys and contains a nonce and the number of attacks that occurred cpt. In this case, the recipient nodes do not wait for the expiration of their timer, but they immediately carry out key renewal and update the period as follows: $\mathsf{Tr} = (\mathsf{Tr} - \mathsf{cpt} * \alpha)$, if the minimum threshold T_{min} has not been reached yet. The attack window will therefore decrease. The different values taken by the timer Tr according to the number of occurred attacks, are summarized in Eq. 5 and Eq. 6 below, and also shown in Fig. 4.

- Absence of attacks ($\mathsf{cpt} = 0$)

$$\mathsf{Tr} = \begin{cases} \mathsf{Tr} + \alpha & \text{if} \quad \mathsf{Tr} \leqslant \mathsf{T}_{max} - \alpha \\ \mathsf{T}_{max} & \text{otherwise} \end{cases} \tag{5}$$

- Presence of attacks ($\mathsf{cpt} \neq 0$)

$$\mathsf{Tr} = \begin{cases} \mathsf{Tr} - (\mathsf{cpt} * \alpha) & \text{if} \quad \mathsf{Tr} \geqslant \mathsf{T}_{min} + (\mathsf{cpt} * \alpha) \\ \mathsf{T}_{min} & \text{otherwise} \end{cases} \tag{6}$$

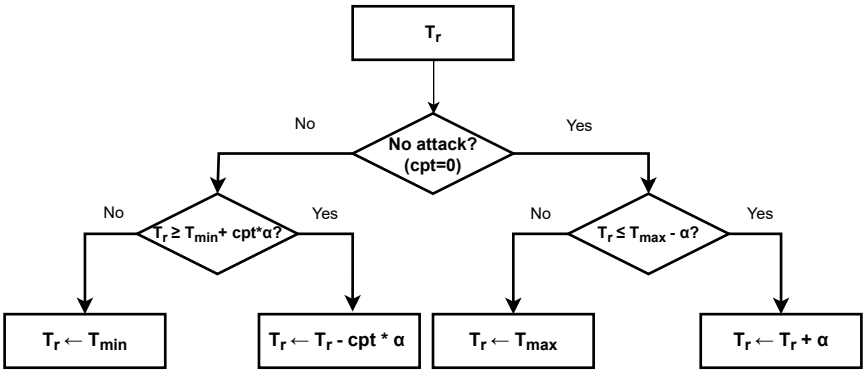


Fig. 4 Refresh period updates

4.3.4. Node addition phase

When a node intends to join the network or participate in its operations, it must go through the server. First of all, the server pre-loads current key materials into the memory of that new node before its deployment in the

network. The pre-loaded parameters include: the minimum T_{min} and maximum T_{max} thresholds of the timer T_r, hash function $h()$, security parameter α and the current one-one function of the gateway node f_g to which the new node will be attached. Moreover, the server generates and assigns to this node a unique identifier n, a one-one function f_n with its corresponding inverse f_n^{-1}. As soon as the deployment is established, the new node generates a random number r_n, then broadcasts a *join* message. This latter contains the information hidden with the one-one function of the gateway and a digest so that the other entities could not read its content.

$$Node_n \rightarrow * : Join, n, f_g(n \| f_n \| r_n), h(n \| f_n \| r_n) \tag{7}$$

Upon receiving the previous message, the gateway proceeds to check whether or not it is authentic. In the case of successful verification, the gateway in turn broadcasts the following message inviting the current members to deal with this addition. In this time, the content of the message is masked with the inverse function f_g^{-1} of the gateways node.

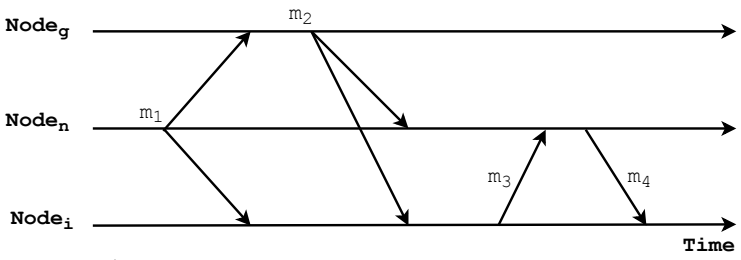
$$Node_g \rightarrow * : g, f_g^{-1}(g \| (n \| f_n \| r_n)), h(g \| (n \| f_n \| r_n)) \tag{8}$$

A current member that has received a join message waits for the reception of another message that would stem from its gateway. If the member does not receive the second message, that means the new node fails its authentication with the gateway, so the *join* message will be ignored. Otherwise, the member checks the authenticity and integrity of the last message. If the verification succeeds, the member will send to the new node an acknowledgment *Ack* message containing essentially its one-one function and the current timer that are hidden with the one-one function of the new node.

$$Node_i \rightarrow Node_n : Ack, i, f_n(i \| f_i \| r_i \| r_n + 1 \| T_r), h(i \| f_i \| r_i \| r_n + 1 \| T_r) \tag{9}$$

When receiving an *Ack* message, the new node uses its inverse and the hash functions to check it. After that, it updates its neighbors vector v_n by adding the identifier i and saves the timer T_r as well as the 2-tuple (i, f_i), with $i \in v_n$, when the verification is succeed. Accordingly, both the member and new node use Eq. 3 to compute their shared pairwise key that will be used when they want to communicate securely. Finally, the new node closes the addition process by sending the last message encrypted with the established pairwise key. The node addition process is illustrated in the Fig. 5 as follows:

$$\mathrm{Node_n} \rightarrow \mathrm{Node_i} : \{r_n \| r_i + 1\} K_{in} \tag{10}$$



m_1 : Join, n, f_g(n||f_n||r_n), h(n||f_n||r_n)

m_2 : g, f_g^{-1}(g||(n||f_n||r_n)), h(g||(n||f_n||r_n))

m_3 : Ack,i,f_n((i||f_i||r_i)||r_n+1||T_{r}),h((i||f_i||r_i)||r_n+1||T_{r})

m_4 : {r_n||r_i+1}K_{in}

Fig. 5 Node addition phase

Note that the pool of one-one functions and their corresponding inverse functions are populated with new functions after each node addition.

