μ KMS: Micro Key Management System for WSNs

Sofiane Aissani, Mawloud Omar, Abdelkamel Tari, Feriel Bouakkaz

► To cite this version:

Sofiane Aissani, Mawloud Omar, Abdelkamel Tari, Feriel Bouakkaz. μ KMS: Micro Key Management System for WSNs. IET Wireless Sensor Systems, 2018. hal-03034606

HAL Id: hal-03034606

<https://hal.science/hal-03034606>

Submitted on 1 Dec 2020

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

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

μ KMS: Micro Key Management System for WSNs

Sofiane Aissani, Mawloud Omar, Abdelkamel Tari, and Feriel Bouakkaz

*^aLaboratoire d'Informatique Médicale, Faculté des Sciences Exactes
Université de Bejaia, 06000 Bejaia, Algérie.*

Abstract

Key management is the basic building block of all the security protocols, and is one of the most challenging issues in wireless sensor networks (WSNs). Centralizing a trusted key management server is not an appropriate solution in such fully distributed networks. In other hand, designing a distributed key management system is a challenging task, due to the constrained characteristics of sensor nodes, which are limited in storage, computation, communication and energy. In the literature, there is a hot research effort on key management purpose for WSNs. The greatest part of the existing solutions focus mainly on the optimization of key number, rekeying frequency and process or the encryption system of the distributed keys. Unfortunately, these systems are implemented as an additional and independent service, involving a considerable overhead. In this paper, we propose μ KMS (Micro Key Management System) for WSNs. μ KMS implements a dissimulation scheme, embedding the rekeying process messages on the unexploited coding space of the exchanged ZigBee packets. We have developed simulations, where the obtained results show the relevance of μ KMS in terms of communication overhead, storage overhead and energy consumption.

Keywords: WSN, Security, Key management, Dissimulation, Energy, μ KMS

1. Introduction

Wireless sensor network (WSN) [1][2] is a promising technology for collecting and gathering information in order to monitor a specific area. Their low-cost and ease of deployment make them the main attractive solution for a plethora of applications in various fields, such as military tracking, fire monitoring, etc. They consist of short range sensing devices that collaborate

to carry out monitoring measurements to the end users. Sensor nodes are characterized by some intrinsic properties representing important design factors, such as energy constraints, limited computation and storage capacities. In addition, many applications require deploying sensors in harsh environments and in large quantities, making very difficult for the manual control of sensors [11].

As in the case of wired and mobile networks, WSN applications should address some important security requirements, as authentication, integrity, confidentiality, etc. Furthermore, even with the limited resource constraint in such network, it should be able resisting to attacks, like eavesdropping, usurpation, fabrication, alteration, etc. Unfortunately, due to its constrained characteristics, classical approaches, such as secure routing and asymmetrical-based encryption are impractical. WSNs require specific solutions, which must be lightweight while providing the security services. Key management is the basic building block of all the security services and is one of the most challenging issues in WSNs.

Historically, several schemes have been proposed for use in WSNs. Although it is hard to accurately determine the number of papers published every year, the search for key terms such as "key management" or "key establishment" in a scholarly source (e.g., Google Scholar) shows a quick evolution on this number: while a few dozens results appear in the period of 2002 and 2003, hundreds are returned in 2007, 2008, 2009 [12], and a hundred thousand in 2016. The main objective of a key management system is the establishment and management of cryptographic-keys in order to provide and maintain secure communications among the network entities. In WSNs, the key management is a challenging issue. Centralizing a key management server, for example on the base station is an impractical solution. The main problem with this approach is that the central server becomes a target of attacks. Nonetheless, when such a server is available and secure, these approaches may become very attractive. In other hand, designing a distributed scheme may involve additional overheads on sensor nodes.

In the literature, several solutions are proposed to address the key management issue in WSNs. Although the availability of various solutions, there still exist limitations. The greatest part of the existing solutions focus only on the optimization of key number, rekeying frequency and process or the encryption system of the distributed keys. Unfortunately, these systems are implemented as an additional and independent service, involving a significant overhead. In this paper, we propose μ KMS (Micro Key Management

System). μ KMS is designed to respond the following requirements: (1) high probability that the base station and the sensor nodes be able to communicate securely when transmitting events data; (2) high robustness against attacks; (3) low storage of keys; and (4) low commutation overhead in the rekeying process. The contribution of this work is double:

1. In the security point of view, μ KMS is robust against the replay attack, it guarantees the private-key exchange integrity even if an attacker modifies the content of packets in transmission, and it preserves the private-key secrecy against both physical and logical attacks.
2. In the efficiency point of view, μ KMS introduces a new form of key exchange through a dissimulation mechanism. μ KMS is an overhead free system, which embeds the rekeying process messages on the unexploited coding space of the exchanged ZigBee packets. As we illustrate in Figure 1, we design μ KMS as a security layer below the ZigBee-based network layer. μ KMS configures each sensor node by one individual private-key with which it guarantees the secrecy of data dissemination. The private-key is periodically updated in an efficient manner without any overhead.

The services ensured by the ZigBee security-layer focus on data confidentiality and integrity through the cryptosystems AES (Advanced standard Encryption) and MIC (Message Integrity Code), respectively. The system μ KMS focuses on key management, which is the basic brick of the security services mentioned above. It allows to generate, distribute and revoke the cryptographic keys used by the confidentiality and the integrity security services. That's why the position of μ KMS is under the security-layer. In the

Application
Security
Network
μKMS
MAC
PHY

Figure 1: μ KSM location in the ZigBee layers

other hand, μ KMS is designed under the network-layer in order to reconstitute correctly the exploited frame fields (e.g., sequence number, radius, etc.) before they can be used by the network-layer protocols.

The remaining of the paper is structured as follows. In Section 2, we present some relevant and recent related works. In Section 3, we give the detailed description of μ KSM and in Section 4, we analyze its security. In Section 5, we analyze through simulations, the performances of μ KSM with respect of storage, communication and energy consumption. Finally, we conclude the paper in Section 6.

2. Related work

There are hundreds consistent works which address the key management problem in WSNs. The authors of [3], [8] and [12] summarize a representative part of them. In this section, we review some relevant and recent schemes. We classify them into two categories, namely, the key communication-aware protocols and the key storage-aware protocols.

2.1. Key communication-aware protocols

In this category, the criterion of communication overhead reducing is privileged. The protocols operate in such a way to minimize as well as possible the online key exchange with the aim to downgrade the communication overhead, and consequently, the energy consumption of the sensor nodes.

