KMMR: An Efficient and scalable Key Management Protocol to Secure Multi-Hop Communications in large scale Wireless Sensor Networks

Abderrahmen Guermazi[1], Abdelfettah Belghith[2,*], Mohamed Abid[3] and Sofien Gannouni[2]
[1] ISET Sfax, CES Research Lab- University of Sfax, Tunisia
[E-mail: abguermazi@gmail.com]
[2] College of Computer and Information Sciences, King Saud University, Saudi Arabia
[E-mail: abelghith@ksu.edu.sa, gnnosf@ksu.edu.sa]
[3] ENIS, CES Research Lab-University of Sfax, Tunisia
[E-mail: mohamed.abid@enis.rnu.tn]
*Corresponding author: Abdelfettah Belghith

Received May 14, 2016; revised September 16, 2016; revised December 3, 2016; accepted December 17, 2016; published February 28, 2017

Abstract

Efficient key distribution and management mechanisms as well as lightweight ciphers are the main pillar for establishing secure wireless sensor networks (WSN). Several symmetric based key distribution protocols are already proposed, but most of them are not scalable, yet vulnerable to a small number of compromised nodes. In this paper, we propose an efficient and scalable key management and distribution framework, named KMMR, for large scale WSNs. The KMMR contributions are three fold. First, it performs lightweight local processes orchestrated into upward and downward tiers. Second, it limits the impact of compromised nodes to only local links. Third, KMMR performs efficient secure node addition and revocation. The security analysis shows that KMMR withstands several known attacks. We implemented KMMR using the NesC language and experimented on Telosb motes. Performance evaluation using the TOSSIM simulator shows that KMMR is scalable, provides an excellent key connectivity and allows a good resilience, yet it ensures both forward and backward secrecy. For a WSN comprising 961 sensor nodes monitoring a 60 hectares agriculture field, KMMR requires around 2.5 seconds to distribute all necessary keys, and attains a key connectivity above 96% and a resilience approaching 100%. Quantitative comparisons to earlier work show that KMMR is more efficient in terms of computational complexity, required storage space and communication overhead.

Keywords: Wireless sensor network; Key management Protocol; Secure multi-hop

The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through research group no RGP-1436-031.

https://doi.org/10.3837/tiis.2017.02.015 **ISSN : 1976-7277**

1. Introduction

Wireless Sensor Networks (WSNs) are often deployed in open areas [1] where they are naturally exposed to many attacks varying from passive eavesdropping to sensor node compromising [2]. Aggressors may obtain secret sensible information, disturb the network functioning or deprive the WSN application from receiving significant amount of data. Secure communications in WSNs have become one of the major concerns for designers [3][4]. Security services (integrity, authentication, confidentiality and privacy) are based on cryptographic algorithms and need a key sharing mechanism. As a result, a Key Management Protocol (KMP) is essential to design secure WSNs [5][6][7][8]. A KMP for WSNs has to efficiently perform : (1) distribution of required keys; (2) node revocation; (3) secure node addition and (4) keys updating.

In this paper, we propose an efficient KMP named KMMR to secure communications in very wide WSNs which are often deployed in large area, like agriculture fields, reserves and forests. Extra wide WSNs using innovative sensing capabilities have already been considered for the optimization of agrochemical application in olive groves, fields monitoring in precision agriculture, soil humidity and precision irrigation [9][10][11][47][51][53]. We argue that, it is practical to use multi-hop routing for wide WSNs, where the base station is only reached by close nodes. KMMR is intended to facilitate secure multihop routing for such WSNs applications by performing key distribution, node revocation, secure node addition and updating the different keys in an efficient way. Since sensor nodes are resource constrained equipments (low computing, limited storage, low range), conventional KMPs such as the sophisticated public key cryptographs RSA[16] cannot be deployed to WSNs. For instance, a Crossbow's Telosb mote (TPR2400) [12] has an 8 MHz MSP430 micro-controller with 10K bytes of RAM and just 48K bytes of program flash memory. The CC2420 radio module provides no more than 250 kbps and an effective transmission range up to 50 meters. Moreover, the mote is powered with two AA batteries and cannot be replaced in harsh environments. For all these reasons, we argue that an efficient KMP which is intended to secure communication in WSNs must be lightweight. This can be achieved by minimizing the computational complexity, communication and storage overhead. A KMP conceived for WSNs should achieve a trade-off between robustness and low-cost. Metrics can be used to verify this very aspect. They mainly fall into two categories, security metrics and efficiency metrics.

For a KMP, security metrics are mainly key connectivity, network resilience, forward and backward secrecy. Key connectivity is usually defined as the probability for sensor nodes to be able to share security keys [13]. Local connectivity concerns just the neighbors of a node while global connectivity concerns the entire network. A KMP having a weak key connectivity may jeopardize the network proper operation. Network resilience is usually measured by the fraction of the network that may be affected by an adversary [14]. A KMP has a good resilience when the impacted fraction of the network is restricted to the communication links with the captured node. Forward secrecy is useful to deny a sensor node from using an old (obsolete) cryptographic key [15]. As such, forward secrecy permits to overcome the negative impact of captured nodes. Whereas, backward secrecy is useful to deny a sensor node from going backwards in time to decrypt earlier received messages which are encrypted with previous keys [15].

Efficiency metrics ascertain the computing and storage requirements of a KMP and its induced communication overhead. KMMR does not use asymmetric cryptography; rather it uses a lightweight symmetric cryptography, hash functions and efficient pseudo-random functions [34, 35]. According to Giacomo and all [16], communication is the most expensive operation in terms of energy consumption for WSNs. KMMR reduces the number of exchanged messages by organizing its transmissions to establish all the shared keys into an upward phase and a downward phase according to a tier based architecture.

The remaining of this paper is organized as follows: Section 2 presents an overview of the basic key distribution schemes and discuss their suitability to secure multi-hop routing in wide WSNs. Section 3 provides a detailed description of all the KMMR operations. Section 4 presents the security analysis and investigates the robustness of the KMMR protocol against known attacks. Experimentation and simulation results as well as performance tests are discussed in section 5. KMMR cost evaluation and comparison to other KMPs are presented in section 5. Finally, conlusions are presented in Section 6.

2. Related Work

Resource constrained sensor nodes are unable to run sophisticated public key cryptography like RSA [17]. Elliptic Curve Cryptography [18][19] promised an easy key management, however it requires a large memory space which in turn inhibits the running of WSN applications [18]. As such, this latter scheme is not suitable to homogenous WSNs lacking super nodes having extra computing and storage capabilities.

Key pre-distribution schemes have been proposed to achieve secure communications in WSNs. Sensor nodes are preloaded with required keying material prior to their deployment. Upon deployment, sensor nodes exchange necessary keying information and establish their required keys. A low-cost scheme consists of preloading all sensor nodes with the same master key. Each pair of nodes can then establish a secure communication link with the lowest computing, storage and communication overhead [20]. This scheme presents, however, the worst resilience as compromising a single node results in compromising all secure communication links. The resilience of this scheme can be significantly improved by erasing the master key once the other keys are established and shared. A much more immune scheme is to distribute a unique pairwise key for each pair of sensor nodes. However, each sensor node must store (N-1) keys, where N is the network size and therefore this scheme lacks scalability. Moreover, it remains difficult for sensor nodes to share keys with new added nodes.

Gligor and al. [21] proposed a random key pre-distribution scheme. In the pre-deployment phase, each sensor node is preloaded with a set of keys taken from a large pool of keys. After deployment, sensor nodes undergo a distributed discovery process where each node broadcasts the list of its key identifiers. Neighboring nodes with common identifiers may establish pairwise key from the pre-shared keys. To guarantee good connectivity, the number of preloaded keys in each node increases linearly with the number of nodes in the network which leads to an important storage requirement in large-scale WSNs. Moreover, when a node is compromised, a great number of keys will be acquired and an adversary may alter other communication links. For a better resilience, other extensions were proposed to improve this basic scheme by employing polynomial pool based key pre-distribution schemes [22], location information [23], [24] or heterogeneity [25]. We argue that these schemes cause an important additional computing load and are not appropriate for homogenous WSNs.

Authors in [26] proposed a security infrastructure called SPINS which consists of a suite of security protocols to secure communications in WSNs. SPINS comprises two secure building blocks: SNEP (Sensor Network Encryption Protocol) and µTESLA (Timed Efficient Stream Loss-tolerant Authentication). SNEP provides Data confidentiality, authentication, integrity, and freshness. µTESLA provides authenticated broadcasting by only using symmetric keys. In SNEP, the base station (BS) plays the role of a Key Distribution Center (KDC). It establishes a session key between every pair of nodes. The scheme has a small memory requirement. The impact of a node capture is limited to just a small part of the network. It only affects wireless links with the neighboring nodes. In wide WSNs, this scheme incurs high communication burden on lengthy multi-hop routing paths, and as such it is not scalable. Furthermore, the failure of the BS will impact the security system of the entire network. TESLA [27] is proposed to authenticate the source of broadcasting. Nevertheless, TESLA is not designed for resource constrained equipments. In fact, it uses expensive computational asymmetric cryptography to authenticate the initial packet. In µTESLA, the Base Station generates a one-way key chain. It randomly chooses the last key K_n, and repeatedly applies a one-way hash function (H) to compute all other keys so that $K_{i-1}=H(K_i)$. Then, the first key K_0 (the last generated key) is preloaded in all sensor nodes and serves as a commitment key. Moreover, µTESLA requires that the BS and all sensor nodes to be loosely synchronized. The sender associates each key of the one-way key chain with a time interval. To send an authenticated packet, the BS computes a Message Authenticated Code (MAC) value on the packet with the next secret key K_j. When a sensor node gets a packet, it verifies that the corresponding key has not yet been disclosed by the BS. This verification is based on its loosely synchronized clock, its maximum synchronization error and the time schedule at which keys are to be disclosed. Since a receiving sensor node is sure that key K_j is only known to the BS, no adversary could have had altered the packet in transit. The sensor node stores the packet in a buffer. At the time of key disclosure, the BS broadcasts the verification key K_j to all receivers. When a sensor node receives the disclosed key, it can verify the correctness of the key by checking that $K_{j-1}=H(K_j)$. Thereafter, the authenticity of the stored packet can be checked. However, in wide WSNs it remains difficult to maintain synchronization. Moreover, the synchronization protocol may itself be subject to attacks.