4.3.5. Key revocation phase

The key revocation phase is carried out according to two cases: when a node leaves the network with its willingness or does not correctly interact with other members or send any message for a long time. That node is automatically excluded from the network. Therefore, its neighbors delete the pairwise keys and security parameters, such as the one-one function shared with that node, so it will not succeed when trying to communicate with them later. That way, the current members do not save obsolete information about nodes having already left the network. Consequently, this allows them to gain some precious memory space. Note that the size of both pools of invertible and their corresponding inverse functions are reduced after each node revocation.

5. Security analysis

In this section, we discuss our scheme's formal and informal analysis. The formal analysis is performed using BAN logic.

5.1. Formal analysis

Formal verification is required to analyze and prove the correctness of the proposed scheme. For this purpose, we use Burrows Abadi Needham (BAN) logic [19] to check whether the proposed key establishment protocol works correctly and does not have redundancies or security flaws. We mainly evaluate secrecy, authentication, and freshness aspects.

a) *Notation*

The symbols P and Q denote principals, X and Y range over statements, and K represents the encryption key. A comma denotes the conjunction. The rest of the notations used in this analysis is given in Table 2.

Notation	Description
$\mathsf{P} \models \mathsf{X}$	P *believes* X
$\mathsf{P} \triangleleft \mathsf{X}$	P *sees* X or P has received a message containing X
$\mathsf{P} \vdash \mathsf{X}$	P *once said* X or P has sent a message including X
$\mathsf{P} \overset{\mathsf{K}}{\leftrightarrow} \mathsf{Q}$	K is the shared key known only to P and Q
$\mathsf{P} \Rightarrow \mathsf{X}$	P *controls* or *has jurisdiction* over X
$\#(\mathsf{X})$	Formula X is *fresh*
(X, Y)	*Concatenation* of X and Y
$\{\mathsf{X}\}_{\mathsf{K}}$	X is encrypted under the key K

Table 2 Notations

b) *Inference rules*

The main logical postulates or rules used in our analysis are listed in the following.

(1) The *selection*:

- R-Select: if a principal P sees a formula, then he also sees its components.

$$\mathsf{R}_{11} : \frac{\mathsf{P} \triangleleft (\mathsf{X}, \mathsf{Y})}{\mathsf{P} \triangleleft \mathsf{X}}; \frac{\mathsf{P} \triangleleft (\mathsf{X}, \mathsf{Y})}{\mathsf{P} \triangleleft \mathsf{Y}} \tag{11}$$

- BS-Select: If P believes that Q said (X,Y), then P believes that Q said X.

$$\mathsf{R}_{12} : \frac{\mathsf{P} \models \mathsf{Q} \vdash (\mathsf{X}, \mathsf{Y}))}{\mathsf{P} \models \mathsf{Q} \vdash \mathsf{X}} \tag{12}$$

(2) From the *message meaning* rule for shared keys, we can deduce the following rule for shared functions. That is, if P believes that the function F is a public function of Q and P sees or receives a message X hidden with

the corresponding private function F^{-1}, then P believes that Q once said X.

$$\mathsf{R2}: \frac{\mathsf{P} \models \overset{\mathsf{F}}{\leftrightarrow} \mathsf{Q}, \mathsf{P} \lhd \mathsf{F}^{-1}(\mathsf{X})}{\mathsf{P} \models \mathsf{Q} \vdash \mathsf{X}} \tag{13}$$

(3) The *nonce verification* or *freshness*: if P believes on the freshness of the statement X and P also believes that Q once said X, then P believes Q believes the statement X.

$$\mathsf{R3}: \frac{\mathsf{P} \models \#(\mathsf{X}) \, , \mathsf{P} \models \mathsf{Q} \vdash \mathsf{X}}{\mathsf{P} \models \mathsf{Q} \models \mathsf{X}} \tag{14}$$

(4) The *jurisdiction*: if P believes that Q has jurisdiction over the statement X and P believes that Q believes the statement X, then P believes the statement X.

$$\mathsf{R4}: \frac{\mathsf{P} \models \mathsf{Q} \Rightarrow \mathsf{X} \, , \mathsf{P} \models \mathsf{Q} \models \mathsf{X}}{\mathsf{P} \models \mathsf{X}} \tag{15}$$

(5) The *fresh inject*: If P believes that some part X of a formula is fresh, then the whole formula (X,Y) must also be fresh.

$$\mathsf{R5}: \frac{\mathsf{P} \models \#(\mathsf{X})}{\mathsf{P} \models \#(\mathsf{X}, \mathsf{Y})} \tag{16}$$

(6) The *belief*: If a principal P believes part X of a statement and the remaining part Y of the same statement, he believes the whole statement (X.Y).

$$\mathsf{R6}: \frac{\mathsf{P} \models \mathsf{X}, \mathsf{P} \models \mathsf{Y}}{\mathsf{P} \models (\mathsf{X}, \mathsf{Y})} \tag{17}$$

c) Analysis

Security analysis using BAN logic goes mainly through four steps: (1) idealization of the protocol to be analyzed, (2) definition of security goals to be achieved, (3) formulation of initial assumptions, and (4) the analysis or the verification of the protocol itself.