In [7], Ruj et al. have proposed pairwise and triple key establishment schemes in mobile hierarchical WSNs. For the pairwise establishment scheme, authors propose a strong Steiner tree design and mapped it to q -composite key pre-distribution scheme with $q = 2$. The result is a model where any set of 2 keys occurs either in two sensor nodes or none. If two sensor nodes wish to communicate, they broadcast their identifiers. According to the latter, they may conclude that if they have a common pair of keys. The pairwise-key is calculated by hashing the common pair of keys concatenated to their identities. Based on the same reasoning, the authors have proposed a triple key distribution to establish a common key between three parties. In this context, a set of 2 keys occurs either in three sensors or none. The common key which will be shared among the sensor nodes is computed by hashing the common pair of keys concatenated to the identities of the three sensor nodes. They have also described how to establish simultaneously pairwise

and triple keys, ensuring that even having a common key K_{ABC} , the sensor node C is not able to compute K_{AB} .

In [13], Zhang et al. have proposed a group-key management scheme for hierarchical WSNs based on threshold cryptography. In the key pre-distribution phase, the base station fixes the total number of groups. For each group, a unique $2t$ degree bivariate polynomial is constructed over a prime finite field. Accordingly, each sensor node obtains a personal secret through the generated polynomial. After the network deployment, the cluster-head distributes to the sensor node members the hierarchical keys calculated using a one-way hash function based on their identifiers. The cluster-head collects t personal secrets of members to derive the group-key. The cluster-head adds its part and interpolates the $t+1$ points on the same polynomial following the threshold secret sharing scheme proposed in [20]. If a cluster-head leaves the group or is captured by attackers, a new cluster-head will be elected and the previous steps are executed again. On the other hand, if a sensor node member leaves the group, the cluster-head deletes its hierarchical key and launch the group-key update. The authors have also proposed a solution for the node isolation problem, which occurs when several sensor nodes in the same cluster are compromised. In this case, the cluster-head sends to the other cluster-heads a joining message, including the identities of legitimate sensor nodes in the cluster. Then, the new cluster-head will establish hierarchical keys with the new sensor nodes and allocates to them the new group-key.

In [16], Chan et al. have proposed solutions for large peer-to-peer WSNs with random key pre-distribution. They have proposed a q -composite random key pre-distribution scheme, where any two neighboring sensor nodes need to find q common keys from their key rings to establish a secure link. After the key discovery phase, each sensor node identifies its neighbors with which it shares at least q keys. Then, through the latter keys, a new communication link key is generated. The choice of q depends on the key pool size and the required probability of successfully performing key-setup with some neighbor. They have also proposed a multi-path key reinforcement to enhance the security of an established communication link key. The established key between two sensor nodes may be residing in some other nodes. Hence, it is necessary to update the key under the condition that don't use the direct link between the two sensor nodes. The proposed approach consists of identifying k independent secure paths created during the initial key-setup and send k random values via them. They have also proposed an improved random-pairwise key scheme. It is based on the observation that not all the

keys need to be stored in the sensor node key ring to have a connected random graph with high probability. Each sensor node needs to store a random set of pairwise-keys chosen through a specific probability.

2.2. Key storage-aware protocols

In this category, the criterion of storage overhead reducing is favored. This aspect is very important due to the limited capacity of sensor nodes in terms of memory. The protocols operate in such a way to minimize as well as possible the preloaded and predefined keys in order to preserve the storage consumption.

In [4], Seo et al. have proposed a certificate-less key management scheme for dynamic hierarchical WSNs. Before the network deployment, the base station establishes a pair of public and private keys for each sensor node. An individual sensor node key is shared with the base station by a Hash-based Message Authentication Code (HMAC). After the network deployment, each sensor node establishes pairwise-keys with its neighbors. A pairwise master-key is established and their respective certificate-less public/private key pairs. In the next step, the cluster-head forms the cluster with its authenticated neighboring sensor nodes and asks the base station to their validation. A cluster-key is shared by all the sensor node members and is used to secure the broadcasted messages. The cluster-key is generated by the cluster-head using HMAC (on a secret parameter and the identifier of the cluster-head) and exchanged using the pairwise-keys. The keys update occurs when a sensor node moves between clusters, when a sensor node is compromised or when a sensor node joins the network. For pairwise-key update, it is not necessary to change the pairwise master-keys, just perform again the pairwise key establishment. For the cluster-key update, the cluster-head chooses a new secret parameter and computes again the HMAC.

In [5], Suganthi and Vembu have proposed an algorithm allowing the establishment of three types of keys for each sensor node without key broadcasting and a minimal base station involvement. The sensor node individual key is used for initial communication with the base station and is calculated through system parameters distributed before the network deployment. The system parameters include a shared pseudo random function, an initial key and the individual sensor node identifier. A pairwise-key is used to secure the communication between two neighbors and a group-key is shared by all the network sensor nodes. These keys are calculated dynamically using a polynomial function among a set of preconfigured functions in the sensor nodes.

The pairwise-keys and the group-key are periodically updated by changing the coefficient of the polynomial function.

In [10], Du et al. have proposed a key management scheme for hierarchical WSNs based on the congruence property of modular arithmetic. The feature of this protocol is that the sensor node only needs a key seed to compute the shared key with its cluster and the group-key. During the cluster training, a mutual authentication between the cluster-head and its sensors is performed via pre-distributed keys. The next phase is the key generation, where the cluster-head sends a key seed for each sensor node. Then, the cluster-head shares with each sensor node a common key computed by hashing the key seed. The sensor nodes of the same cluster share a group-key computed through the modular arithmetic over the key seeds. Due to the congruence property of the latter, the group-key is the same at each sensor node even if the key seeds are different. In the case of a new or a compromised sensor node, the key seed will be updated through a distributed protocol. The authors have proposed also an intra-cluster and inter-cluster key establishment scheme if two simple sensor nodes need to communicate securely.

2.3. Overall discussion and objectives

Regarding the reviewed solutions, we notice the presence of a close correlation between the criteria of key storage and key communication overheads. The communication overhead appears when the sensor nodes try to establish common keys among themselves. This process is inevitable in the case of a limited key pool, which is, however, gainful in terms of storage load. In the other hand, when a sensor node is given a sufficient key pool, it will not be constrained later to establish specific keys with its interlocutors, and hence, gain in terms of communication load. To solve these issues, the reviewed solutions focus mainly on the optimization of key number, rekeying frequency or the encryption system of the distributed keys.