LEAP [28] and its enhanced version LEAP+ [29] are referred to as the basic schemes of hierarchical KMPs. LEAP is designed to support in-network processing. LEAP+ permits to each sensor node the establishment of 4 types of keys: an individual key which is pre-shared with the BS, a pairwise key shared with each neighbor, a cluster key shared with all neighbors and a global key shared with all network nodes. When the WSN is deployed, each node broadcasts a hello message. Each two adjacent nodes calculate a common pairwise key through a preloaded initial key. Thereafter, the initial key is erased from all nodes' memories. It is assumed that a sensor node cannot be compromised before a time period t_{min}. A sensor node requires a minimum test time, t_{est}, to identify its neighboring nodes and establish with them the Pairwise Keys. It is expected that $t_{est}<t_{min}$. Otherwise, when an adversary captures a sensor node, it can easily pick up the initial key and deduces all established Pairwise Keys. Each sensor node has its own cluster key. The distribution of a cluster key is protected with the established pairwise keys. If a sensor node has *d* neighbors, it needs *d* messages to distribute its cluster key. The distribution of the global key can be protected with the cluster keys. To authenticate the announcement of a node revocation, LEAP uses µTESLA. This requires that all nodes are synchronized. Local broadcast authentication is carried out using the one-way key chain [30].

In LEAP the broadcasting of hello messages is not authenticated; an aggressor may inject false hello messages to start a resource exhausting attack. Moreover, Deng et al. [31] showed that the pairwise key establishment scheme used by both LEAP and LEAP+ is not that robust. Indeed, when an adversary discovers the initial key, it may derive the pairwise key shared between any pair of nodes. They recommended that the initial key must be used for secure pairwise key exchange. Therefore, pairwise keys should be generated randomly instead of deriving them from the initial key. The rather high activity of broadcasting when the WSN is deployed is another shortcoming of LEAP. Accordingly, each node broadcasts a hello message to proclaim its presence, and waits for acknowledgment messages to authenticate its neighbor nodes. This storm of broadcasting cannot be exempt of collisions which may severely affect the key connectivity. Furthermore, LEAP does not determine how all sensor nodes can be aware about the beginning of the deployment phase. A phase shift between sensor nodes leads to the erasure of the initial key before the completion of the establishment of the necessary pairwise keys which severely disrupts the establishment and sharing of the remaining keys.

Spanning Tree Key Management [32] (STKM) builds a tree for the rekeying process. Each sensor node S_i is loaded with three keys: a global key K_r and two individual keys (K_{BS},S_i) and (K_{Si},BS) shared with the BS. A hello message is used to construct the spanning tree. The BS is considered as the root of the tree. The hello message is protected with K_r. When a sensor node S_i detects a malicious neighbor node, it sends to the BS a secure REFRESH-REQ message containing the ID of the malicious node. Upon receiving a REFRESH-REQ message, the BS generates a new K_r and broadcasts a secure REFRESH message containing this new K_r and a list of malicious nodes. The REFRESH message is protected with the (K_{BS},S_i). The REFRESH Message must reach each sensor node within the WSN. This incurs a high communication burden in wide WSNs which jeopardizes the scalability of STKM. Moreover, when a node is compromised, the key K_r is revealed and all secure links are exposed to adversaries. STKM does not provide a good resilience.

Sam Blackshear and al. proposed the Hybrid key management scheme R-LEAP+ [33] that combines two protocols: LEAP+ [29] and the Random Pre-distribution scheme [21]. As such, R-LEAP provides a hybrid solution to the problem of key distribution via a time-limited key establishment and a randomized key pre-distribution. R-LEAP preserves the desirable properties of each scheme. Another hybrid key management protocol is proposed in [34]. This scheme enhances LEAP by using random seeds in the key generation. Therefore, each sensor node receives a ring which contains seeds randomly selected from a large pool. Two neighbor nodes seek for shared seeds. Thereafter, they arrange a random number and use it with a permutation function in order to transform the seed. The generated new seed is used as input for a pseudo-random function which is computed with the Master Key in order to share different types of keys. The hybrid approach is not convenient to secure long multihop routing paths. Rather it is more suitable to distribute keys for cluster based-routing protocols where all nodes can directly access the base station. Moreover, if many nodes are captured the seeds can be revealed, and an adversary may perform the same key generation. Furthermore, this approach is not deterministic and it can incur a storage overhead for wide WSNs.

3. The KMMR Protocol

In designing KMMR, we opted to use a Temporary Key for a short period in order to distribute the required keys. This solution seems to stand as the lowest-cost among all key distribution schemes. Key distribution uses a lightweight local process, which permits to

reduce significantly the number of exchanged messages, the computational complexity and the storage space. Furthermore, this technique is valid for any sensor network topology. As recommended in [31], the derivation of the required keys does not use the Temporary Key. A good pseudo-random function and a one-way hash function serve to generate on the fly all key values. The Temporary Key serves only for a short period to secure the key distribution; after which and it disappears from all sensor node memories. KMMR distributes two categories of keys: the broadcast keys and the unicast keys. The broadcast keys category contains two types of keys: the Global Broadcast Keys (one-to-all) and the Local Broadcast Keys (one-to-many). The unicast keys category contains the Pairwise Keys (one-to-one) and the Individual Keys; the latter are pre-shared with the BS to ensure privacy. To authenticate the Global Broadcast, KMMR uses a once authentication key extracted from a one-way key chain [30], [4]. However, it does not require time intervals like µTESLA.

3.1 Overview

Before deployment, each sensor node is pre-loaded with keying material: a Temporary Key (TK), an Individual Key (IK) and a commitment key K0 which is the first key of a one-way key chain already generated and saved by the BS. The TK is intended to be used for a very short period of time. When the WSN is deployed, KMMR uses the TK to distribute Local Broadcast Keys (one-to-many) and Pairwise Keys (one-to-one) in a secure manner. The TK is used during both of the following phases:

- Phase1: distribution of Local Broadcast Keys and establishment of the secure network structure
- Phase2: Pairwise Keys distribution and confirmation of created secure links

At the end of Phase2, the TK is erased from the memory of all the deployed sensor nodes. In addition to the features performed in Phase1 and Phase2, KMMR carries the distribution of the Global Broadcast Key, the Node revocation, and the Secure nodes addition.

Distribution of Local Broadcast Keys and establishment of the secure network structure: the start of Phase1 marks the beginning of the WSN deployment. The BS initiates the broadcast of a Secure Path Discovery Message (SPDM) that permits to create the secure network structure and assign each sensor node to a given tier. In addition, the broadcast of the SPDM allows the secure distribution of Local Broadcast Keys (LBKs). Recall that LBKs are useful to secure the one-to-many communications. For instance, in the Directed Diffusion protocol [37], the diffusion of interest messages can be protected by LBKs. Likewise in LEACH [38] and its variants [41], intra-cluster communications between a cluster head and member nodes can be secured with the Local Broadcast Key which has the same utility as that of a cluster key.

Pairwise Keys distribution and confirmation of established secure links: in Phase2, each sensor node locally shares Pairwise Keys (PWKs) with its neighbors. Indeed, a generated PWK is incorporated in a Join Request Message. After that, a Join Response Message is sent back to confirm the sharing of the PWK and the establishment of the secure link. At the end of Phase2, KMMR enforces the erasure of the TK from all sensor nodes. This should be accomplished before the minimum time T_{min} required by an aggressor to be able to read data from the internal memory of a sensor node. Studies carried out in [31] affirmed that it is possible for an experimented attacker to pull keys from a memory of a Telosb mote or a Micaz mote in 10 seconds, had he/she knows the exact memory locations of these keys. Several techniques can protect sensors against tampering, however, tamper-proof sensor nodes are very expensive and KMMR is rather designed to be used with ordinary sensors. We then

tacitely consider that the maximum time to erase the Temporary Keys from the entire network should not exceed T_{min} = 10 seconds. We shall show that KMMR requires a much lower time.

Distribution of the Global Broadcast Key : besides the LBK and PWK distributions, KMMR is able to securely distribute a Global Broadcast Key (GBK) which is useful to protect the one-to-all traffic. The Base station generates the new GBK and proceeds by a global broadcast. The global broadcast is authenticated with the next key taken from the one-way key chain and protected with the LBKs.

Node revocation: KMMR not only provides key distribution, but also node revocation. Compromised nodes must be revoked. To do so, all PWKs shared with a compromised node must be erased from healthy neighbors. Moreover, the LBK of non-compromised neighbors must be updated. It should be noted that an IDS should be used to detect node capture or the existence of malicious internal nodes.

Secure node addition: the WSN should be rejuvenated as nodes are energy depleted or revoked. New sensor nodes must authenticate with existing nodes and vice-versa. Thereafter, they should be able to share required keys (LBK, PWK and GBK). To this end, KMMR uses the keying material (the Temporary Key and the one-way Key chain) in the same manner as in the initial deployment. This allows secure network reorganization regardless of the number or the positions of these new added nodes.