1) *Idealization*

The idealized form of our key establishment protocol is presented as follows:

$\mathit{Message}\quad 1.\quad \mathsf{A} -> \mathsf{B} : \mathsf{F}_{\mathsf{a}}^{-1}(\mathsf{ra}), \mathsf{H}(\mathsf{ra})$

$Message \quad 2. \quad B- > A : F_b^{-1}(rb, ra), H(rb, ra)$
$Message \quad 3. \quad A- > B : \left\{ rb, A \overset{K_{ab}}{\leftrightarrow} B \right\} K_{ab}$

From the previously exchanged messages, we get the following idealized forms expressed differently:

I1: $B \lhd (F_a^{-1}(ra), H(ra))$
I2: $A \lhd (F_b^{-1}(rb, ra), H(rb, ra))$
I3: $B \lhd (\left\{ rb, A \overset{K_{ab}}{\leftrightarrow} B \right\} K_{ab})$

2) *Goals*
The main goal of the proposed key establishment protocol will be achieved when two principals A and B generate a pairwise key K_{ab}, which must be authentic and known only to them. This results in the following statements:

G1: $B \models A \overset{K_{ab}}{\leftrightarrow} B$; G2: $A \models B \overset{K_{ab}}{\leftrightarrow} A$; G3: $B \models A \models B \overset{K_{ab}}{\leftrightarrow} A$

3) *Assumptions*
To analyze this protocol, we first give the following assumptions, which state the initial beliefs of the principals at the beginning of the protocol.
Initially, each device must trust the one-one and hash functions pre-shared with each of its neighbors, which results in the following assumptions:

A1: $A \models A \overset{F_a}{\leftrightarrow} A$, A2: $B \models A \overset{F_a}{\leftrightarrow} A$, A3: $A \models A \overset{F_b}{\leftrightarrow} B$
A4: $B \models B \overset{F_b}{\leftrightarrow} B$, A5: $A \models A \overset{H}{\leftrightarrow} B$, A6: $B \models A \overset{H}{\leftrightarrow} B$

In addition, each principal also believes in the integrity, authenticity, and freshness of the generated secret value, which leads to the following formulas:

A7: $A \models ra$, A8: $A \models \#(ra)$, A9: $B \models rb$, A10: $B \models \#(rb)$, A11: $B \models \#(ra)$
Finally, every principal knows that its neighbor controls or has jurisdiction over the secret value that it generates, which results in the following:

A13: $A \models B \Rightarrow rb$, A14: $B \models A \Rightarrow ra$

4) *Formal proof or verification*
The main steps of the proof are described as follows.

- The principal B receives the first message from A, and by applying the R-select rule R_{11} to the idealized form I1 or by breaking conjunction, we get:
F1: $\mathsf{B} \lhd \mathsf{F}_{\mathsf{a}}^{-1}(\mathsf{ra})$
Since we have the hypotheses A2 and F1, the message-meaning rule R2 for shared functions applies and yields the following:
F2: $\mathsf{B} \models \mathsf{A} \vdash \mathsf{ra}$
By using the freshness rule R3 over the assumption A11 and the previous result F2, we obtain:
F3: $\mathsf{B} \models \mathsf{A} \models \mathsf{ra}$
Since we have the assumption A14 and F3, the jurisdiction rule R4 applies and yields the following:
F4: $\mathsf{B} \models \mathsf{ra}$
From F4, A9 and R6, we get:
F5: $\mathsf{B} \models (\mathsf{ra}, \mathsf{rb})$
From A6 and F5, we obtain the first goal
G1: $\mathsf{B} \models \mathsf{H}(\mathsf{ra}, \mathsf{rb})$, with $\mathsf{ra} < \mathsf{rb}$, that means **G1:** $\mathsf{B} \models \mathsf{A} \overset{\mathsf{K}_{\mathsf{ab}}}{\leftrightarrow} \mathsf{B}$.

- The principal A receives the second message from B and by applying the R-select rule R_{11} to the idealized form I2 or by breaking conjunction, we have:
F6: $\mathsf{A} \lhd \mathsf{F}_{\mathsf{b}}^{-1}(\mathsf{rb}, \mathsf{ra})$
Since we have the hypothesis A3 and F6, the message-meaning rule R2 for shared functions applies and yields the following:
F7: $\mathsf{A} \models \mathsf{B} \vdash (\mathsf{rb}, \mathsf{ra})$.
By applying the refresh inject rule on the assumption A8, we get:
F8: $\mathsf{A} \models \#(\mathsf{rb}, \mathsf{ra})$.
We use the nonce verification rule R3 over the results F8 and F7 to get:
F9: $\mathsf{A} \models \mathsf{B} \models (\mathsf{ra}, \mathsf{rb})$.
From the received message, the principal A can deduce the following result:
F10: $\mathsf{A} \models \mathsf{B} \Rightarrow (\mathsf{ra}, \mathsf{rb})$
The jurisdiction rule R4 is applied to the results F9 and F10 to obtain:
F11: $\mathsf{A} \models (\mathsf{ra}, \mathsf{rb})$.
From A5 and F11, we obtain the second goal:
G2: $\mathsf{A} \models \mathsf{H}(\mathsf{ra}, \mathsf{rb})$, with $\mathsf{ra} < \mathsf{rb}$, that means **G2:** $\mathsf{A} \models \mathsf{B} \overset{\mathsf{K}_{\mathsf{ab}}}{\leftrightarrow} \mathsf{A}$.

