# Crypto-Security Contribution in WSNs

Nourreddine Mitta
Laboratory of Electrical Genius & Energy Systems Ibntofail University-Faculty of Science –Kénitra-Morocco
Noureddine_mitta@yahoo.fr

Rachid Gouri, Hlou Lamari
Laboratory of Electrical Genius & Energy Systems Ibntofail University-Faculty of Science–Kénitra-Morocco
Elgouri.rachid@yahoo.fr

Brahim Moalige
Vice-President of Moroccan Cryptography Association
Ibntofail University-Faculty of Science-Kénitra-Morrocco
Bmoalige@hotmail.coml

***Abstract*—Securing a wireless communication has generally a vital importance, particularly when this communication is in a hostile environment like in wireless sensor networks (WSNs). The problem is how to create cryptographic keys between sensor nodes to ensure secure communications. Limited resources of sensor nodes make a public key cryptosystem such as RSA not feasible. So, most solutions rely on a symmetric cryptosystem. In this paper, we propose a new key mana-gement scheme based on symmetric cryptography which is well adapted to the specific properties of WSNs. The evalu-ation of our solution shows that it minimizes memory occupation, ensures scalability, and resists against the hardest attack: com-promised nodes.**

***General Terms*—Network Security.**

***Index Terms*—WSN, PDKR, Sink, TinyPK, Tiny ECC, SKNP, SPINS, LEAP.**

## I Introduction

The convergence of technological advances in micro-electronics and wireless communications has enabled the emergence of a promising area: Wireless Sensors Networks (WSNs). WSNs come from the combination of embedded systems and distributed systems. WSNs have opened the way for a multitude of research areas and the huge interest generated by researchers activities calls for broad fields of applications in the near future.

Sensors appear as miniaturized systems, equipped with a processing unit and storage of data, a unit of wireless transmission and a battery. Organized as a network, the sensors (or nodes) of a WSN, despite resource constraints in computing capacity, storage, and energy, have to play an essential role in quasi all domains of human environment. They are primarily dedicated to collect data from physical phenomena such as monitoring global warming and send them to a base station (also called the sink) [1]. Figure 1 show an example of a WSN composed of ten sensor nodes deployed randomly around a base station. Depending on the size of the deployment area, the transmission range of the sensor nodes, and the base station, sensor nodes can communicate with the base station directly or indirectly by computing a hop-by-hop route to it. Many barriers to the common deployment of WSNs have to be overcome before they can reach their full maturity. Among these obstacles, the security problem is acute and must be addressed adequately and in accordance with the binding characteristics of WSNs. Because of their constraints and their deployment in unattended and hostile environments, the different nodes of a WSN are vulnerable to node compro-mising and also to physical damage [2]. In addition, the use of wireless transmission makes WSNs permeable to all sorts of malicious attacks. Consequently, security is a real challenge to rise.

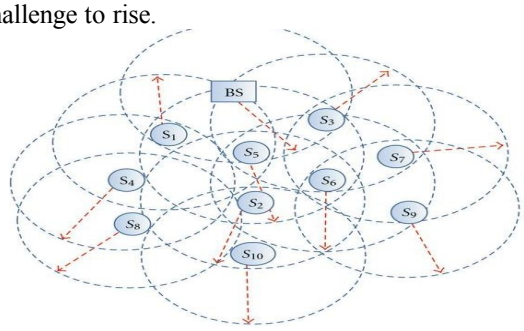


Figure 1 Example of WSN

Several key management protocols for WSNs [3-4-5-6-7-8] were proposed to respond to the security requirements of these environments. Unfortunately, node compromising is rarely or not enough investigated and most of these protocols have a weak resilience to this attack. In this paper, we present a symmetric-based key management solution for WSNs called PDKR (PROTOCOL DIAGRAM KEY RECOVERY SOLUTION for WSNs). PDKR is a simple and robust solution to secure node-to-node and node-to-base station communi-cations. PDKR assumes a random deployment


doi:10.4304/jetwi.6.1.32-39

of nodes. It builds a diagram that spans all the sensor nodes. This diagram allows key refresh with small costs. Simulation results show that PDKR is very resilient to node compromise while preserving energy consumption at the level of sensor nodes.

The rest of the paper is organized as follows. First, we discuss related work in Section 2. In Section 3, we present our solution and give its detailed algorithms. Section 4 is devoted to an analysis and a simulation of the proposed solution. Section 5 concludes our work.