3.2 Phase1: Distribution of Local Broadcast Keys and establishment of the secure network structure

To deploy the WSN, the BS initiates the broadcasting of a Secure Path Discovery Message (SPDM). The SPDM is protected by the TK, hence the Message Authenticated code (MAC) is used to make sure that the SPDM is not coming from an external node. Moreover, the BS incorporates the commitment key K_1 such that $K_0 = H(K_1)$ in the SPDM to prevent a malicious internal or external node to be able to start Phase1. A nonce value is used to avoid the replay attack. Moreover the SPDM includes necessary fields to share the LBK and the tier. By default, the BS is assigned to tier zero.

$$SPDM(BS \Rightarrow *) : Tier, K_1, Nonce_{BS}, Enc_{TK}(LBK), MAC_{TK}(M)$$

Upon the reception of an authentic SPDM (Received $MAC_{TK}(M)$ = Computed $MAC_{TK}(M)$ and $K_0 = H(K_1)$), the sensor node stores the received nodeID, tier, and LBK in its Secure Neighbors table. The SPDM is rejected if not authentic. The receiver nodes which are the neighbors of the BS, generate their proper LBKs, get assigned to tier 1 and prepare the broadcasting of an SPDM by updating the LBK, the tier and the MAC fields. As mentioned earlier, a key value can be generated with a low-cost hash function and a good pseudo-random function. The transmission of the prepared SPDM is subject to a randomization period selected by each node as explained next. In turn, these nodes broadcast their SPDM to reach nodes in tier 2 and henceforth progressively and consecutively the different tiers are established until leaf nodes are reached. The local start time of Phase1 at any given node, denoted by T_{s1}, is set to the instant of the local reception of the first authentic SPDM.

$$T_{s1} = Reception\ Instant\ of\ the\ first\ authentic\ SPDM$$

In KMMR, all receiver nodes but the direct neighbors of the BS wait for a predefined collecting period τ_1 to collect all incoming SPDM messages from the previous lower tier. The

tier to which the node gets itself assigned to is equal to the minimum numbered tier among all received SPDM plus one. Then, the receiver node generates its proper LBK and prepares the broadcasting of an SPDM by updating the LBK, the tier and the MAC fields.

The rational behind the use of a collecting period is two fold. First, it allows the node to get assigned to the proper tier. Second, it permits to rebroadcast to the next tier just one SPDM as a response to all the ones received from the previous tier. The rebroadcast of the SPDM allows each sensor node to create secure links with its neighbors and share its LBK. Thereafter, all possible secure multi-hop paths are traced. It should be noted that neighbor nodes belonging to the previous tier are considered as secure routers towards the base station.

Collision reduction in Phase1: Often, reducing the number of transmission collisions is crucial in WSNs as it improves the energy saving. For KMMR, this is extremely important because in addition to energy saving, collisions can significantly impact the key connectivity and thereafter the establishment of the secure links.

To decrease the number of collisions in KMMR, some measures are taken. On one-hand, we have decreased the number of SPDMs transmitted throughout the sensor network by the use of the collecting period τ_1. Furthermore, transmissions of the SPDM are subject to a randomization period τ_2. Any node but the BS, selects a random waiting within τ_2 before attempting to transmit. The period τ_2 is composed of a large number of mini slots. In our KMMR, the slot duration is set equal to the transmission of the largest control message. For instance, in the case of a Crossbow's Telosb mote (TPR2400) [12] having a transmission rate of 250 Kbps, the slot duration is around 1.5 millisecond as the largest control packet length in KMMR does not exceed 49 bytes including the header of the underlying Physical layer and Medium Access Control header [40]. The last slot of period τ_2 is reserved to account for the last scheduled transmission. The question naturally arises as how to fix the length of τ_2, or equivalently how many slots do we need in τ_2 so that the probability of a collision gets very low. Let M be the number of mini slots in the randomization period τ_2. Let n be number of nodes having a packet to transmit (i.e.; the number of transmitting neighbors). Then the probability that these n nodes select exactly m slots, $m \leq n$, is readily given by:

$$P_r[m, n, M] = \frac{\binom{M}{m}\binom{n-1}{m-1}}{\binom{n+M-1}{M-1}}$$

The probability of having free collision transmissions given M mini slots and n transmitting neighbors is the probability to select exactly n mini slots, that is m = n. It is then given by:

$$P_r[\textit{collision free transmissions / M and n}] = P_r[n, n, M] = \frac{\binom{M}{n}}{\binom{n+M-1}{M-1}} \quad \textbf{(1)}$$

From **Eq. 1**, we readily deduce that for a small number of intended transmitting neighbors (up to 4), the probability of collision free transmissions approaches 1 given that the number of mini slots is higher than 60. For the adopted sensor deployment (see **Fig. 1**), any upward (namely during Phase1) or downward (namely during Phase2) transmission has indeed a number of intended neighbors equals to 3 for all sensors but those on the diagonal where it is 5 instead. Consequently, for our case of a Crossbow's Telosb mote (TPR2400) [12], using a randomization period of 100 milliseconds accounts for 66 mini slots which are sufficient as per **Eq. 1** to have a very low probability of collision.

Now, each node having an SPDM to transmit, first selects a random number of mini slots within τ_2 and then behaves according to the underlying Medium Access Control Protocol (e.g.; Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA)) where the selected number of slots defines the waiting time in the idle channel state. In the case of a Time

Division Multiple Access (TDMA) Medium Access Control, the selected slot represents the transmission slot itself. Taking into account the collecting and the randomization periods, the local instant to broadcast (namely to submit a transmission request to the underlying Medium Access Control) its SPDM, denoted by T_b, is then given by:

$$T_b = T_{s1} + \tau_1 + \tau_2 \quad \textbf{(2)}$$

3.3 Phase2: Pairwise Keys distribution and confirmation of established secure links

The sole objective of Phase2 is for each node to establish secure links with its candidate routers by sharing with each of them a PWK. This is achieved by the exchange of two control messages: a Secure Join Router Message (SJRM) and a Secure Router Response Message (SRRM). The transmission of these control messages are also subject to the same constraint of randomization in the same way as described in Phase1. We use here the same randomization period $\tau 2$. No collecting period is neither needed nor used in Phase2.

A SJRM includes the node tier, a nonce value and the encrypted PWK. As explained above, a sensor node S is able to generate encryption keys with a low-cost hash function and a good pseudo-random function. This message is protected by the TK. The nonce value is useful to avoid replay attacks.

$$SJRM(S \rightarrow R) : Tier, NonceS, Enc_{TK}(PWK), MAC_{TK}(M)$$

When a router node receives an authentic SJRM, it stores the shared PWK in its Secure Neighbors table and then confirms the sharing of the PWK by sending a SRRM to the subordinate node. The SRRM is also protected with the shared PWK.

$$SRRM(R \rightarrow S) : Tier , NonceR , NonceS , MAC_{PWK}(M)$$

Upon receiving an authentic SRRM as a response to its SJRM, the subordinate marks the secure router as available and the creation of the secure link is confirmed. It should be noted that the erasure of the TK must be done as soon as Phase2 is accomplished, that is to say when a router node has shared the PWKs with all expected subordinate nodes.

Acceleration of the beginning of Phase2: To avoid possible interference and collisions between traffic belonging to Phase1 and Phase2, the simplest solution is to start Phase2 when the entire network has finished Phase1. As such, the communication channel is cleared from any traffic belonging to Phase1. However, this solution lengthen the time needed to finish Phase1 and Phase2, and is not appropriate for large WSNs. The objective is to try to finish Phase2 as soon as possible to proceed to the erasure of the TKs, and at the same time not exceeding T_{min}. The earlier we complete Phase2, the sooner we can erase the temporary keys and consequently the better against any eventual aggression.

Indeed, a sensor node is not obliged to wait launching Phase2 until all sensor nodes complete Phase1. The ground base rule here is to make sure that Phase1 traffic does not interfere with that of Phase2. This is essentially the rationale behind the multi tier structure adopted in KMMR. To enforce this ground base rule, it suffices to separate both traffic by at least 2 tiers, that is when Phase1 is being executed by nodes at tier i, Phase2 may be started by nodes belonging to tiers less or equal to $(i - 3)$. For instance, if we consider the network topology portrayed in **Fig. 1**, a sensor node S belonging to tier i such that $i \geq 1$ may start Phase2 when the broadcast of Phase1 has reached tier j such that $j \geq (i + 3)$. Let T_{s2} be the local earliest instant at which a sensor node can start its Phase2 from the instant it had received the first authentic SPDM, that is the instant it had started its Phase1. We readily have:

$$T_{s2} \geq T_b + 2 \times (\tau_1 + \tau_2) = T_{s1} + 3 \times (\tau_1 + \tau_2) \quad \textbf{(3)}$$

This enhancement reduces significantly the waiting time for a node to start Phase2. As explained above, a sensor node must erase its TK as soon as it completes Phase2. Indeed, when a node finishes the sharing of the PWKs with its routers (parent and sibling nodes established by Phase1), it waits that its subordinate nodes (its neighbors of the next tier as established by Phase 1) share with it their PWKs. Let T_e be the instant at which the node can erase its Temporary Key. T_e is computed based on the period of randomisation of nodes in the immediate higher tier and their T_{s2}. The exchange of the two messages SJRM and SRRM should be accomplished with each subordinate node at the immediate higher tier before the erase of the TK.

$$T_e = Ts_2 + 2 \times \tau_2 + \tau_1 \quad \textbf{(4)}$$

Whereas, leaf nodes erase the TK as soon as they finish the sharing of PWKs with their parent and sibling nodes.