- The principal B receives the third message from A, and by applying the message meaning rule for shared keys to the goal G1 and the idealized form I3, we will have:

F12: $B \models A \vdash B \overset{K_{ab}}{\leftrightarrow} A$.
On the other hand, when applying the rule R5 on A10, we obtain
F13: $B \models \#(B \overset{K_{ab}}{\leftrightarrow} A)$.
Finally, we use the nonce verification rule on the results F12 and F13 to deduce the third goal
G3: $B \models A \models B \overset{K_{ab}}{\leftrightarrow} A$.

5.2. Informal analysis

We informally discuss the resilience of the proposed scheme against some well-known attacks, such as eavesdropping, man-in-the-middle, node compromising, replay, and forward and backward secrecy attacks.

Eavesdropping attack
An intruder can eavesdrop on and intercept the relevant network traffic exchanged between legitimate participants for further analysis. During the pairwise key generation process, an attacker can obtain neither the pairwise keys nor key materials, allowing him/her to compute these secret keys. No sensitive data is transmitted in plain text, but it is always sent securely. They are all hidden or protected with both inverse and hash functions. Furthermore, after the key establishment phase, all the communications between two legitimate entities are protected by the use of appropriate established pairwise keys. The proposed scheme is therefore resilient against such attacks.

Man-in-the-middle attack
We distinguish two attack scenarios where an adversary captures and alters the content of the *request* and/or *reply* messages exchanged during the pairwise key establishment phase between two legitimate nodes.

Attack 1. When an adversary changes the identifier of the sender i to k, the recipient j will be able to detect such an attack. Two cases may arise.

- $k \in \nu j$: that means k belongs to the neighbor vector of the receiver j. In other words, k is also a neighbor of j. In this case, the recipient node j finds and uses the inverse function of k, noted f_k, instead of the function f_i of i, to calculate the image of the first part of the received message. As a result, it gets a completely different image or value. After that, when it applies the hash function to the last result, it will find a footprint different from the one emitted by the sender. Hence, the recipient immediately detects that the message has been altered.

– $k \notin \nu j$: this means that k is not in the neighborhood of the recipient node j. In this case, the receiver fails when it attempts or tries to retrieve the initial hidden information because it does not find the appropriate inverse function to be used for this purpose. As a result, this node suspects an attack and thus discards the message.

Attack 2 Whenever an intruder replaces the hidden information or the image $f_i^{-1}(x)$ with another one, let's say, $f_k^{-1}(x)$ with $k \neq i$, the recipient also detects such attack, because it obtains a hash value that will not match one of the senders. Thus, the proposed scheme is secure against the Man-in-the-middle attack.

Node capture attack
In this type of attack, when an adversary captures a node, he/she can gain access to its memory. The proposed scheme resists this kind of attack well since, on the one hand, each node holds a small number of keys, allowing it to communicate securely with its direct neighbors. On the other hand, these keys are not stored in clear text, but they are hidden or masked by using the inverse function of the node. Therefore, the compromise of a given device has no impact on the other nodes and communication links.

Replay attack
A replay attack is a security threat where the malicious entity repeats or re-injects the old valid messages intercepted during previous communication sessions into the network. The proposed scheme can withstand this security attack since transmitted messages contain random secrets. On the other hand, when an adversary replies a captured message after a waiting time equals to t, If $t > Tr$, upon the reception of that replay message, the verification does not succeed because all devices have already refreshed their pairwise keys. When $t > 2Tr$, verification fails because all inverse functions have been updated. As a result, the recipients will reject this old replayed message coming from the malicious node.

Forward and backward secrecy
Forward and backward secrecy is guaranteed because all pairwise keys are renewed after the expiration of the timer Tr. On the other hand, the old one-one functions are erased from the memories of nodes and replaced by the new ones after the timer $2Tr$ elapses. Thus, the new secret keys are independent of the old keys. Therefore, the old keys become unusable, and they can not be able to decrypt new encrypted messages. In the same way,

the new keys can not be used to decrypt old messages sent in the previous communication sessions.

6. Performance analysis

The performance of the proposed scheme is compared with the recent protocols [9] and SSEKM [12] introduced in the related work section. The evaluation is performed according to storage, communication, computation costs, and energy consumption metrics. The simulations are conducted using MATLAB environment. We consider a network where nodes are randomly deployed in a square area of 100 m^2. Nodes are equipped with wireless communication antennas with a range of 10 meters. Moreover, the energy model presented by Heinzelman et al. [20] is used to estimate the amount of dissipated energy. According to this model, when a node sends a data of k bits length over a distance d, it dissipates a quantity of energy (joule) estimated to

$$E_t(k, d) = k * (E_{elec} + E_{amp} * d^2) \tag{18}$$

whereas when it receives the same amount of data, it consumes

$$E_r(k) = E_{elec} * k \tag{19}$$

where $E_{elec} = 50 * 10^{-9}$j/bit and $E_{amp} = 100 * 10^{-12}$j/bit/m^2.
The notation used in this section is summarized in Table 3.