The main goals of our proposal are the following. First, the design of a robust key management solution, which guarantees the following security proprieties: (1) private-key exchange integrity, (2) replay attack resistance, private-key secrecy, (3) Sybil attack resistance, (4) node replication attack resistance, (5) selecting forwarding attack resistance, and (6) past secrecy. Second, addressing the trade-off between the key commutation and storage overheads. Differing to the literature, where the key management is implemented as an additional and independent service, a key dissimulation mechanism should be designed. This mechanism allows to embed the exchanged

keys on the unexploited coding space of the packets, and hence, solving the issue of communication overhead. To address the storage issue, the key establishment should be managed by the base station. In each sensor node could be kept only one common key shared with the base station, which will be updated by several fragments through the dissimulation mechanism.

3. μ KMS: Micro Key Management System

In this section, we present the detailed description of μ KMS characteristics and operations.

3.1. Network model and assumptions

We consider a network composed of a set of η wireless sensor nodes deployed on a zone of interest, supervised by a single base station. The network follows a flat architecture, in which no specific functions are assigned to specific sensor nodes. We assume that the sensor nodes are homogeneous regarding the hardware characteristics, such as storage, battery power, sensing, processing, and communication capacities. Sensor nodes can be mobile and there are no assumptions regarding the reliability of wireless mediums in terms of transmission. We assume that the base station is sufficiently secured and has no constraints on storage and computation. Each sensor node ϑ_i has a unique identifier, and is preconfigured with a unique private-key, denoted by $K_{i,0}$, shared with the base station. The rekeying operation is performed by the nodes themselves, which is executed at each period of time ΔT . The latter parameter denotes the required amount of time at each rekeying round. At each round R_j , a sensor node ϑ_i generates a new private-key $K_{i,j}$, and erases its current private-key $K_{i,j-1}$ with the new one after the reception of the acknowledgment from the base station. Indeed, the parameter ΔT influences highly the rekeying frequency, and must be adjusted relating to the targeted application sensitivity. A trade-off between the required security level and the performances should be taken in consideration. A low value of ΔT offers a good resilience of the system against passive attacks, however, with an overload in terms of computation, and vice-versa. Table ?? summarizes the used notations.

We suppose that the system doesn't include compromised sensor nodes at the initialization step and the attacks are performed after the network deployment. An attacker could be static or mobile and could perform active or passive attacks by considering internal and external attacks. An attacker

	Notation	Description
	η	Network size
	κ	Private-key size
	$(M)_K$	Message M encrypted with the key K
	ϑ_i	Sensor node i identifier
	ΔT	Necessary period of time to perform the rekeying process
	RM_i	Rekeying message sent by ϑ_i
Notations	R_j	j^{th} rekeying round
	$K_{i,0}$	Preloaded ϑ_i private-key
	$K_{i,j}$	ϑ_i private-key at the round R_j
	n	Number of parts to constitute a private-key
	$\mathbb{P}_{S,D}\langle M \rangle$	Message M dissimulated over a packet sent by S to D
	$\mathbb{T}$	Table of identifiers and keys correspondance
	$\mathbb{T}(\vartheta_i)$	Function which outputs the private-key holden by ϑ_i
	$\mathbb{T}(K_i)$	Function which outputs the identifier of node holding K_i

could perform an external attack by compromising a sensor node and/or extracting its secret cryptographic parameters. An internal attacker could decrypt and/or modify the encrypted messages with the private-keys that it holds. An internal attacker could also drop messages and overhear the communication between sensor nodes in its range.

The proposed approach is quite adapted for specific potential applications, where a permanent connectivity between the base station and the sensor nodes is highly necessitated. This requirement is imposed by several domains, such as remote medical systems, real-time video transmissions, industrial monitoring applications, intruder tracking, as well by many other applications. In such systems, is extremely important that the sensor nodes communicate the sensed data systematically in real-time to the base station. This aspect doesn't present a hard assumption since it is present in quite a lot of application types. The permanent connectivity between the base station and the sensor nodes represents the main characteristic over which μKMS operates. It exploits the fact that, in all the cases, the sensed data will be transmitted to the base station to dissimulate information about the sensor node keys. Furthermore, μKMS exploits the hardware capacity of the base station in key management operations.

3.2. Initialization of μKMS

This phase is executed in offline before the network deployment. The sensor nodes are deployed after the initial private-key pre-distribution. Each sensor node ϑ_i is pre-configured with a private-key $K_{i,0}$. The latter represents the unique private-key maintained by the sensor node, which is shared exclusively with the base station and is used for both authentication and data confidentiality protection. Later, this key will be a subject of rekeying in order to protect its secrecy against passive attacks. Moreover, in order to protect its secrecy against physical attacks, this key is stored in the sensor node volatile memory, independently to the used hardware technology. With this manner, even if a sensor node is physically captured, it would be hard for the attacker keeping the private-key availability when that node is out of energy source. The base station maintains locally a table, denoted by $\mathbb{T}$, keeping the mapping of sensor node identifiers to their corresponding private-keys. The base station has an unbuilt algorithm which takes a sensor node identifier as input and outputs its corresponding private-key and vice-versa. The table $\mathbb{T}$ can be interrogated over two specific operations, namely $\mathbb{T}(K_i)$ and $\mathbb{T}(\vartheta_i)$. The function $\mathbb{T}(K_i)$ outputs the sensor node identifier ϑ_i holding the private-key K_i and the function $\mathbb{T}(\vartheta_i)$ outputs the sensor node ϑ_i private-key.

We note that, later, new sensor nodes could be added to the network either to maintain it, or to replace the removed ones. In μKMS , we can add sensor nodes at any time, the base station simply preloads the private-key of the new sensor node. A node can be removed from the network either if it is suspected to be a malicious or if its energy is achieved. In both cases, the base station removes the sensor node identifier and its corresponding private-key from the table $\mathbb{T}$.