3.4 Global Broadcast Key distribution

The Global Broadcast Key GBK is known by the entire network and updated periodically. A secure distribution of the GBK consists in protecting it with the LBKs. As such, we tacitely assume that the GBK distribution may start at any time but after the erasure of the last TK in the network. As a result, the corresponding signaling traffic does not interfere with that of Phase1 and Phase2. The BS encrypts the new GBK with its LBK and broadcasts it. Nodes which are directly connected to the BS can decrypt the new GBK. In turn, they protect the GBK with their own LBK and rebroadcast it to sensor nodes belonging to the next tier. This process is repeated until leaf nodes are reached, and can protect the GBK distribution from external attacks. However, a malicious internal node may impersonate the BS and launches a false update of the GBK. To avoid such attacks, the broadcast of the GBK must be authenticated. To this end, KMMR uses a one key authentication which is included in an Authenticated Broadcast Message (ABM). The disclosure of the hash key is done after a period of time sufficient to guarantee that the message broadcast has reached all sensor nodes. In addition, to avoid those malicious nodes from distributing a false GBK, the BS incorporates the next key (k_i) in the ABM, and the MAC field is computed with the key k_{i+1} such that $k_i = H(k_{i+1})$. A random text is encrypted with the new GBK and included in the ABM.

$$ABM(BS \Rightarrow *): K_i, Enc_{GBK}(Text) , MACK_{i+1}(M)$$

When the diffusion of the ABM is at least two tiers away, the BS prepares the broadcasting of a second message: the Hash Key Disclosure Message (HKDM). As its name suggests, it permits to disclose the hash key K_{i+1} and to distribute securely the new GBK. To ensure confidentiality, the new GBK is encrypted at each hop with the correspondent LBK. To be sure that the new GBK is really generated by the BS, the text which corresponds to the earlier cipher text is sent in clear.

$$HDKM(BS \Rightarrow *): K_{i+1}, Enc_{LBK}(GBK) , Text , MAC_{LBK}(M)$$

When a sensor node receives the Hash Key Disclosure Message, it can verify with the disclosed hash key k_{i+1} the authenticity of the ABM and it can decrypt the new GBK. Moreover, if $Enc_{newGBK}(Text)$= Received Cipher Text, the sensor node will be sure that the new GBK is not altered by an internal malicious node.

3.5 Node revocation

For many Intrusion Detection Systems [41, 52], detection of malicious nodes may be accomplished by just monitoring the behavior of neighbors. Customarily, the BS represents the server to manage the IDS [41] in the WSN. Our proposed KMP takes into account this

consideration and provides secure communication between non compromised sensor nodes and an IDS which operates in the BS. Secure Report Messages (SRM) containing local events and neighbors monitoring information are sent from sensor nodes to the BS. SRM are protected with the Individual Key.

$$SRM(S \rightarrow BS): RouterID, TimeStamp, Enc_{IK}(IDSuspectNode, report), MAC_{PWK}(M)$$

When the IDS server detects compromised nodes, the BS sends an Authenticated Broadcast of Alarm Message (ABAM) which contains a black-list. Two sequential keys are extracted from the one-way key chain: K_j and K_{j+1}. K_j is incorporated in the ABAM. As explained above, key K_j allows sensor nodes to be sure that the BS is indeed the one that started the broadcasting of the ABAM, by checking that $K_{j-1}=H(K_j)$. Key K_{j+1} is used in computing the MAC field. The ABAM is authenticated with the delayed disclosure of the hash key k_{j+1}. It is crucial to incorporate the black-list in the ABAM as this allows sensor nodes to discover the list of malicious nodes as soon as possible.

$$ABAM(BS \Rightarrow *): K_j, Enc_{GBK}(BlackList), MAC_{Kj+1}(M)$$

The BS sends the Hash Key Disclosure of Alarm Message (HKDAM) when the downwarding traffic of the ABAM is sufficiently far away.

$$HKDAM(BS \Rightarrow *): K_{j+1}, MAC_{LBK}(M)$$

When the hash key K_{j+1} is disclosed, a sensor node S is able to verify the authenticity of the ABAM. A healthy node first erases the shared PWKs with the compromised nodes, and then generates a new LBK. A Secure Key Update Message SKUM is sent to each non compromised neighbor S' to share the new LBK. This message is protected with the PWK.

$$SKUM(S \rightarrow S'): NonceS, Enc_{PWK}(LBK), MAC_{PWK}(M)$$

Node S receives in turn, a Secure Acknowledgment of Key Update Message (SAKUM) from each healthy neighbor, say S', that confirms the sharing of the new LBK. Accordingly, a compromised node will not be able to interpret secure local broadcast messages.

$$SAKUM(S' \rightarrow S): NonceS', NonceS, MAC_{PWK}(M)$$

Finally, the BS proceeds by updating the GBK. KMMR guarantees the forward secrecy as nodes that had already left the network became unable to decrypt future transmitted secured messages.

3.6 Secure nodes addition

Newly added sensor nodes must be authenticated by existing nodes and vice versa. Thereafter, they can share secret keys and establish secure links with neighboring nodes. The rekeying mechanism proceeds similarly to the first deployment of the WSN. Newly added sensor nodes are preloaded with a new TK value which is generated on the fly by the BS. The new TK is distributed in the same manner as in the GBK distribution. Indeed, the distribution of the new TK is authenticated with the disclosure of the next key of the one-way key chain, whereas the encryption is made with LBKs. The BS proceeds by broadcasting an SPDM. This allows old nodes to establish secure links with new nodes, and to share with them LBKs and PWKs according to the key distribution process of Phase1 and Phase2. Thereafter, the new TKs are removed from all sensor nodes. As a final task, the BS updates the GBK as explained above. Through this WSN rejuvenation, KMMR has to perform a secure reorganization of the WSN which leads to maintaining the secure network structure. Accordingly, new candidate routers with enough energy are added securely to the WSN. Some WSN applications involve backward secrecy; this can be easily obtained by a key refresh when existing old sensor nodes receive a new SPDM. Consequently, new sensor nodes are not able to decrypt old secure messages.

4. SECURITY ANALYSIS

An attacker may achieve many types of attacks against WSNs: Eavesdropping attack, jamming attack, Sybil attack, altering messages, node capture, black hole attack, etc. Each attack has a specific goal and can be classified to be either passive or active [42], and external or internal [43]. Eavesdropping is a passive attack aiming to obtain a copy of exchanged messages. Data encryption is an efficient parade against eavesdropping. Active attacks are however much more damageable. For instance, an external aggressor may flood the network with useless traffic to paralyse the WSNs. Furthermore, if an attacker discovers secret information (cryptographic keys), it may act as a malicious internal node able to launch more harmful attacks.

The KMMR framework stands against different known attacks. Firstly, as KMMR relies on authenticated broadcast messaging, it remains difficult for an adversary to prevent the arrival of the broadcast SPDM to sensor nodes; in addition non authenticated messages are forbidden from being rebroadcasted. As such, KMMR is resilient against jamming attacks. Secondly, the replay of old messages is not allowed in KMMR due to the use of a nonce field that guarantees freshness. As a result, KMMR is resilient against replay attacks. The KMMR robustness against replay attacks is confirmed using AVISPA tools. Thirdly, KMMR inherently rejects all non authenticated messages and as such it is very resilient against sybil attacks claiming and using fake identities. Fourthly, KMMR requires a very small period (2.5 seconds for a network composed of around 1000 nodes) to distribute the necessary keys among all nodes within the network. The temporary keys are then erased from all sensor nodes memories. Furthermore, keys are not calculated, rather they are generated on the fly and exchanged by neighbor nodes. Once the TKs are erased, any compromised node can only affect its local links with the rest of the network. As such, KMMR is very resilient against node compromising. Last but not least are the black hole attacks mounted by an external node. Since KMMR permits the joining of the network only via the secure node addition, it remains resilient against this type of attacks. To test the robustness of KMMR, we used AVISPA (Automated Validation of Internet Security Protocols and Applications) [44] [45].

5. PERFORMANCE EVALUATION OF THE KMMR FRAMEWORK

KMMR is a Key Management Protocol which is intended to secure multi-hop routing in wide WSNs deployed for example in huge agriculture and cultivation fields.

5.1 Network model

KMMR is destined to manage, control and monitor huge cultivation and agriculture fields of tens of hectares. Usually such fields of fruit plants, such as olive, vine and various fruit trees, are well organized and judiciously placed in a way to leave an appropriate area for each tree. As a result, we may view such an agriculture field as a grid of cells where each cell is dedicated to a single tree as illustrated in **Fig. 1**. Without loss of generality, we regard our field to be divided into (k×k) cells. In reality the agriculture field does not usually have a perfect square shape, it suffices here to take the portion of the square that represents the actual filed.

Each tree, equivalently each cell, is controlled by one sensor node. As such, a sensor node has 8 neighbors. Such WSNs are convenient to detect fire [46] in a spacious cornfield or to control irrigation systems [47] in a large fruit-tree fields. To ensure an adequate connectivity between sensor nodes belonging to adjacent cells while limiting the signal interference, the

transmission range of a mote should be adjusted in such a way not to reach nodes beyond adjacent cells. To this end, the transmission range of a mote, r, should be regulated such that $c \leq r < 2 \times c$ where c denotes the length of a cell edge. Experiments on Telosb motes deployed in a plane area showed that the transmission range is up to about 50 meters. For c equal to 25 meters for instance and a WSN containing 961 sensor nodes, the covered area amounts to 600625 m^2 which is about 60 hectares.

As explained above, when KMMR is initiated, it performs a secure key distribution and the creation of a secure multi-tier structure. Each sensor node is assigned to a given tier t, $t \in [0,k-1]$. The BS is placed on the top left corner (tier 0) to test KMMR with the maximum number of hops, but evidently other set up are possible and do not impact our modeling. A parent link connects two sensor nodes belonging to successive tiers, while a sibling link connects two nodes belonging to the same tier.