6.1. Storage overhead

The storage overhead can be defined as the total memory space required to store keys and security parameters. In IFKMS, each node saves a refresh period Tr along with its thresholds T_{min}, T_{max}, one 3-tuple, and as many 2-tuples as it has neighbors in its communication range. Furthermore, a node with d neighbors has to store at most d pairwise keys.
In the experiments, it is assumed that the values of l_{id}, l_f and l_k are set to 4, 16 and 16 bytes respectively. Furthermore, the value of m in SSEKM, is fixed to 50 since the probability of sharing at least one key is high when the pool contains 1000 keys.
The overall storage overhead in the function of the network size is shown in Fig. 6. From the figure, we can observe that the storage overhead increases in the compared schemes with the number of nodes in the network. However, IFKMS has less storage overhead when compared to the other protocols. When the network size is equal to 10, for instance, our scheme saves about

Notation	Description
l_{id}	Size of an identifier of a (node, key, or polynomial), refresh value, polynomial share (coefficient), secret value, vector element
l_k	Size of a key
l_t	Size of message type (Req, Rep, ACK)
l_f	Size of one-one function or polynomial
l_{of}	Size of one-one function's output
l_e	Size of an encrypted message
l_h	Size of a hash value
H	Hash operation
F	Image computation
Mod	Modulus operation
PE	Polynomial evaluation
M	Multiplication operation
E/D	Encryption/Decryption operations

Table 3 Notations

3, 83% of memory space in comparison with the protocol [9]. This gain progressively increases to reach approximately 67, 54% when the number of nodes reaches 100 in the network. That is a growth of about 63, 71%.

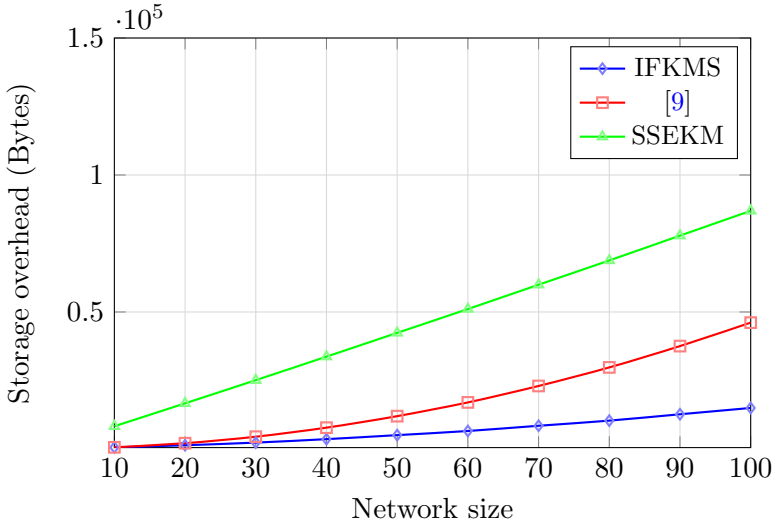


Fig. 6 Storage overhead vs. Network size (m=50)

6.2. Communication overhead

The communication cost can be measured as the total amount of data exchanged in the whole network. In our scheme, nodes do not have to communicate during the setup phase. That is why we were just interested in communication load during the key establishment phase. In this second phase, to establish pairwise keys with all adjacent members, a node must send one *request*, receive $\mathtt{d}$ *responses* and send back $\mathtt{d}$ encrypted messages, where $\mathtt{d}$ represents the number of nodes in its direct neighborhood.

The communication cost, expressed in bytes, in the function of the hash algorithm during the key establishment phase is illustrated in Fig. 7. Note that we considered a network of 100 nodes in the simulations. The figure indicates that the communication cost progressively increases in the network when increasing the hash value for the compared schemes. However, the results show that IFKMS is comparable to SSEKM and it is better than [9]. The amount of data required to be exchanged between adjacent nodes for key establishment purposes in IFKMS is less than the latter. For instance, when the hash value is equal to 16 bytes, the total quantity of information exchanged is about 18.6 Kbytes in our scheme against 128.6 Kbytes in [9]. Consequently, our scheme requires roughly 85.54% fewer data exchanged than that protocol. This gain in terms of communication progressively decreases up to 67.92% when the hash value reaches 64 bytes.

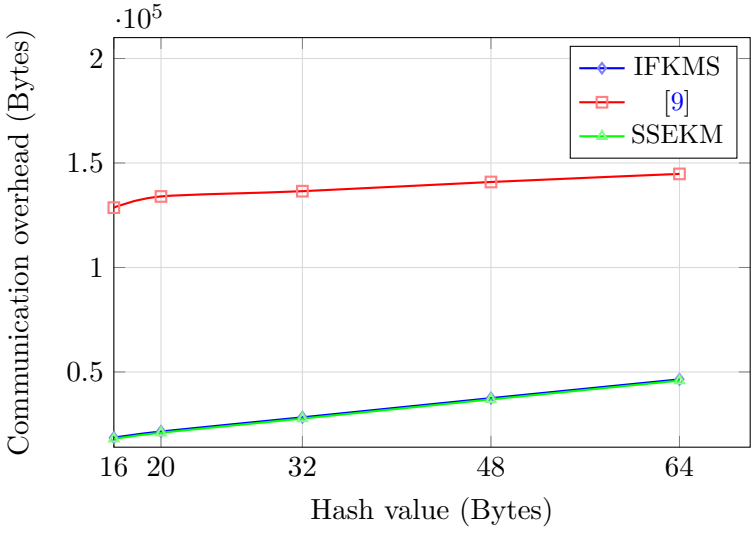


Fig. 7 Communication overhead vs. Hash algorithm (n=100 Nodes)