3.3. Rekeying in μKMS

This operation happens periodically, in which the sensor node private-keys are authenticated and refreshed. We denote by $K_{i,j-1}$ the current ϑ_i private-key and $K_{i,j}$ its new one. The efficiency of this operation, which is common to the most previous works, is related to the number of communication rounds and the computation overhead. Two sensor nodes wishing to communicate securely, which either do not have a common key or are not within communication range, communicate through a set of intermediate sensor nodes. If any of the latter nodes are compromised, then the message can be altered, and neither the communicating sensor nodes will be able to

detect the alteration. At the round R_j , each sensor node ϑ_i generates its new private-key $K_{i,j}$ and computes the rekeying message RM_i , such as

$$RM_i = ((K_{i,j})_{K_{i,j-1}}, \vartheta_i)_{K_{i,j-1}}. \quad (1)$$

Then, it divides the rekeying message into n parts $P_1, P_2, \dots, P_n$, such as $P_1 \| P_2 \| \dots \| P_n = RM_i$. Afterwards, the sensor node ϑ_i starts to submit each part over a dissimulated packet $\mathbb{P}_{i,BS}(P_l)$ to the base station at each data packet sending or forwarding. Upon receiving all the required parts, the base station concatenates them and extracts the identifier ϑ_i from the packet header. This identifier represents the source sensor node of the transmitted packet carrying the last part P_n . The sensor node identifier is included in order to guarantee the integrity of the message sent to the base station. The base station extracts the ϑ_i current private-key from its locally such as

$$K_{i,j-1} = T(\vartheta_i), \quad (2)$$

with which it decrypts the rekeying message. Finally, it checks if the identifier carried by the rekeying message corresponds to the computed one. If it holds, the base station decrypts the first part of the rekeying message, updates its locally regarding the new private-key $K_{i,j}$ of the sensor node ϑ_i , and responds with a positive acknowledgment to the source sensor node. Otherwise, an attack may be suspected and a negative acknowledgment is returned. When receiving the acknowledgment, the sensor node erases the old private-key $K_{i,j-1}$ with the new one $K_{i,j}$.

3.4. Exploitable coding space by μKMS

There are two specifications available for WSN communication, namely IEEE 802.15.4 [14] and ZigBee [9]. The first is a standard for low-rate wireless personal area networks that was developed by IEEE (Institute of Electrical and Electronics Engineers) and contains a number of security suites. Basically, it provides access control, integrity, confidentiality and replay protection; however, it does not deal with authentication or key management. IEEE 802.15.4 defines a communication layer at the second level in the OSI (Open System Interconnection) model and its main purpose is to allow communication between two devices. ZigBee is built upon IEEE 802.15.4. This standard defines a communication layer at the third level and above in the OSI model. Its main purpose is to create a network topology (hierarchy) to

Figure 2: ZigBee data packet format

let a number of devices communicate among them, and to add extra communication features such as authentication, encryption and association. The ZigBee network layer natively supports star, tree and generic mesh networks [6]. In what follows, we provide a description of the ZigBee data packet structure, highlighting the exploitable coding space used by μ KMS.

Figure 2 illustrates the format of ZigBee data packets on the network-level, which is composed of the frame control, the destination and source addresses, the radius, the sequence number and the data payload. The system μ KMS exploits these fields as follows:

Frame control field: this field includes the frame type, the protocol version and other important control flags. This field is fully used by the network-layer and no coding space could be exploited to embed the μ KMS rekeying process messages.

Destination and source address fields: the used space for the destination and source address fields is proportional to the network size η . Hence, a coding space of

$$CS_{adr} = 32 - 2 \cdot \log_2(\eta) \quad (3)$$

can be used by μ KMS. CS_{adr} denotes in bits the exploitable coding space over the addresses fields. Note that the ZigBee architecture supports at up a network size of 2^{16} sensor nodes. Figure 3 (a) illustrates the size of coding space exploited over the destination and source address fields in function of the network size.

Radius field: this field is used by the network-layer to each sent packet to determine the maximum number of hops. For example, if the initial value of the radius equals to 3, the message will not be relayed more

than three times. The value of the radius is decremented every time the packet is relayed, and when the radius value becomes zero, the packet will no longer be relayed. The radius field is proportional to the network size η and, in the worst case, the maximal length of a given routing path is equivalent to η . Therefore, a coding space of

$$CS_{rad} = 8 - \log_2(\eta) \quad (4)$$

can be used by μ KMS. CS_{rad} denotes in bits the exploitable coding space over the radius field. Figure 3 (b) illustrates the size of coding space exploited over the radius field in function of the network size.

Sequence number field: this field is used by the network-layer to sort the data packets, which are used mainly for acquittal receptions. This field is specific to the targeted application, which is proportional to the size of the observed phenomena. For the maximum space reserved to the payload field, which is 122 bytes, μ KMS exploits in bits from the sequence number field

$$CS_{seq} = 8 - \log_2(122^{-1} \cdot |M|), \quad (5)$$

where CS_{seq} denotes in bits the exploitable coding space over the sequence number field. Figure 3 (c) illustrates the size of coding space exploited over the sequence number field in function of message size.

Data payload field: this field offers no possibility of dissimulation, due to the dynamic space reserved for the data. Adding μ KMS rekeying process messages over this field involves an additional overhead in the network.

In all the cases, the sensed data will be transmitted to the base station. The data packet header is with a size of 8 bytes with or without the disseminated μ KMS rekeying process messages. μ KMS exploits the remaining coding size, which are not used by the network-layer, with no additional communication overhead. The efficiency of μ KMS is highly depended on the network nature and the targeted application. Compared to the other network technologies, WSNs is often targeted for data with lightweight sizes. For example, temperature, humidity, soil moisture, wind (speed and/or direction),

Figure 3: Size of coding space exploited by μ KMS in function of network and message sizes following the ZigBee data packet format

pressure, etc. Figure 3 (d) illustrates the size of coding space exploited by μ KMS in function of both network and message sizes. The estimated value range of the exploitable coding space for WSN applications is highlighted in different color.

3.5. Fragmentation of μ KMS keys

The number n of required parts is an important parameter, which represents the needed number of ZigBee packets to refresh a single private-key. The parameter n depends mainly on the private-key size κ and the exploitable coding space over the packet, which is highly related to the message and network sizes. In fact, when the network and message size increase, the coding space used by μ KMS decrease, and hence, the number of necessary parts increases. The parameter n can be estimated by

$$n = \kappa / \left(48 - 3 \cdot \log_2(\eta) - \log_2(122^{-1} \cdot |M|) \right). \quad (6)$$

The parameter κ is proper for the type of application. If the application requires a high security level, larger private-keys to be considered. In this

Figure 4: The necessary number of ZigBee data packets to send a complete private-key in function of network and message sizes

Family of algorithms	Key size (bits)
2TDEA	80
3DES-112 and 3TDEA	112
Systems of encryption RC4, RC2 and AES-128	128
3DES-168	168
AES-192	192
AES-256	256

case, the number of required parts increases in order to send the complete private-key. The parameter κ depends mainly on the implemented system of encryption. Figure 4 illustrates the ZigBee packet number needed by μKMS in function of both the network and message sizes under different systems of encryption, which are presented in Table ?? . We note that, in the worst case, κ varies between 5 and 15 packets. This is quite good for several types of real-time based WSN applications, when the number of packets to transmit is high.