## II Background and Motivation

Key management is the process by which cryptographic keys are generated, stored, protected, transferred, loaded, used, and destroyed [3]. Figure 2 lists a selection of existing key mana-gement solutions proposed for WSNs in the literature, for a detailed state of the art see [3-4]. Most existing key mana-gement solutions are based on symmetric cryptography mainly because of its reasonable energy consumption. Asymmetric cryptography involves the use of a pair of keys (public key and private key) to encrypt and decrypt messages. Each node in the network has a public and a private key, the first is known throughout the network, the second is secret, that is, known only by the node. The source node encrypts messa-ges using the public key of the destination node, and this latter uses its private key to decrypt received messages. In symmetric cryptography, the source and the destination use the same key to encrypt and decrypt messages. Asymmetric cryptography offers better resistance against node compromise attack and allows scalability but requires an additional part on software and hardware of the nodes. Some researchers investigated asymmetric cryptographic tools and propose adapted solutions. Examples of such solutions are Tiny Public Key (TinyPK) [9] and Tiny Elliptic Curve Cryptosystem (Tiny ECC) [10].

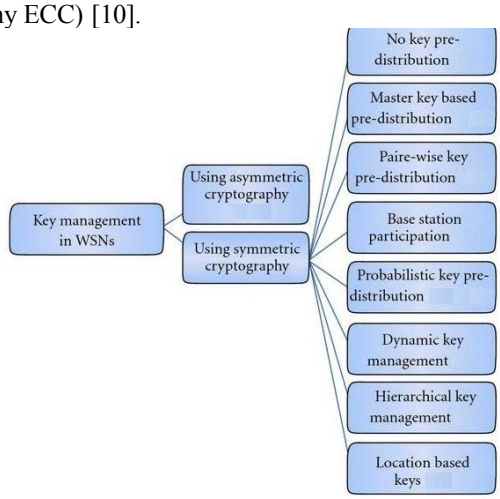


Figure 2 Some Existing key management solutions for WSNs

With symmetric cryptography, the simplest idea is to load secret information in the sensor nodes before their deployment in the network. This secret information deployed in the network may be the secret key itself or auxiliary information that helps nodes to derive the real secret key shared by the nodes. With this secret key, nodes can securely exchange messages [3]. The main disadvantage of this solution is that compromising one node (access to the pre-loaded key) might lead to compromise the entire network. To overcome this limitation, several researchers propose schemes that establish pairwise keys rather than a unique global key. For example in [11], the authors focus on developing cost-saving mechanisms while weakening the threat model. They propose Key Infection, a lightweight security protocol suitable for use in noncritical commodity sensor networks where an attacker can monitor only a fixed percentage $a$ of communication channels. With Key Infection, a node wishing to communicate securely with other nodes simply generates a symmetric key [4] and sends it in the clear to its neighbors.

In [12], Blon describes an optimal class of symmetric key generation systems solution. In this solution, some of the possible link keys in a network of size $N$ are represented as a $(\lambda+1) \times N$ key matrix. The scheme stores small amount of information in each sensor node, so that some pair of nodes can calculate corresponding field of the matrix, and uses it as the link key. This solution is $\lambda$-*secure*, meaning that keys are secure if no more than $\lambda$ nodes are compromised. Another $\lambda$-*secure* solution is presented in [13] and called *Polynomial-based key pre-distribution scheme*. This scheme distributes a polynomial share (a partially evaluated polynomial of degree$\lambda$) to each sensor. So, each sensor node stores a polynomial with $(\lambda+1)$ coefficients and every pair of sensor nodes can establish a key using the property of symmetry of polynomials. The solution is $\lambda$-*secure*, meaning that coalition of less than $\lambda+1$sensor nodes knows nothing about pairwise keys of others.

In [14], the authors propose *BROSK* (BROadcast Session Key negotiation protocol). With *BROSK* every node broadcasts a message containing its nonce. So, every two neighbouring nodes that hear each other can compute a common key which is function of their two nonces. Neighbouring nodes authenticate themselves with a pre-deployed key which is supposed to be unreachable in the case the node is captured. In [15], the authors propose a variation of this protocol where the pre-deployed key is used only for a restricted period of time during which nodes establish pairwise keys. Then, the pre-deployed key is erased. However, ***HELLO*** messages used to establish pairwise keys are sent in the clear. So, an attacker that captures a node and also eaves drops ***HELLO*** messages can use the IDs and nonces contained in these messages to derive established keys.

Perrig et al. propose in [16] *SPINS*, a key management protocol that relies on a trusted base station to distribute keys. SPINS contains two parts: *SNEP* (Secure Network Encryption Protocol) that protects communications between a node and the base station or between two nodes, and $\mu$ ***TESLA*** (micro time efficient streaming loss-tolerant authentication) that serves to authenticate packets coming from the base station. The first part is unsuitable to energy constraint of nodes because any

communication between two nodes must pass through the base station. The second part needs additional memory space to store authentication keys. In [17], the authors propose *LEAP* (Localized Encryption and Authentication Protocol); a key management protocol intended to support a several communication patterns. In this protocol, each node stores four types of keys: individual, pairwise, cluster, and group. An individual key is a key shared between a node and the base station. A pairwise key is shared between a node and each of its neighbours.