The impact of the network size on the communication overhead during the pairwise key establishment phase is given in Fig. 8. Note that the hash value is set to 16 bytes during simulations while the network size ranges from 10 to 100 nodes. As depicted in this figure, IFKMS considerably reduces the number of messages exchanged during this phase when compared with the scheme [9]. The figure shows that the communication overhead increases slowly in IFKMS and SSEKM, whereas this raises quickly in the other scheme. For instance, when the number of nodes reaches 100, the overall communication load in the network rises to 129.56 kbytes in [9], while it is only 18.71 kbytes in ours. That is a difference of about 85.58%.

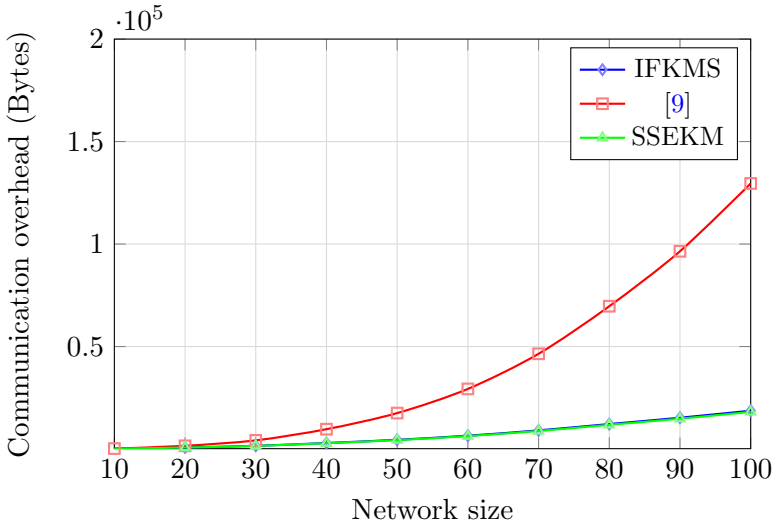


Fig. 8 Communication overhead vs. Network size (h=16 bytes)

6.3. Comparative study

A comparison between IFKMS with the related schemes introduced in Section 3 is summarized in Table 4. This comparison concerns the storage, communication, and computation overhead at the constrained node side. Notice that the evaluation of the two last requirements is performed when computing just one pairwise key between a pair of nodes. In addition, the established keys are not considered in the memory occupation estimation for all the compared schemes. For example, concerning the storage overhead, every node in our scheme needs to store $(\mathtt{d}+2)$ one-one functions, $(\mathtt{d}+1)$ nodes identifiers, and 3 refresh values. At the same time, in [9], each sensor stores a $\mathtt{t}$-degree polynomial shares, as well as a row of $\mathtt{BL}$ and a column

of BU matrices of (N/2) elements each, with N represents the size of the network. Whereas, in SSEKM [12], to each node is assigned an identifier, m keys and a non-convergent recursive sequence with its first term. For the communication overhead, to compute one pairwise key in IFKMS, a node needs to send a request with an encrypted message and receive a reply from the adjacent node. Both the request and the response contain a message type, a node identifier as well as a hidden and hash values. Meanwhile, in [9], a sensor sends its identifier, the row of BL matrix, and an encrypted identifier. At the same time, it receives an ACK, a row of BL matrix, an identifier, an encrypted message, and a MAC. While in SSEKM, one node has to send and receive an identifier, a key chain identifier as well as a nonce and hash value. Finally, in terms of computation overhead, to establish one key in IFKMS, a node needs to perform two one-one function evaluations, three hash operations, and one encryption, while in [9], a sensor carries out polynomial evaluation, vector multiplication, encryption/decryption, and MAC operations. Whereas, a node performs only two recursive function evaluations in SSEKM.

Scheme	Storage	Communication	Computation
IFKMS	$(d+4)*l_{id}+(d+2)*l_f$	$2(l_t+l_{id}+l_{of}+$ $+l_h)+l_e$	$2F+3H+E$
RKP [2]	$(k+1)*l_k+k*l_{id}$	$2k*l_{id}$ or $2k*l_e$	$k*(E+D)$
[4]	$s'(l_f+l_{id})$	$(s'+1)*l_{id}$ or $4s'*l_{id}+l_e$	PE or D
EKM [5]	l_k+l_f	$2*l_e$	$2H+E+D+PE$
NISKC [6]	$l_{id}+l_f$	l_t+l_f	PE
[7]	$s*l_k+g*l_f+$ $(3s+g+1)*l_{id}$	$2*(3s+g+1)*l_{id}$	PE or H
[9]	$N*l_{id}+l_f$	$(2+N)*l_{id}+$ $l_t+2*l_e+l_h$	$(N/2)*M+$ $E+D+H+PE$
[11]	$2*l_f+l_k$	l_e	$PE+Mod+D$
SSEKM [12]	$m*l_k+2*l_{id}+l_f$	$8*l_{id}+2*l_h$	$2*F$
H2KD [13]	$l_{id}+l_k$	$3*l_{id}+l_e$	–
[14]	$l_{id}+l_k+l_h$	$6*l_{id}+2*l_h$	$3*(XOR+H)$
[15]	l_k	$4*l_{id}+2*l_t$	$2*M$