4. Security analysis

In this section, we analyze the security of μ KMS against the most known and usual attacks in the context of key management. The first part of this section includes an overall discussion about some of its important security features. In the second part, we present a formal analysis under the BAN logic formalism.

4.1. Overall discussion

An attacker attempts to compromise the keys integrity, availability and confidentiality. The physical security of WSN is not considered due to the uncontrolled environment where the nodes are deployed [22, 23]. That's why we consider the attacker having the control of the communication channel on which he/she can perform both passive and active attacks. The attacker can listen the communication channel and replay the collected packets later. Furthermore, he/she can modify them or inject false ones. We suppose that the attacker is not resource-limited in terms of storage and computational abilities.

μ KMS guarantees the private-key exchange integrity even if an attacker modifies the content of packets in transmission. Indeed, when a sensor node sends the new private-key to the base station, an adversary may alter the packets by modifying their contents. However, upon receiving all the packets $\mathbb{P}_{S,BS}\langle P_l \rangle$ and concatenating them, such as

$$RM_S = P_1 || P_2 || \dots || P_n = ((K_{S,j})_{K_{S,j-1}}, \vartheta_S)_{K_{S,j-1}}, \quad (7)$$

the base station extracts the sender sensor node identifier $\vartheta_{S'}$ from the packet header, extracts the sensor node $\vartheta_{S'}$ private-key such as $K_{S',j-1} = T(\vartheta_{S'})$, uses the latter key to decrypt the rekeying message RM_S , and compares ϑ_S to $\vartheta_{S'}$. If $\vartheta_S \neq \vartheta_{S'}$, the base station suspects the attack of alteration and responds to the sender sensor node by a negative acknowledgement.

μ KMS is secure against the replay attack. An attacker may intercept the rekeying messages at the round R_j and replays them later. We distinguish two cases, namely when the replay attack is performed before or after the round R_{j+1} . In the first case, the attacker may replay the rekeying messages before the sensor node changes its current private-key. Therefore, the base station receives the correct concatenated rekeying message with a slight delay without any dysfunctioning. In the second case, when an attacker replays the rekeying messages, the sensor node has already changed its private-key

and the base station is aware about the new private-key. Hence, upon constituted RM_S , the base station could easily verify that the private-key $K_{S',j-1}$ extracted through $\mathbb{T}(S')$ doesn't correspond to the private-key used to encrypt RM_S .

An attacker may try to compromise a sensor node private-key. We distinguish two cases, namely physical and logical attacks. In the case of physical attack, an attacker may capture physically the sensor node from which he/she tries to read the saved private-key. μKMS is secure against this attack, because the sensor node private-key is stored in the volatile memory. Hence, even the sensor node is captured it is difficult for the attacker to obtain the saved private-key. In the case of logical attack, an attacker cannot compromise the private-key. Only the way to get the sensor node new private-key is to have the old one. When a sensor node sends the new private-key to the base station, an adversary may intercept the packet and tries to compute the current private-key. In this case, the robustness of μKMS depends on the reliability of the encryption system.

The Sybil attack happens when a malicious sensor node operates in the network under several identities. In the framework of μKMS , no key exchange is designed among the sensor nodes. The key establishment is performed between the sensor nodes and the base station. Hence, even if a malicious sensor node communicates with several identities, it doesn't compromise the exchanged keys between the base station and the other sensor nodes. The Sybil attacker can be detected by the base station. Upon receiving all the packets, the base station concatenates them and extracts the sender sensor node identity $\vartheta_{i'}$ from the packet header. Then, it extracts its private-key such as

$$K_{i',j-1} = \mathbb{T}(\vartheta_{i'}), \quad (8)$$

uses the latter key to decrypt the rekeying message RM_i , and authenticates the sender sensor node if $\vartheta_i = \vartheta_{i'}$. The function $\mathbb{T}$ returns a unique key, and hence, μKMS resists thoroughly to the Sybil attack.

An attacker may try to capture a sensor node and uses its identity. Even if the attacker succeeds in this operation, he/she can't be aware of the sensor node actual private key, which is preserved in its volatile memory. That's why the attacker with a correct identity can't crypt the latter correctly. Even if the replication attack is happened, it doesn't compromise μKMS . Another variant of this attack scenario can be happened if the attacker breaks the

sensor node functioning and/or its transmitted messages. A such attack can be detected by the base station if the received key part number isn't enough to establish the sensor node key.

This selecting forwarding attack occurs when an attacker selects the messages to forward from/to the base station or sensor nodes. The probability that this attack succeeds is negligible due to the network architecture which is flat. Thus, if a compromised node don't forward a given message, there is a great probability that another node has already received and relayed it. If no other sensor node receives the message, two cases could be considered. In the first scenario, the attacker doesn't transmit the packet containing the key part to the base station. The latter detects a such anomaly since it should wait for a certain number of key parts before constituting the key. In the second scenario, the attacker doesn't transmit the acknowledgement packet to the sensor node. In this case, the latter node will retransmit the key again over another routing path until its message is acknowledged. In both the cases, the security of μKMS is not compromised.

When the sensor nodes send their keys to the base station, an adversary may try to intercept the packets and discover the used encryption key. Even if he/she succeeds in this operation, there is no way to discover the old key from $((K_{i,j})_{K_{i,j-1}}, \vartheta_i)_{K_{i,j-1}}$. The old keys are used only in encryption and they are not part of the message content. Moreover, the old keys are not saved anywhere. Upon receiving the acknowledgment from the base station the sensor node in question removes that key from its memory.

4.2. Formal analysis

In order to test the correctness of μKMS , we use the BAN logic formalism [21]. This formalism allows the description of the beliefs of trustworthy parties involved in the security protocols and the evolution of these beliefs as consequences of rules and messages. Under this formalism, we evaluate the key authentication, secrecy and freshness. The used notation is summarized in Table ??.