A cluster key is a key shared between a node and all neighbouring nodes. A group key is a key common to the entire network. The individual key is preloaded. After deployment, neighbouring nodes establish pairwise keys. They authenticate themselves using a pre-deployed key which is erased as soon as pairwise keys are established. To establish cluster keys and the group key, nodes use broadcasts and message relaying. The protocol uses ***μ Tesla*** [16] to authenticate broadcasts.

Liu et al. propose in [18] *LBKs* (location-based keys) that relies on location information to achieve key management. The keys are established according to the geographical location of sensor nodes. However, knowing the geographical location of nodes is not guaranteed with random deployment. Eschenauer and Gligor [19] propose a scheme based on a random key pre-distribution. In this scheme, each sensor randomly picks a set of keys and their identifiers from a key pool before deployment. Then, a shared-key discovery phase is launched where two neighbours exchange and compare list of identities of keys in their key chains. Basically, each sensor node broadcasts one message and receives one message from each node within its radio range where messages carry key ID lists. So, any pair of nodes has a certain probability to share at least one common key. The challenge of this scheme is to find a good trade off between the size of the key pool and the number of keys stored by nodes to achieve the best probability. The main drawback of this approach is that if the number of compromised nodes increases, the fraction of affected links also increases. Other solutions use the principle of probabilistic key pre-distribution [21, 22] introduced in [19], For example, the authors of [21] suppose that the deployment area is a grid-based structure of $t*n$ cells called groups. Groups contain the same number of sensor nodes. The protocol uses $t*n$ key pools such that neighbouring key pools have more keys in common. Sensor nodes are deployed with the key pool that corresponds to their group in the deployment area. After deployment, nodes sharing keys can communicate directly. Nodes that do not share keys must establish a path key using their neighbours. In [20] the authors propose to increase the amount of key overlap required in the shared-key discovery phase. Their scheme called ***Q-****composite* requires $Q$ common keys to establish a link key. Link between a pair of sensor nodes is set as a hash of all common keys. The scheme improves resilience because the probability that a link is com-promised, when a sensor node is captured, decreases, but probability of key sharing also decreases because a pair of nodes has to share $Q$ keys instead of one.

Eltoweissy et al. [22] propose *EBS* (Exclusion-based System), a dynamic key scheme that assigns each node $k$ keys from a key pool of size $k + m$. If node capture is detected, rekeying occurs throughout the network. However, the authors [22] did not indicate a method for detecting a compromised node. More-over, even if a small number of nodes in the network are compromised, information in the entire network could be discovered as well as in [13].

Section 4 summarizes the properties of these different solutions together with the proposed one within a table.

In general, existing symmetric key management solutions for WSNs focus particularly on the efficiency of key establishment after the deployment of the network. However, they do not deal with key refresh which makes key management dynamic and adds a further difficulty to the task of attackers. Furthermore, existing solutions neglect the effect of captured node attacks.

We develop in this paper a key management framework well adapted to WSNs challenges especially scalability. We focus on establishing a key refresh scheme with minimum costs that allows dealing with the resistance against the hardest attack: node compromising.

## III PDKR: A PROTOCOL DIAGRAM KEY RECOVERY SOLUTION FOR WSNS

In this section, we describe a new key management protocol for WSNs. Our main objective is to offer a robust and simple security framework that meets the resource constraints of sensor nodes. The main idea of *PDKR* is to build a diagram in a secure manner and while conserving energy after a random deployment of nodes. Thereafter, this diagram is used for rekeying to save communications. In fact, with a diagram only $log_2(n)$ messages are necessary to rekey a network of $n$ nodes. We begin by presenting the assumptions and notations used in the design of the solution, and then we give the detailed algorithms.

### *3.1. Assumptions and Notation*

Our solution relies on the following assumptions.

(i)The sensor network is static (nodes are not mobile).

(ii)The sensor nodes are homogeneous: the sensor nodes are similar in their processing capacity, communication, energy, and storage.

(iii)The deployment is random: the neighbours of any node are not known prior to deployment.

(iv)An attacker can listen to all traffic, reflect old messages, or inject its own messages.

(v)The compromise of a node implies that all information stored in its memory is known by the attacker.

(vi)The base station has no constraints on the capabilities of computing, storage and cannot be compromised.