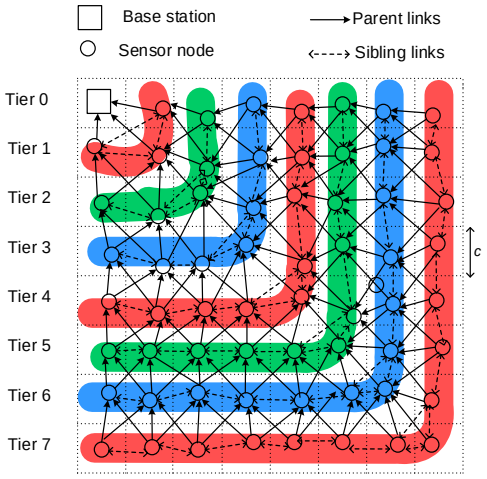

Fig. 1. Sensor Network Model

Let T_k represents the total number of nodes in a topology of k tiers according to the adopted disposition of nodes as portrayed in **Fig. 1**. We then have:

$$T_k = k^2 \tag{5}$$

and consequently for a total number of deployed nodes equal to N, the number of tiers in the topology is given by:

$$k = \lceil \sqrt{N} \rceil \tag{6}$$

Let N_p represents the number of parent links in our topology of **Fig. 1**. N_p is readily given by:

$$N_p = \sum_{t=1}^{k-1} (6t - 3) = 3(k-1)^2 \tag{7}$$

Let N_s be the total number of sibling links in the network. The number of siblings (nodes) in tier t is just 2t+1. Consequently, the total number of possible sibling links is given by:

$$N_s = \sum_{t=1}^{k-1} (2t + 1) = k^2 - 1 \qquad k \geq 2 \tag{8}$$

From **Eq. 7** and **8**, we can deduce the total number of possible secure links (parent links and sibling links) denoted by N_{sl}:

$$N_{sl} = 3(k-1)^2 + k^2 - 1 \tag{9}$$

For the (8 × 8) WSN of **Fig. 1**, we have $N_p = 147$, $N_s = 63$ and $N_{sl} = 210$. **Eq. 9** is useful as a

reference value to which we shall compare the effective number of created secure links by KMMR and then compute its key connectivity. To experiment KMMR with real sensors, we have implemented KMMR in the NesC language [48] which is fully compatible with the TinyOS distribution [49]. The security suites AES-CBC-MAC-128 and AES-CTR are used respectively for authentication and encryption [50]. The key size is set to 128 bits. The code is experimented using 16 Telosb motes [12] deployed in an olive agriculture field. Due to the limitation of the available number of Telosb motes, the same code of KMMR protocol is also tested using the TOSSIM simulator with up to 1000 sensor nodes.

5.2 Performance evaluation

Performance of KMMR is investigated through both experimentations and simulations. Let us first consider the key connectivity metric in Phase1. **Fig. 2** portrays the total number of created secure links as a function of the network size (i.e.; the total number of deployed sensors). The key connectivity of our KMMR framework is compared to the total possible secure links N_{sl} given by **Eq. 9**, and also to that obtained by the immediate broadcast of the first received SPDM in Phase1 without the collecting and randomization periods. The latter is used as a second referential value to ascertain the efficiency of the randomization and collection periods adopted in Phase1.

The middle curve in **Fig. 2** shows the effective number of created secure links by KMMR which is very close to the total number of possible secure links as given by **Eq. 9**. This clearly shows the remarkable efficiency of KMMR in establishing the secure links. Notice that when no collection and randomization periods are used, as given by the lowest curve of **Fig. 2**, the number of created secure links is much lower.

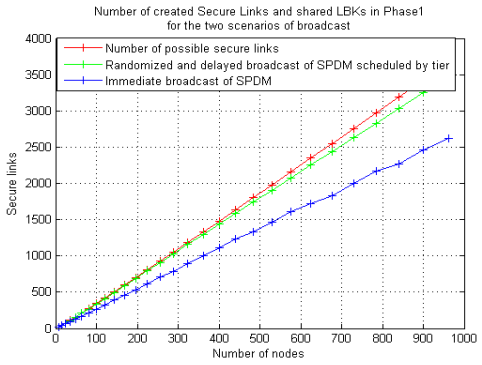

Fig. 2. Number of shared LBK for the two scenarios of broadcast

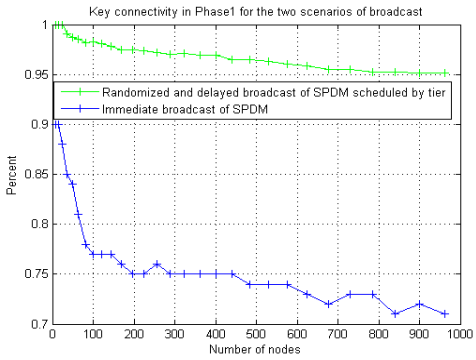

Fig. 3. LBK connectivity for the two scenarios of broadcast

Fig. 3 portrays the same information as that of **Fig. 2** but expressed in terms of key connectivity (percentage of shared LBKs). The modest key connectivity of the immediate broadcast of the first received SPDM (the lower curve) is essentially due to the important number of collisions. Our proposed KMMR provides a high key connectivity in Phase1 way above 0.95 (the upper curve) even for high values of the number of deployed sensors.

Let us now investigate the required time to achieve Phase1 for the two scenarios of broadcast: Instantaneous broadcasting and the adopted Randomized broadcasting scheduled by tier. **Fig. 4** portrays the required time to finish Phase1 as a function of the network size. For a number of deployed sensors equal to 961, the instantaneous broadcasting represented by the lower curve attains 1.2 seconds while that using the adopted scheme in KMMR (the upper

curve) attains 2 seconds. The slight increase in the finishing time of Phase1 is worth the large improvement accomplished in the key connectivity portrayed on **Fig. 3**.

On the other hand, **Fig. 5** displays the number of created tiers with each of the two scenarios of the SPDM broadcasting as a function of the total number of deployed sensors. The adopted KMMR scenario of broadcasting creates virtually the same number of tiers as the theoretical network model of **Fig. 1** given by **Eq. 6** (both curves are completely superimposed). However, the instantaneous broadcast of the first SPDM creates more tiers as any sensor node located on the diagonal may receive the first SPDM from any theoretical sibling sensor node.

Simulation results of the key connectivity in Phase2 are given by **Fig. 6**. We clearly notice that virtually all created links in Phase1 get secured in Phase2, meaning that the endpoints of the created links exchanged successfully their PWKs (both curves are virtually superimposed).

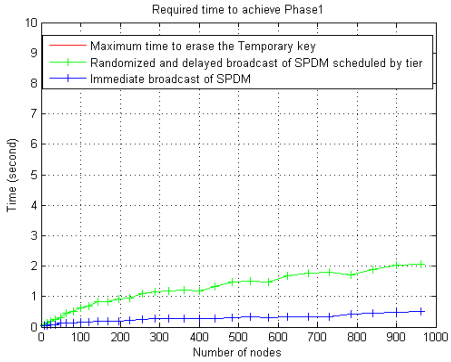

Fig. 4. Required time to achieve Phase1

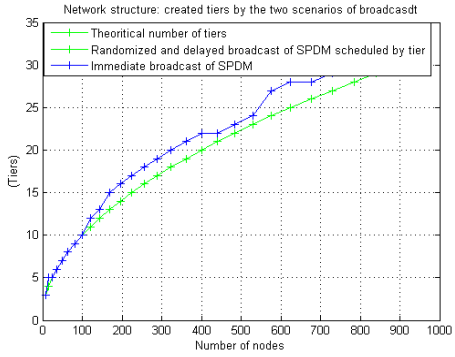

Fig. 5. Number of created tiers

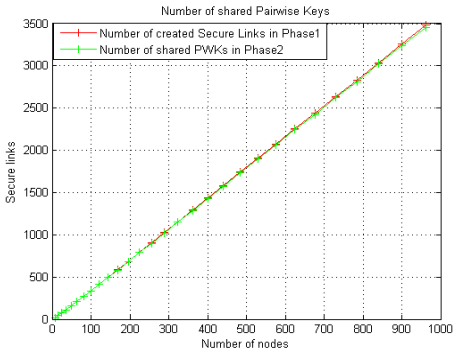

Fig. 6. Number of shared PWKs in Phase2

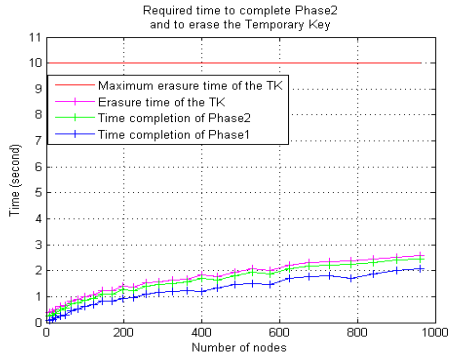

Fig. 7. Required time to complete Phase2

Now let us study the required time for the completion of Phase2 and consequently that of the erasure of the temporary keys. The three lower curve of **Fig. 7** provide the required time for KMMR to perform all operations of Phase1 and Phase2, and more importantly the time needed for the last TK erasure. From **Fig. 7**, we observe that only around 2.5 seconds are needed to attain the erasure of the last TK within a network containing 961 nodes.