From the BAN logic deduction process, we test μKMS under the following rules:

$$\text{R1} : \frac{P \models Q \stackrel{K}{\leftrightarrow} P, P \triangleleft \{X\}_K}{P \models Q \vdash X}, \quad (9)$$

	Notation	Description
Used BAN logic notation	X_i	Key part
	$P \models X$	P believes X
	$P \triangleleft X$	P sees X
	$P \stackrel{K}{\leftrightarrow} Q$	K is never disclosed by any principal except P and Q
	$P \Rightarrow X$	P controls X
	$\#(X)$	X is fresh
	X_1, X_2	Concatenation of X_1 and X_2

$$R2 : \frac{P \models Q \Rightarrow X, P \models Q \vdash X}{P \models X}, \quad (10)$$

$$R3 : \frac{P \triangleleft X_1, P \triangleleft X_2, \dots, P \triangleleft X_n}{P \triangleleft (X_1, X_2, \dots, X_n)}, \quad (11)$$

and

$$R4 : \frac{P \models \#X, P \models V \vdash X}{P \models V \models X}. \quad (12)$$

The first step of the BAN logic deduction process consists of the protocol idealization. We elaborate the logical formula corresponding to each transmitted message. In this context, we model the first round of communication, namely $V \rightarrow S : X_1, V \rightarrow S : X_2, \dots, V \rightarrow S : X_n$, by the following specification

$$I1 : S \triangleleft X_1, S \triangleleft X_2, \dots, S \triangleleft X_n, \quad (13)$$

where S and V represents, respectively the base station and the intermediate/source sensor node, and

$$X_1, X_2, \dots, X_n \equiv \{K'\}_K. \quad (14)$$

We model the second round of communication, namely $S \rightarrow V : \{K'\}_{K'}$, by the following specification

$$I2 : V \triangleleft \{V \stackrel{K'}{\leftrightarrow} S\}_{K'}. \quad (15)$$

The second step consists of the security objectives definition. The first goal relies on the key authenticity and secrecy. Indeed, the base station must

authenticate the sensor node, which sends the new key and must be sure that this key is known exclusively by itself and that node. We model this goal by

$$G1 : S \models V \overset{K'}{\leftrightarrow} S. \quad (16)$$

The second goal relies on the key freshness, where the sensor node must be sure that the base station is aware of the new key. This objective is represented by

$$G2 : V \models S \models V \overset{K'}{\leftrightarrow} S. \quad (17)$$

The third step consists of the translation of assumptions into logical formulas. The base station and the sensor node already having a common secret key. This key is either pre-shared or exchanged during the previous session, such as

$$A1 : S \models V \overset{K}{\leftrightarrow} S. \quad (18)$$

The sensor node itself, which generates the new key K' , and hence, it is evident that the latter node trusts the key freshness. We model this aspect by

$$A2 : V \models V \overset{K'}{\leftrightarrow} S \quad (19)$$

and

$$A3 : V \models \#(V \overset{K'}{\leftrightarrow} S). \quad (20)$$

Finally, the base station is aware that the sensor node is responsible of the new key generation, such as

$$A4 : S \models V \Rightarrow V \overset{K'}{\leftrightarrow} S. \quad (21)$$

The fourth step consists of the goals demonstration using the BAN logic rules, the considered assumptions and the formulas derived from the communication messages. We have to take in charge the set of assumptions (A1, A2, A3, and A4), the set of rules (R1, R2, R3, and R4) and the obtained formulas after the messages idealization (I1 and I2) in order to demonstrate the goal G1 and G2.

To demonstrate G1, we proceed as follows. By using I1 and R3, we get

$$\frac{S \triangleleft X_1, S \triangleleft X_2, \dots, S \triangleleft X_n}{S \triangleleft \{V \xleftrightarrow{K'} S\}_K}. \quad (22)$$

By using the latter result, the assumption A1 and the rule R1, we get

$$\frac{S \models V \xleftrightarrow{K} S, S \triangleleft \{V \xleftrightarrow{K'} S\}_K}{S \models V \vdash V \xleftrightarrow{K'} S}. \quad (23)$$

By using the latter result, the assumption A4 and the rule R2, we get

$$\frac{S \models V \Rightarrow V \xleftrightarrow{K'} S, S \models V \vdash V \xleftrightarrow{K'} S}{S \models V \xleftrightarrow{K'} S}. \quad (24)$$

The latter result represents the first objective corresponding to

$$S \models V \xleftrightarrow{K'} S, \quad (25)$$

in which S authenticates V , and S is sure that K' is secret.

To demonstrate G2, we proceed as follows. By using the rule R1, the assumption A4, and the formula obtained by the idealization I2, we get

$$\frac{V \models V \xleftrightarrow{K'} S, V \triangleleft \{V \xleftrightarrow{K'} S\}_{K'}}{V \models S \vdash V \xleftrightarrow{K'} S}. \quad (26)$$

By using the result obtained in the previous formula, the assumption A3 and the rule R4, we get

$$\frac{V \models \#(V \xleftrightarrow{K'} S), V \models S \vdash V \xleftrightarrow{K'} S}{V \models S \models V \xleftrightarrow{K'} S}, \quad (27)$$

which represents the second objective. The sensor node V is sure that the base station believes the new key K' .

5. Performance evaluation

In this section, we evaluate the performances of μ KMS by simulations. We compare μ KMS with other protocols to underline the efficiency and the suitability of our solution.

5.1. Simulation parameters

The simulations are developed using Matlab programming environment (version 7.11.0.584). We consider a randomly deployed network of a set of sensor nodes in an area of 1000m^2 . The network size η varies between 50 and 1000 sensor nodes and the private-key size regarding the following algorithms: 2TDEA, 3DES-112, AES-128, 3DES-168, AES-192 and AES-256. The sensor nodes have the same physical characteristics and are equipped with wireless communication interfaces with a communication range of 20m. The simulator considers whether a radio link exists between any pair of sensor nodes according to the distance which separates them. We assume that at each period of 30sec, the sensor nodes transmit the sensed data to the base station. We suppose that the data is with a size of 1 byte, the identifier size is of $\log_2(\eta)$, and the rekeying process is executed at each $\Delta T = 150\text{sec}$. We use the same energy consumption model of Heinzelman et al., used for wireless communication hardware in [19]. If the sensor node transmits a k -bit packet over a distance d , it consumes in Joule

$$E_{\text{tra}} = k \cdot (E_{\text{elec}} + E_{\text{amp}} \cdot d^2), \quad (28)$$

where E_{elec} denotes the electrical energy, which represents the energy per bit consumed by the transmitter electronics and E_{amp} denotes the empirical energy, which represents the energy dissipated in the transmission amplifier. When receiving a k -bit packet, it consumes in Joule

$$E_{\text{rec}} = k \cdot E_{\text{elec}}. \quad (29)$$

In the simulations, the empirical energy is set to $100 \cdot 10^{-12}$ Joule and the electrical energy to $50 \cdot 10^{-9}$ Joule.