Table 4 Overhead comparison of different schemes

6.4. Energy consumption

We evaluate the performance of IFKMS with respect to dissipated energy in the entire network when nodes exchange required messages for pairwise key computation purposes. This energy consumption is directly related to the size and the number of messages sent and received by each node in the network. This explains why the Fig. 9 is very similar to Fig. 7, and the Fig. 8 and Fig. 10 are very close. Fig. 9 shows the dissipated energy in communication in the function of the hash algorithm during the key establishment phase. The network size is set to 100 nodes as in Fig. 7. As illustrated in this figure, the energy consumption increases with increasing the hash value for the compared schemes. However, IFKMS and SSEKM

widely outperform the scheme proposed in [9]. This can be explained by the high amount of data required to be exchanged between nodes in the latter scheme compared with IFKMS and SSEKM. For instance, when the hash value is equal to 64 bytes, the total energy dissipated in IFKMS is about 19633.06µJ against nearly 60551.97µJ in [9]. As a result, our protocol saves roughly $67,58\%$ of energy compared with that scheme.

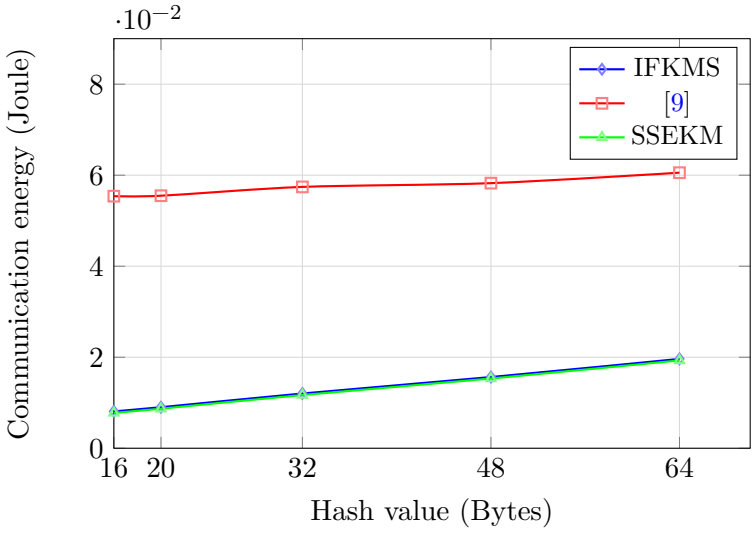


Fig. 9 Communication energy vs. Hash algorithm (n=100 Nodes)

Fig. 10 plots the impact of the network size on the communication energy during pairwise key establishment. Note that during simulations, we have kept the same values of the hash and the network size as those in Fig. 8. As shown in this figure, IFKMS and SSEKM considerably reduce the energy consumption during this phase when compared to the scheme [9]. We can observe that when the number of nodes reaches approximately 40 nodes in the network, the dissipated energy starts to rise rapidly in [9]. At the same time, it remains stable and increases slowly in ours. For example, whenever the network size reaches 100 nodes, the energy consumed in the entire network is 7951.6µJ in IFKMS against about 54352.12µJ in [9]. As a result, our scheme saves up $46400,52$µJ of energy.

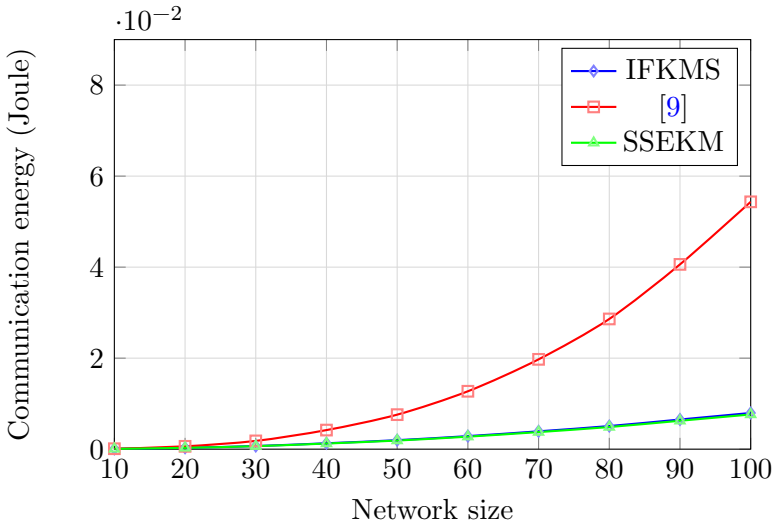


Fig. 10 Communication energy vs. Network size (h=16 bytes)

7. Conclusion

IoT networks are increasingly used in various applications such as Medical Internet of Things. Securing communications in such systems is challenging due to severe resource constraints and heterogeneity of IoT devices. In this paper, we proposed a new key management scheme that exploits the features of inverse functions to protect communications between devices. Therefore, sensitive data is hidden during message exchanges by employing one-one functions to defeat security attacks. Moreover, a variable re-keying period is used, making our scheme more adaptable and resilient to attacks. The security and performance analysis showed that IFKSM meets basic security goals including secrecy, integrity and authentication, and ensures resilience since keys are refreshed frequently in the presence of attacks in the system. It is also efficient in terms of energy consumption as well as memory, communication, and computation overheads when compared with most studied protocols, and it has approximately the same performance as other ones.