Recall that the update of the Global Broadcast Key requires the broadcasting of two messages: ABM and HKDM. The first message is broadcast until reaching all leaf nodes. Collisions are not that problematic here and it is not necessary to reschedule this broadcasting. It is enough for a node to receive a copy of the ABM from any one of its neighbors. The lower curve of **Fig. 8** shows that an ABM reaches all sensor nodes in about 0.5 seconds for a network size of 961 nodes. The upper curve of **Fig. 8** shows the time to complete the GBK distribution. It does not exceed 1.4 seconds even for wide WSNs having up to 1000 nodes. Conducted

simulations showed that 100% of the deployed sensor nodes shared the GBK.

A fast node revocation process offers more effective resilience. Damages caused by malicious nodes may be limited to direct links with non-compromised neighbors. Through simulations, we have estimated the necessary time for KMMR to perform the revocation of a malicious node. In the worst case, the malicious node is located on the extremity of the sensor network. Such a position provides the longest multi-hop path between the IDS server and the compromised node.

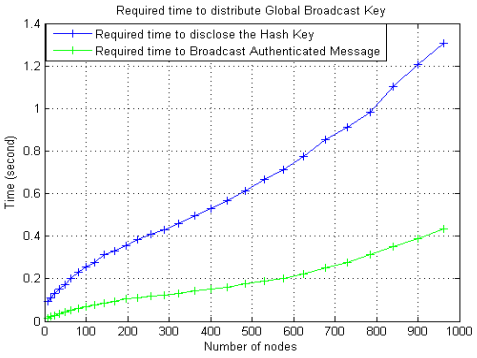

Fig. 8. Required time for GBK distribution

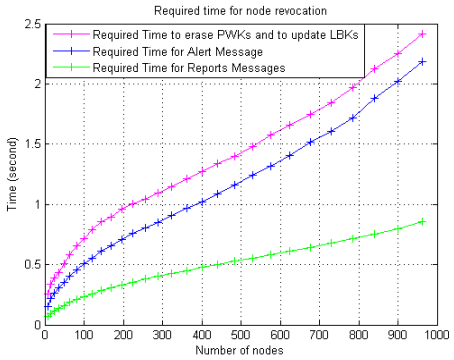

Fig. 9. Required time for node revocation

Simulation results of node revocation are portrayed on **Fig. 9**. Secure Report Messages require less than one second to reach the BS for a network size of 961 nodes. The middle curve represents the time needed for a Secure Alarm Message to reach the entire network. The top most curve shows that the erasure of the PWK from all non-compromised nodes and the updating of their LBKs are done in less than 2.5 seconds. The required global time, which is the sum of the required time for node revocation and the required time for updating the GBKs, does not exceed 4 seconds. Recall that the necessary time (T_{min}) to reveal information from the memory of a captured mote is 10 seconds [31]. As such, KMMR provides a good resilience in wide WSNs having up to 1000 sensor nodes. Damages are limited to direct links with the compromised node. KMMR leaves then around 6 seconds to the IDS server to analyze Alarm Messages and detect the existence of a compromised node.

In case of simultaneous compromising attacks, the IDS server will surely take more time to make decisions. This does not considerably affect the time taken by KMMR to revoke compromised nodes as it is able to send several nodeID of malicious nodes in a one Secure Alarm Message by using the adopted black-list technique.

Secure node addition in KMMR uses basically the same keying mechanism as that of the first deployment of the WSN.

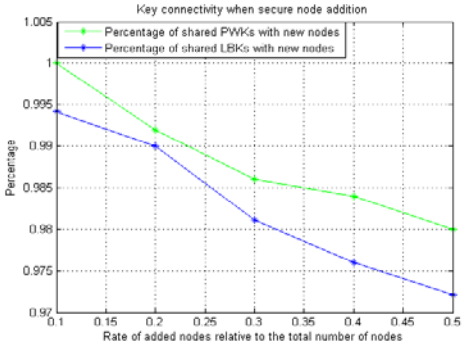

Fig. 10. Key connectivity with secure node addition (network size 961 nodes)

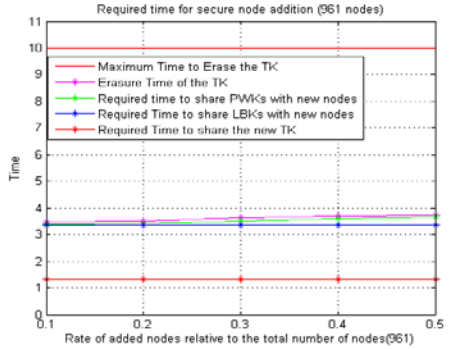

Fig. 11. Required time for secure node addition (nework size 961 nodes)

New nodes are pre-loaded with new Temporary Keys. The only difference is the required time to distribute the TKs to existing nodes which is almost the same time taken to distribute the GBK. Simulations results given in **Fig. 8** showed that the required time is less than 1.5 seconds for the large considered WSN of 961 nodes. To investigate the scalability of the secure node addition in KMMR, simulations tests are performed with 961 sensor nodes. The results are depicted on **Fig. 10** as a function of the fraction of the number of new nodes among the 961 nodes.

The lower curve of **Fig. 10** shows the key connectivity of LBKs reaching more than 99% when the fraction of added node is between 10% and 20%. Even, when the fraction of added nodes increases to 40% and 50%, the percentage of shared LBKs remains between 97% and 98%. The percentage of the shared PWKs, depicted by the upper curve, levels between 98% and 100% for all tested fractions of added sensor nodes. While KMMR provides an excellent key connectivity for secure node additions, it remains crucial to verify the time required to achieve the various operations involved in the secure node addition. This is provided in **Fig. 11.** We observe that KMMR achieves all the operations of the secure node addition in less than 4 seconds for our WSN comprising 961 nodes.

6. COMPARISON WITH EARLIER WORK

KMMR provides efficient keying mechanisms to secure multi-hop routing in wide WSNs and maintain an appropriate trade-off between robustness and cost. The cost is expressed in terms of required storage space, communication overhead and computational complexity.

6.1 Storage overhead

To join the network in KMMR, a node acquires a Temporary Key and a commitment Key from the one-way key chain. The symmetric cryptographic algorithm AES uses keys with 16 bytes long, only 32 bytes are then needed. Compared to the random key predistribution schemes [18], KMMR requires a much smaller storage space. Indeed, for the random key pre-distribution schemes, the required space in a sensor node increases considerably with the network size. By against in KMMR, the number of stored keys remains the same independently of the network size. **Fig. 12** depicts the number of keys required to be stored as a function of the network size. We observe that our proposed KMMR outperforms by far the random key pre-distribution schemes for network sizes above 150 nodes.

A row in the neighbors table requires at most 40 bytes (nodeID, Tier, lastNonce, LBK, PWK). Given 8 neighbors per node, the size of the neighbors table is then about 320 Bytes. Knowing that a Telosb mote provides 10 Kb of RAM, the required space needed by KMMR to store the neighbors table as well as all keys amounts to less than 5% of the whole space. As KMMR supports node revocation, a black-list may be stored in the flash memory of a sensor node. Knowing that an ID of a Telosb mote takes 2 bytes, one kilo bytes of flash-memory permits to store a black-list that may contain up to 512 malicious nodes. Telosb mote contains 48Kb of flash memory, and therefore KMMR requires only a very small memory space.

6.2 Communication overhead

KMMR uses several types of messages to ensure the distribution of the keys needed for the creation of the secure network structure, the secure nodes addition, the node revocation and the keys updating. Messages may include many fields such as address fields, key value, nonce, time stamp, tier field and MAC field. The size of the used messages varies between 8 and 32 bytes. According to studies on Telosb and Micaz [16], communication is the most expensive

operation in terms of energy consumption. We here compare the number of exchanged messages in KMMR to share LBKs and PWKs with those of both LEAP [28] and SNEP [26].

For KMMR, each node has to share its LBK with its neighbor nodes by broadcasting a SPDM. For a WSN comprising N nodes, N messages are then sent during the distribution of the LBKs. By against, in LEAP a node needs to send a message for each neighbor to share with it the cluster key. Recall that a LBK and a cluster key have around the same payload. **Fig. 13** depicts the number of exchanged messages by KMMR and LEAP as a function of the network size. We clearly observe that over all network sizes, KMMR requires much less communication overhead than LEAP.

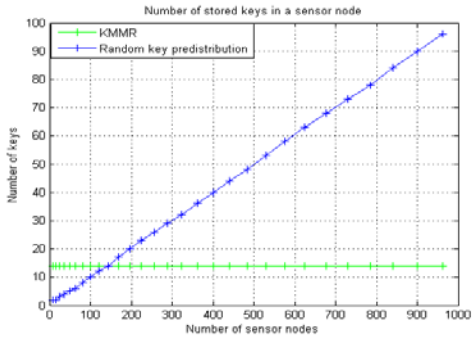

Fig. 12. Stored keys in a sensor node

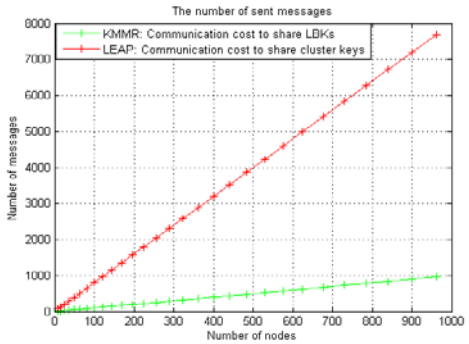

Fig. 13. Number of sent messages by KMMR to share LBKs and by LEAP to share cluster keys

To share PWKs in KMMR, each node requires to send a SJRM to each of its router (parent or sibling node) and receive a SRRM in response. By against, in LEAP, each node has to broadcast a Hello Message and receive an Acknowledgement Message from each one of its neighbors. Assuming the same topology, the number of routers in KMMR including only neighbors on the same and the previous tiers is much less then the total number of neighbors and as such the number of exchanged messages to establish all PWKs in LEAP is much higher than that in KMMR.