The performances of our protocol are compared to the protocols MAKM [10] and EEKMS [5], which are described in Section 2. The evaluated criteria are the storage overhead, the transmission overhead, the reception overhead and the communication energy consumption. We have measured the communication load based on the size of the transmitted packets. The impacts studied are the network size and the used system of encryption in a simulation period of 1000sec. Table ?? summarizes the overall of the simulation parameters.

	Parameter	Value
	Simulation time	1000sec
	Area of deployment	1000m ²
	Network size (η)	50 to 1000 sensor nodes
Parameters of simulation	Communication range	20m
	Data size	1 byte
	Frequency of data transmission	30sec
	Period of rekeying process	150sec
	Empirical energy	$100 \cdot 10^{-12}$ Joule
	Electrical energy	$50 \cdot 10^{-9}$ Joule

5.2. Metrics of comparison

The key storage requirement represents the total memory space needed to store the network keys. In MAKM scheme, each sensor node needs to store a key seed with a size of 128 bytes and the base station public-key. The cluster-head holds an authentication key with a size of 16 bytes, a pair of keys with respective sizes of 40 bytes and 20 bytes [18], and a certificate with a size of 86 bytes [15]. The rate of cluster-head number represents 10% regarding the network size. The total storage space is measured in bytes by

$$S_{\text{MAKM}} = \eta \cdot \left((128 + \kappa) \cdot 0.9 + 274 \cdot 0.1 \right). \quad (30)$$

In EEKMS scheme, we don't take in consideration the space storage needed to the keys of neighbor sensor nodes. We consider only the necessary keys used in the rekeying process. If we consider r be the number of polynomial functions and t be their degree, is needed $r \cdot t + r$ bytes for the polynomial functions, one byte for the random number and specific space for the group-key. The total storage in bytes is measured by

$$S_{\text{EEKMS}} = \eta \cdot \left((r \cdot t + r) + 1 + \kappa \right). \quad (31)$$

In the simulations, r and t are set, respectively, to 10 and 3. In μ KMS, each sensor node needs only to store its individual private-key, which is shared only with the base station, so the total storage in bytes is measured by

$$S_{\mu\text{KMS}} = \eta \cdot \kappa. \quad (32)$$

When MAKM updates the keys, the cluster-head sends two independent messages. The first one is transmitted in diffusive manner, which contains its identifier and certificate. The second message is transmitted to the member sensor nodes of its cluster, which contains the key seed. Each sensor node responds to the cluster-head by a message containing its identifier, the hash of its identifier and its key. EEKMS requires to transmit the identifiers of the polynomial function and a random number for each sensor node. In μ KMS, there is no additional message to transmit in the rekeying process. μ KMS exploits the packets, which are sending periodically through the application.

5.3. Obtained results

Figure 5a compares the transmission overhead in function of the network size. It's understandable that the transmission overhead increases with the increasing of the sensor node number. We note that μ KMS is largely better than the other protocols. μ KMS exploits the data transmission by embedding the rekeying information. In the other hand, MAKM and EEKMS generate additional messages, so more overhead in terms of transmission. Figure 5b compares the reception overhead in function of the network size. μ KMS don't generate specific messages in the rekeying process, and hence, the reception overhead doesn't increase. Even if μ KMS follows the best results, but are close to those of EEKMS. This is interpreted by the fact that EEKMS doesn't generate an important amount of messages in the rekeying process in contrast of MAKM. That's why the difference reaches 1400 Kbyte with a network size of 1000 sensor nodes. Figure 5c compares the communication energy consumption in function of the network size. Indeed, the simulations are developed under the Heinzelman et al. model [19], where the energy consumption depends mainly on the communication load (transmission and reception). Following μ KMS, the communication happens only from the data transmission. That's why the energy consumption is lower than the other protocols. Even with the same frequency of rekeying, EEKMS is better than MAKM because of the size of needed messages in the rekeying process is considerably lower than those needed by MAKM.

Figure 6a compares the transmission overhead in function of the system of encryption. The main difference between the used encryption algorithms is the key size. We note that the transmission overhead in EEKMS and μ KMS are constant, which are independent of the used algorithm. This is understandable because EEKMS doesn't transmit keys in the rekeying process. It sends only the polynomial function identifier and a set of stamps.

In μ KMS, the keys are hidden in the communication messages, so the key size doesn't influence the transmission overhead. However, following MAKM, where the key size increases, the transmission overhead increases as well because the transmitted keys influence the message size. Figure 6b compares the reception overhead in function of the system of encryption. Again, we note that μ KMS is better than the other protocols. Following MAKM, the transmission overhead increases with the increasing of the used key size, and hence, the reception overhead increases as well. Figure 6c compares the communication energy consumption in function of the system of encryption. The consumed energy depends mainly on the communication load. The latter remains constant in the case of μ KMS and EEKMS. However, EEKMS

Figure 5: The network size impact

expends more energy than μ KMS, because it generates extra messages in the rekeying process independently of the transmitted data. In MAKM, the energy consumption increases when the key size increases. μ KMS is better than the other protocols. This is justified by the fact that there is no extra information are generated in the key exchange. MAKM needs to exchange keys when rekeying, so the exchanged key size depends on the system of encryption, which highly influences the communication overhead and consequently the energy consumption.

Figure 7a compares the storage overhead in function of the network size. The obtained results demonstrate that μ KMS and MAKM don't require lot of storage space even for 1000 of network size. The storage overhead

(a) Transmission overhead

(b) Reception overhead

(c) Communication energy consumption

Figure 6: The system of encryption impact

Figure 7: Storage overhead

in EEKMS increases considerably when the network size increases. This is interpreted by the registration of all the polynomial functions. In MAKM, 90% of sensor nodes store only the key seed and the base station public-key. In μ KMS, each sensor node stores only its individual private-key. Figure 7b compares the storage overhead in function of the system of encryption, in which the obtained results are also in favor of μ KMS. We note a slight increasing of storage overhead with the different systems of encryption. The results demonstrate that μ KMS is flexible and the storage requirement is approximately independent of the system of encryption. Moreover, we note that μ KMS has no constraints in terms of storage requirements according to the actual development of technology. For example, with the highest size of keys, the storage capacity needed in the entire network is only about 30 Kbytes.