As further work, we intend to extend the proposed scheme by integrating the group key establishment mechanism to deal with group communication, and implement this solution in a real application of IoT environment.

Acknowledgments

This work has been carried out under the research activities of CEDRIC laboratory at CNAM, Paris, France, and jointly with LIMED laboratory, which is affiliated to the Faculty of Exact Sciences, University of Bejaia, Algeria. It has been sponsored by the General Directorate for Scientific Research and Technological Development Program of the Ministry of Higher Education and Scientific Research (DGRSDT), Algeria.

References

[1] Z. Bi, L. Da Xu, C. Wang, Internet of things for enterprise systems of modern manufacturing, IEEE Transactions on industrial informatics 10 (2014) 1537–1546. doi:10.1109/TII.2014.2300338.

[2] L. Eschenauer, V. D. Gligor, A key-management scheme for distributed sensor networks, in: Proceedings of the 9th ACM conference on Computer and communications security, ACM, 2002, pp. 41–47. doi:https://doi.org/10.1145/586110.586117.

[3] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, M. Yung, Perfectly secure key distribution for dynamic conferences, Information and Computation 146 (1998) 1–23. doi:https://doi.org/10.1006/inco.1998.2717.

[4] D. Liu, P. Ning, R. Li, Establishing pairwise keys in distributed sensor networks, ACM Transactions on Information and System Security (TISSEC) 8 (2005) 41–77. doi:https://doi.org/10.1145/1053283.1053287.

[5] Z. Mahmood, H. Ning, A. Ghafoor, A polynomial subset-based efficient multi-party key management system for lightweight device networks, Sensors 17 (2017) 670. doi:https://doi.org/10.3390/s17040670.

[6] V. Kumar, R. Kumar, S. Pandey, Polynomial based non-interactive session key computation protocol for secure communication in dynamic groups, International Journal of Information Technology 12 (2020) 283–288. doi:https://doi.org/10.1007/s41870-018-0140-1.

[7] J. Zhang, H. Li, J. Li, Key establishment scheme for wireless sensor networks based on polynomial and random key predistribution scheme, Ad Hoc Networks 71 (2018) 68–77. doi:https://doi.org/10.1016/j.adhoc.2017.12.006.

[8] K. Lu, Y. Qian, M. Guizani, H.-H. Chen, A framework for a distributed key management scheme in heterogeneous wireless sensor networks, IEEE transactions on wireless communications 7 (2008) 639–647. doi:10.1109/TWC.2008.060603.

[9] A. M. Manasrah, A. R. AL-Rabadi, N. A. Kofahi, Key pre-distribution approach using block lu decomposition in wireless sensor network, International Journal of Information Security (2019) 1–18. doi:https://doi.org/10.1007/s10207-019-00477-4.

[10] J. Chen, K. Ji, Z. Shi, W. Liu, Implementation of block algorithm for lu factorization, in: 2009 WRI World Congress on Computer Science and Information Engineering, volume 2, IEEE, 2009, pp. 569–573. doi:10.1109/CSIE.2009.814.

[11] A. Albakri, L. Harn, S. Song, Hierarchical key management scheme with probabilistic security in a wireless sensor network (wsn), Security and Communication Networks 2019 (2019). doi:https://doi.org/10.1155/2019/3950129.

[12] V. Kumar, N. Malik, G. Dhiman, T. K. Lohani, Scalable and storage efficient dynamic key management scheme for wireless sensor network, Wireless Communications and Mobile Computing 2021 (2021). doi:https://doi.org/10.1155/2021/5512879.

[13] O. AbuAlghanam, M. Qatawneh, W. Almobaideen, M. Saadeh, A new hierarchical architecture and protocol for key distribution in the context of iot-based smart cities, Journal of Information Security and Applications 67 (2022) 103173. doi:https://doi.org/10.1016/j.jisa.2022.103173.

[14] L. Bai, C. Hsu, L. Harn, J. Cui, Z. Zhao, A practical lightweight anonymous authentication and key establishment scheme for resource-asymmetric smart environments, IEEE Transactions on Dependable and Secure Computing (2022). doi:DOI:10.1109/TDSC.2022.3203874.

[15] S. M. Hussein, J. A. López Ramos, A. M. Ashir, A secure and efficient method to protect communications and energy consumption in

iot wireless sensor networks, Electronics 11 (2022) 2721. doi:https://doi.org/10.3390/electronics11172721.

[16] W. B. Heinzelman, A. P. Chandrakasan, H. Balakrishnan, An application-specific protocol architecture for wireless microsensor networks, IEEE Transactions on wireless communications 1 (2002) 660–670. doi:DOI:10.1109/TWC.2002.804190.

[17] M. Nafi, S. Bouzefrane, M. Omar, Matrix-based key management scheme for iot networks, Ad Hoc Networks 97 (2020) 102003. doi:https://doi.org/10.1016/j.adhoc.2019.102003.

[18] D. Dolev, A. Yao, On the security of public key protocols, IEEE Transaction on information theory 29 (1983) 198–208. doi:10.1109/TIT.1983.1056650.

[19] M. Burrows, M. Abadi, R. M. Needham, A logic of authentication, Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences 426 (1989) 233–271. doi:https://doi.org/10.1098/rspa.1989.0125.

[20] W. R. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, in: Proceedings of the 33rd annual Hawaii international conference on system sciences, IEEE, 2000, pp. 10–pp. doi:10.1109/HICSS.2000.926982.