SNEP considers the BS as the Key Distribution Center responsible for sharing session keys (Pairwise Keys) between any two neighbor nodes. As a result, in SNEP a node at tier *t* requires to send *2t* messages to share its PWK with a parent node, and *2t + 1* messages to share its LBK with a sibling node. Using **Eq.7** and **8**, the total number of sent messages to share all possible session keys is readily given by:

$$\sum_{t=1}^{k-1} 2t \times (6t-3) + \sum_{t=1}^{k-1} (2t+1) \times (2t+1) = \sum_{t=1}^{k-1} 16t^2 - 2t + 1$$

KMMR, on the other hand, uses a local process to share the PWKs. As a result, the number of messages sent by SNEP to share PWKs is much higher than that required by KMMR.

6.3 Computational cost

KMMR uses low cost symmetric encryption/decryption algorithms to achieve key distribution, node revocation, secure node addition and keys updates. We here estimate the number of encryption/decryption operations in KMMR and then compare it to that of LEAP.

Fist of all, let us consider the sharing of the LBKs. For KMMR, a sensor node needs to generate on the fly its LBK and encrypts it with its TK. The sensor node further needs to decrypt the LBK received from each neighbor. The number of such a decryption is then equal to the number of neighbors, say *d*. For LEAP, the cluster key distribution is protected with the

PWK. Therefore, each sensor node needs d encryptions to share the cluster key with its neighbors and also d decryptions accomplished by the neighbors. As a result, KMMR incurs much less computation overhead than LEAP.

Now let us consider the sharing of the PWKs. In KMMR, a sensor node should generate and encrypt the PWKs to be shared with its routers. Thereafter, each router node decrypts the received PWKs. The exchange of the PWKs is protected by the TK. If d is the number of neighbor nodes, KMMR needs $2d$ encryption/decryption operations to share the PWKs. For LEAP, a sensor node does not make encryption/decryption operations to share PWK. Rather a PWK is computed by a Hash function which uses the values of the initial key and the node ID. A priori, LEAP generates a low computational cost to share PWKs; but this presents a breach of security in LEAP. Hence, if an adversary reveals the initial key, it may derive any PWK; thereafter the resilience of the entire network is stalled.

Furthermore, in KMMR the BS proceeds with a global broadcast to update the GBK which is authenticated with the next hash key (taken from the one-way key chain) and protected by the LBKs. Therefore, each node has to perform one decryption to know the new value of the GBK and one encryption to protect it with its proper LBK. As such, for a WSN comprising N nodes, the update of the GBK requires just $2(N-1)$ encryption/decryption operations. On the other hand, in a node revocation, each neighbor deletes the PWKs shared with the compromised node and updates its LBK. The update of the LBKs is protected by the PWKs shared between non-compromised nodes. If d is the number of healthy neighbors, each neighbor has to perform $(d-1)$ encryptions and $(d-1)$ decryptions. As a result, the total number of encryption/decryption operations when a node is revoked is $2(d-1).(d-1) = 2(d-1)^2$. KMMR incurs less storage/communication/computation overheads than earlier work such as SNEP, Random Key Pre-distribution schemes and LEAP.

Table 1 provides a qualitative comparison between the different considered key management schemes in terms of communication, storage and computing overheads, as well as in terms of scalability, resilience, and key connectivities (pairwise, local and global). A (+) sign reflects a complete fulfilment of the corresponding quality measure, a minus sign means that the corresponding quality measure is either not fulfilled or poorly satisfied, and a (+/-) reflects a certain degree of fulfilment. KMMR is the only scheme that requires low communication, storage and computing overheads.

Table 1. Summary of qualitative comparison

Quality Measure ---------- Key distribution	Low Comm. overhead	Low Comp. overhead	Low Storage overhead	Scalability	Resilience	Pairwise key Connectivity	Local key Connectivity	Global key Connectivity
KDC-based key pre distribution	-	+	+	-	+	+	-	-
µTESLA	+/-	+/-	+	+	+	-	-	+
STKM	+/-	+/-	+/-	-	-	+	-	-
LEAP+	+	+/-	+	+	-	+	+	+
R-LEAP	+/-	+/-	+/-	+	+/-	+	+	+
random key pre distribution	+/-	+	+/-	+/-	+/-	+	-	-
KMMR	+	+	+	+	+	+	+	+

Table 2 provides a quick recall to the definition of the acronyms used throughout the paper.

Table 2. Summary of notations

Notation	Description	Notation	Description
BS	Base Station	SPDM	Secure Path Discovery Message
KMP	Key Management Protocol	SJRM	Secure Join Router Message
TK	Temporary Key	SRRM	Secure Router Response Message
LBK	Local Broadcast Key	ABM	Authenticated Broadcast Message
GBK	Global Broadcast Key	HKDM	Hash Key Disclosure Message
PWK	Pairwise Key	ABAM	Authenticated Broadcast Alarm Message
H()	One-way hash function	HKDAM	Hash Key Disclosure of Alarm Message
MAC	Message Authentication Code	SKUM	Secure Key Update Message
EncK(M)	Encrypting of message M with symmetric key K	SAKUM	Secure Ack of Key Update Message

7. CONCLUSIONS

KMMR performs key distribution, node revocation, secure node addition and key updates in an efficient way by using local lightweight processes organized into tiers. KMMR takes advantage of this multi-tier organization to schedule different types of broadcasts allowing a significant reduction of collisions, enhancement of key connectivity, significant rapidity in the achievement of many protocol operations, and hence energy saving. KMMR ensures node revocation by erasing all shared keys with a compromised node. It performs secure node addition without any restriction on the number of added nodes or their positions. Through an efficient key updating, KMMR ensures forward secrecy as well as backward secrecy. Security analysis clearly showed the resistance of KMMR against most known attacks such as Jamming attacks, Replay attacks, Sybil attacks, Node compromising and Black hole attacks. The robustness against MiM and replaying attacks is validated using the AVISPA tools. Experimentation results show that KMMR is scalable and efficient. Compared to other Key Management Protocols, KMMR is computationally more efficient, and requires lower storage and lower communication overheads. We are currently investigating the use of the KMMR as a framework to secure multi-hop routing and clustering of large scale WSNs.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, 38(4):393 – 422, 2002. Article (CrossRef Link)

[2] C. Karlof and D. Wagner, "Secure routing in WSN: attacks and countermeasures," *Ad Hoc Networks*, 1(23):293 – 315, 2003. Article (CrossRef Link)

[3] Y. Zhou, Y. Fang, and Y. Zhang, "Securing wireless sensor networks: a survey," *Communications Surveys Tutorials IEEE*, 10(3):6 – 28, 2008. Article (CrossRef Link)

[4] Q. Yang, X. Zhu, H. Fu, and X. Che, "Survey of security technologies on wireless sensor networks," *Journal of Sensors*, 2015:842392:1–842392:9, 2015. Article (CrossRef Link)

[5] J. Zhang and V. Varadharajan, "Wireless sensor network key management survey and taxonomy," *JNCA*, Elsevier, 33(2):63–75, 2010. Article (CrossRef Link)

[6] M.A. Simplicio Jr., P.S. L.M. Barreto, C.B. Margi, and T.C.M. B. Carvalho, "A survey on key management mechanisms for distributed wireless sensor networks," *Computer Networks*, 54(15):2591–2612, 2010. Article (CrossRef Link)

[7] T. Bonaci, et. al., "Node capture attacks in wireless sensor networks: A system theoretic approach," *CDC2010*, Atlanta, USA, pp. 6765–6772, 2010. Article (CrossRef Link)

[8] H. Maddar, A. Trad, A. Guermazi, and S. Ben Othman, "Secopp+: A secure dynamic scheme for adding new nodes in secopp protocol," *WSCAR'14*, pages 1–5, Jan 2014. Article (CrossRef Link)