6. Conclusion and future works

In this paper, we have proposed μ KMS, an efficient and reliable key management system for WSNs based on dissimulation. The latter technique is not used only to hide the exchanged keys, but also to make use the un-exploitable coding space with no communication overhead. We have also proposed a mechanism of key fragmentation, communication and reconstitution. Regarding the implemented system of encryption, we have measured

the required packet number to send a complete private-key in function of both network and message sizes. Finally, we have evaluated the performances of μ KMS by simulations with comparison to two concurrent protocols in the literature, in which it demonstrates encouraging results.

In our future works, we propose to contribute to the research activities on energy saving issues, especially in the case of WSNs or in the general context of the Internet of things. Indeed, one of the possible ways that we propose to explore more this aspect of dissimulation by considering other communication standards and other types of exchanging information. The network saves much energy that is often expended while exchanging continually control information, such as energy equipment status, acknowledgments, positioning information, etc. All those useful information could be dissimulated in normal data packets without overhead, and hence, gain thoroughly in terms of efficiency. As part of real-time applications, intensive transfer of packets on the network to offer concealment process the ability to carry a large amount of information.

Acknowledgments

This work was carried out in the framework of research activities of the laboratory LIMED, which is affiliated to the Faculty of Exact Sciences of the University of Bejaia.

References

- [1] Fahmy, H-M-A.: 'Wireless Sensor Networks: Concepts, Applications, Experimentation and Analysis', Springer, Book, Signals and Communication Technology, 2016
- [2] Kamila, N-K.: 'Handbook of Research on Wireless Sensor Network Trends, Technologies, and Applications', IGI-Global, Book, 2016, DOI: 10.4018/978-1-5225-0501-3
- [3] Nisha, Dave, M.: 'Storage as a parameter for classifying dynamic key management schemes proposed for WSNs', Proceedings of the IEEE International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), India, 2016, pp. 51–56

- [4] Seo, S-H., Won, J., Sultana, S., Bertino, E.: 'Effective key management in dynamic wireless sensor networks', *IEEE Transactions on Information Forensics and Security*, 2015, 10, (2), pp. 371–383
- [5] Suganthi, N., Vembu, S.: 'Energy efficient key management scheme for wireless sensor networks', *International Journal of Computers Communications and Control*, 2014, 9, (1), pp. 71–78
- [6] Pietro, R-D., Guarino, S., Verde, N-V., Domingo-Ferrer, J.: 'Security in wireless ad-hoc networks: a survey', *Computer communications*, 2014, (51), pp. 1–20
- [7] Ruj, S., Nayak, A., Stojmenovic, I.: 'Pairwise and triple key distribution in wireless sensor networks with applications', *IEEE Transactions on Computers*, 2013, 62, (11), pp. 2224–2237
- [8] He, X., Niedermeier, M., de-Meer, H.: 'Dynamic key management in wireless sensor networks: a survey', *Journal of Network and Computer Applications*, 2013, 36, (2), pp. 611–622
- [9] ZigBee Alliance: 'Standards: ZigBee Specification', ZigBee Standards Organization, Document ID 053474r20, 2012, URL: <http://www.zigbee.org/download/standards-zigbee-specification/>
- [10] Du, D., Xiong, H., Wang, H.: 'An efficient key management scheme for wireless sensor networks', *International Journal of Distributed Sensor Networks*, 2012, Article ID 406254
- [11] Challal, Y., Ouadjaout, A., Lasla, N., Bagaa, M., Hadjidj, A.: 'Secure and efficient disjoint multipath construction for fault tolerant routing in wireless sensor networks', *Journal of Network and Computer Applications*, 2011, (34), pp. 1380–1397
- [12] Simplicio, M-A., Barreto, P-S-L-M., Margi, C-B., Carvalho, T-C-M-B.: 'A survey on key management mechanisms for distributed wireless sensor networks', *Computer Networks*, 2010, 54, (15), pp. 2591–2612
- [13] Zhang, W., Shen, Y., Lee, S.: 'A cluster-based group key management scheme for wireless sensor networks', *Proceedings of the 12th Asia-Pacific Web conference*, 2010

- [14] IEEE: 'IEEE Standard 802.15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks', LR-WPANs, 2006, URL: <http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf>
- [15] Meulenaer, G., Gosset, F., Standaert, F-X., Pereira, O.: 'The energy cost of communication and cryptography in wireless sensor networks', Proceedings of the IEEE international conference on wireless and mobile computing, networking and communication, 2004, pp. 580–585
- [16] Chan, H., Perrig, A., Song, D.: 'Random key pre-distribution schemes for sensor networks', Proceedings of the 2003 IEEE Symposium on Security and Privacy, 2003, pp. 197–213
- [17] Joux, A.: 'A one round protocol for tripartite Diffie-Hellman', Chapter Book, Algorithmic Number Theory, 2000, Volume 1838 of the series Lecture Notes in Computer Science, pp. 385–393
- [18] Certicom Corporation: 'SEC 1: Elliptic Curve Cryptography', Standards for Efficient Cryptography, 2000
- [19] Heizelman, W., Chandrakasan, A., Balakrishnan, H.: 'Energy-efficient communication protocol for wireless sensor networks', Proceedings of the Hawaii international conference on systems, 2000
- [20] Shamir, A.: 'How to share a secret', ACM Communications, 1979, (22), pp. 612–613
- [21] Burrows, M., Abadi, M., Needham, R.: 'A Logic of Authentication', ACM Transactions on Computer Systems, 1990, 8(1), pp. 18–36
- [22] Cho, H-H., Chen, C-H. Shih, T-K., Chao, H-C.: 'Survey on underwater delay/disruption tolerant wireless sensor network routing', IET Wireless Sensor Systems, 4(3), 2014, pp. 112–121
- [23] Jokhio, S-H., Jokhio, I-A, Kemp, A-H.: 'Node capture attack detection and defense in wireless sensor networks', IET Wireless Sensor Systems, 2(3), 2012, pp. 161–169