[9] M. Pérez-Ruiz, J. Agüera, J.A. Gil, et al., "Optimization of agrochemical application in olive groves based on positioning sensor," *Precision Agriculture (Springer)* 12(4): 564-575, 2011. Article (CrossRef Link)
[10] M.H. Anisi, G. Abdul-Salaam, A.H. Abdullah, "A survey of wireless sensor network approaches and their energy consumption for monitoring farm fields in precision agriculture," *Precision Agriculture (Springer)* 16(2): 216-238, 2015. Article (CrossRef Link)
[11] L. Pan, V.I. Adamchuk, D.L. Martin, et al., "Analysis of soil water availability by integrating spatial and temporal sensor-based data," *Precision Agriculture (Springer)* 14(4): 414-433, 2013. Article (CrossRef Link)
[12] Inc. CrossbowTechnology. Telosb mote platform.
[13] R.D. Pietro, et al., "Connectivity properties of secure wireless sensor networks," *SASN'04*, pages 53–58, 2004. Article (CrossRef Link)
[14] A. Gupta and J. Kuri, "Deterministic schemes for key distribution in wireless sensor networks," *COMSWARE 2008*, January 5-10, 2008, Bangalore, India, pages 452–459, 2008. Article (CrossRef Link)
[15] H. Alzaid, D. Park, J.M.G. Nieto, C. Boyd and E. Foo, "A forward and backward secure key management in wireless sensor networks for PCS/SCADA.," *S-CUBE 2009*, Pisa, Italy, September 7-9, 2010, pages 66–82, 2009. Article (CrossRef Link)
[16] G. DeMeulenaer, F. Gosset, F.X. Standaert, and O. Pereira, "On the energy cost of communication and cryptography in wireless sensor networks," *WiMob 2008*, Avignon, France, 12-14 October 2008, Proceedings, pages 580–585, 2008. Article (CrossRef Link)
[17] A. Shamir R. Rivest and L. Adlemani, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, 21(2):120–126, 1978. Article (CrossRef Link)
[18] A. Liu and P. Ning, "Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks," *IPSN'08*, USA, pages 245–256, 2008. Article (CrossRef Link)
[19] X. Zhang, J. He and Q, "Wei. EDDK: energy-efficient distributed deterministic key management for wireless sensor networks," *EURASIP JWCN*, 2011. Article (CrossRef Link)
[20] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: Link Layer Security Architecture for wireless sensor networks," *SenSys 2004*, Baltimore, MD, USA, November 3-5, pages 162– 175, 2004. Article (CrossRef Link)
[21] L. Eschenauer and V.D. Gligor, "A key-management scheme for distributed sensor networks," *CCS 2002*, Washington, DC, USA, November 18-22, pages 41–47, 2002. Article (CrossRef Link)
[22] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," *CCS 2003*, Washington, DC, USA, pages 52–61, 2003. Article (CrossRef Link)
[23] D. Liu and P. Ning, "Location-based pairwise key establishments for static sensor networks," *SASN '03*, pages 72–82, 2003. Article (CrossRef Link)
[24] N.T.T. Huyen, M. Jo, T.D.Nguyen and E.N. Huh, "A Beneficial Analysis of Deployment Knowledge for Key Distribution in Wireless Sensor Networks," *Security and Communication Networks*, 5(5): 485-495, May 2012. Article (CrossRef Link)
[25] P. Traynor, et al. "Establishing pairwise keys in heterogeneous sensor networks," *INFOCOM 2006*, pp. 1-12, 2006, Barcelona, Spain, 2006. Article (CrossRef Link)
[26] A. Perrig, et. Al., "SPINS: security protocols for sensor networks," *Wireless Networks*, 8(5): 521-534, 2002. Article (CrossRef Link)
[27] A. Perrig, R. Canetti, D.X. Song, and J. D. Tygar, " Efficient and secure source authentication for multicast," *NDSS 2001*, San Diego, California, USA, 2001.
[28] S. Zhu, S. Setia, and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks," *CCS 2003*, Washington, DC, USA, October 27-30, pages 62–72, 2003. Article (CrossRef Link)
[29] S. Zhu, S. Setia, and S. Jajodia, "Leap+: Efficient security mechanisms for large-scale distributed sensor networks," *TOSN*, 2(4):500–528, 2006. Article (CrossRef Link)
[30] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, 24(11):770–772, 1981. Article (CrossRef Link)

[31] J. Deng, C. Hartung, R. Han, and S. Mishra, "A practical study of transitory master key establishment for wireless sensor networks," *SecureComm 2005*, pages 289– 302, 2005. Article (CrossRef Link)
[32] M.L. Messai, M. Aliouat, and H. Seba, "Tree based protocol for key management in wireless sensor networks," *EURASIP JWCN*, 910695, 2010. Article (CrossRef Link)
[33] S. Blackshear and R.M. Verma, "R-leap+: Randomizing leap+ key distribution to resist replay and jamming attacks," in *Proc. of SAC '10*, pages 1985–1992, New York, NY, USA, 2010. Article (CrossRef Link)
[34] R. Geetha and E. Kannan, "A hybrid key management approach for secure communication in wireless sensor networks," *Indian Journal of Science and Technology*, 8(5): 1-8, 2015. Article (CrossRef Link)
[35] E.B. Barker and J.M. Kelsey, "Recommendation for random number generation using deterministic random bit generators," *TC, NIST*, USA, 2012. Article (CrossRef Link)
[36] D.E. Eastlake, J.I. Schiller, and S. Crocker, "Randomness requirements for security," *BCP 106*, RFC 4086, 2005.
[37] C. Intanagonwiwat, R. Govindan, D. Estrin, J.S. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans*, Netw., 11(1): 2–16, 2003. Article (CrossRef Link)
[38] W.B. Heinzelman, et al., "An application-specific protocol architecture for wireless micro sensor networks," *IEEE TWC* 1(4): 660– 670, 2002. Article (CrossRef Link)
[39]] X. Liu, "A survey on clustering routing protocols in wireless sensor networks," *Sensors*, 12(8): 11113-11153, 2012. Article (CrossRef Link)
[40] 802.15.4-2006-IEEE standard for information technology– local and metropolitan area networks– specific requirements– part 15.4: MAC and PHY specifications for low rate wireless personal area networks (wpans). IEEE Xplore, 2006. Article (CrossRef Link)
[41] N.A. Alrajeh, et. al., "Intrusion detection systems in wireless sensor networks: A review," *Int. J. Distributed Sensor Networks*, Vol. 2013, 7 pages, 2013. Article (CrossRef Link)
[42] M.M. Patel and A. Aggarwal, "A survey of attacks, security mechanisms and challenges in wireless sensor networks," in *Proc. of ISSP*, March 2013. Article (CrossRef Link)
[43] X. Huang, et al., "Effective algorithm for protecting WSNs from internal attacks in real-time," *ACSW'16*, ACM, New York, NY, USA, 2016. Article (CrossRef Link)
[44] The AVISPA Team. Hlpsl tutorial. http://www.avispa-project.org, 2006.
[45] The AVISPA Team. Automated validation of internet security protocolds an applications user manual. http://www.avispa-project.org, 2006.
[46] J. Lloret, et al., "A wireless sensor network deployment for rural and forest fire detection and verification," *Sensors*, 9(11): 8722-8747, 2009. Article (CrossRef Link)
[47] G. Bitella, et al., "A novel low-cost open-hardware platform for soil water content and multiple soil-air-vegetation," *Sensors*, 14(10): 19639-19659, 2014. Article (CrossRef Link)
[48] Nesc: A prog. language for deeply networked systems. http://nescc.sourceforge.net/, 2007.
[49] Tinyos. http://www.tinyos.net/, 2012.
[50] N. Sastry and D. Wagner, "Security considerations for IEEE 802.15.4 networks," *WiSe '04*, pages 32–42, 2004. Article (CrossRef Link)
[51] I. Jemili, A. Belghith and M. Mosbah, "A Synchronous Tiered Based Clustering Algorithm for large-scale Ad hoc Networks," *Wireless and Mobile Networking*, Vol. 284, pp. 41-55, 2008. Article (CrossRef Link)
[52] T. Gazdar, A. BenSlimane and A. Belghith, "Secure clustering scheme based keys management in VANETs," in *Proc. of 73rd IEEE Vehicular Technology Conference (VTC Spring 2011)*, Budapest, Hungry, 2011. Article (CrossRef Link)
[53] W. Akkari, B. Bouhdid and A. Belghith, "LEATCH: Low energy adaptive tier clustering hierarchy," *Elsevier, Procedia Computer Science*, Vol. 52, pp. 365-372, 2015. Article (CrossRef Link)

Abderrahmen Guermazi is currently a Technologist Professor in computer science department, at the Higher Institute of Technological Studies (Institut Supérieur des Etudes Technologiques, ISET) of Sfax –Tunisia. He obtained National Aggregation Degree in Computer Science in 1998. He is the founder and founding and the current responsible of the Professional Master Degree "Development of Computer Systems and Networks" at ISET Sfax. Since 2008, He has been an active member of the Computer and Embedded System Laboratory at the National School of Engineers of Sfax (ENIS). He contributed in he organization of several workshops and conferences. His research and teaching interests focus on Wireless Sensor Networks, Routing and Security. He has several publications in refereed international conferences and peer reviewed journal.

Dr. Abdelfettah Belghith received his Master of Science and his PhD degrees in computer science from the University of California at Los Angeles (UCLA) respectively in 1982 and 1987. He is since 1992 a full Professor at the National School of Computer Sciences (ENSI), University of Manouba, Tunisia. He is currently on a sabbatical leave at King Saud University, Saudi Arabia. His research interests include computer networks, wireless networks, multimedia Internet, mobile computing, distributed algorithms, simulation and performance evaluation. He runs several research projects in cooperation with other universities, research laboratories and research institutions. He is the Past chair of the IEEE Tunisia section, the chair of the IEEE ComSoc and VTS Tunisia Chapters, and the Director of the HANA Research Laboratory (www.hanalab.org) at the National School of Computer Sciences. He published more than 300 research papers in international journals and conference proceedings.

Mohamed ABID is currently a Professor at the Engineering National School of Sfax (ENIS), University of Sfax, Tunisia. He received the Ph. D. degree from the National Institute of Applied Sciences, Toulouse (France) in 1989 and "thèse d'état" from the National School of Engineering of Tunis (Tunisia) in 2000 in Computer Engineering & Microelectronics. His current research interests include hard/soft co-design, System on Chips, Reconfigurable and Embedded Systems. Currently, he is the director of the Computer Embedded System laboratory CES-ENIS, (http://www.ceslab.org). He is a founding member and responsible of doctoral program at ENIS, 2003-2010. He served in national and international conference organization and program committees. He also served as a co-editor of several Special Issues in peer reviewed International Journals. He coordinated and participated in several international research and innovation projects. He supervised more than 20 PhDs and 50 Masters. He published more than 220 book chapters and research papers in international journals and conference proceedings. Dr. Abid has also served as Guest Professor at several international universities and as a Consultant to research & development.

Dr. Sofien Gannouni received his Master degree in Computer Science from Paul Sabatier University (Toulouse III - France), and his PhD degree in Computer Science from Pierre & Marie Curie University (Paris VI - France). Currently, he is an Assistant Professor at the Computer Science Department, the College of Computer and Information Sciences, King Saud University. His main research interests include service-oriented computing, distributed computing, parallel processing, middleware, sensor networks applied to health care monitoring systems, and brain computer interface systems. He published more than 40 research papers in international peer reviewed journals and refereed international conference proceedings.