(vii)The communication channels are bidirectional; if a node $u$ can receive a message from node $v$, then $u$ can send a message to $v$.

(viii)A base station which is generally the sink is responsible for initiating the key management process.

(ix)Each sensor node has a unique identifier.

Table 1 shows the notations that are used to write algorithms in the remaining of the paper. Each sensor node $i$, including the base station, maintains the following variables.

(i)$Father_i$ : the father of the sensor in the final recovery diagram. The base station is the root of the diagram, so $Father_{BS}$ is set to ***null***.

(ii)$Level_i$ : the level of the sensor in the diagram. The level of the root is zero, that is $Level_{BS}=0$.

(iii)$Sons_i$ : a list containing the identifiers of the sons of node $i$ within the diagram.

(iv)$Neighbours_i$ : a list containing the identifiers of the neighbours of node $i$. This list is maintained to cope with node failure and node capture as described further in this section.

TABLE 1.
NOTATION.

| Notation | Description |
|---|---|
| $S_i$ | $i^{th}$ sensor node in the network. $S_i$ denotes the (unique) identifier of the sensor node. |
| $\{M\}_k$ | The encryption of message $M$ with key $k$. |
| $S \rightarrow *: M$ | A node $S$ broadcasts the message $M$, any node in the radius of perception of the BS receives the message $M$. |
| $MAC_k(M)$ | The Message Authentication Code of the message $M$ with the symmetric key $k$. |
| $A \| B$ | The concatenation of the information $A$ with information $B$. |
| $N_i$ | A nonce generated by node $S_i$. |

The protocol uses the following types of messages.

(1)$\{HELLO, Sender\_ID, Sender\_Level, MAC_{Kr}(Sender\_ID, Sender\_Level, Sender\_Father)_{Kr}\}$ : this message is used to construct the recovery diagram. It is encrypted with the pre-deployed key $K_r$. ***Sender_ID*** is the identifier of the sensor node that sends the message. The ***Sender_Level*** is the position of the node in the diagram. So, the base station which is the root of diagram is at position ***0***.

(2)$\{REFRESH, BS, New\_K_r, Mal\_List, MAC_{KBS}, Si(BS, new\_K_r)\}_{KBS}$, ***Si*** :this message initiated by the base station is used to update key $K_r$ which is shared by all the sensor nodes. ***Mal_List*** is a list of nodes that are suspected to be malicious (captured nodes).

(3)$\{REFRESH\text{-}REQ, S_i, <Mal_{ID}>_{Ki\text{-}BS}\}$: this message is sent by a sensor node to the base station to request a key refresh. A node $S_i$ requests a key refresh when it suspects a neighbour ***Mal_ID*** to be captured.

(4)$\{JOIN, S_n, N_n, MAC_{Kr}(S_n, N_n)\}_{Kr}$: this message is used by a new node to join the network. $S_n$ is the identifier of the new node and $N_n$ is a nonce generated by him.

(5)$\{Join\text{-}Ack, S_i, Level_i, N_n, N_i, MAC_{Kr}(S_i, Level_i, N_i)\}_{Kr}$ : this message is used by neighbouring nodes to acknowledge the receipt of a Join request.

(6)$\{Father, S_n, S_i, N_i\}_{Kr}$: this message is used by a new node $S_n$ to inform surrounding nodes that its father in the recovery diagram is node $S_i$.

### *3.2. ALGORITHM*

Each sensor node $S_i$ is launched with three keys: $K_{i,BS}$, $K_{BS,i}$ and $K_r$. $K_{i,BS}$ and $K_{BS,i}$ are shared with the base station. They both serve to secure communications between the sensor node $S_i$ and the base station. $K_{i,BS}$ serves to encrypt/decrypt messages sent by $S_i$ to the base station while $K_{BS,i}$ serves to encrypt/decrypt messages sent by the base station to $S_i$. The aim of using two keys is to make more difficult the task of a cryptanalysis attacker. Key $K_r$ is shared by all nodes of the network; this key is used to encrypt (decrypt) messages immediately after deployment.

After deployment, each node copies its key $K_r$ in its RAM (volatile memory) and removes it from its non-volatile memory (EROM). If an attacker captures (physical access) a node after deployment, he will not have access to the key $K_r$ rapidly. The base station initiates the construction of recovery diagram by broadcasting a ***HELLO*** message as follows:

$$BS \rightarrow *: \{HELLO, BS, 0, null, MAC_{Kr}(BS, 0, null)\}_{Kr} \quad (1)$$

The purpose of this message is to discover neighbouring nodes of the base station. Upon receiving the message for the first times each node $S_i$ set, its father to ***BS*** and sets its level in the diagram to ***1***. Then, $S_i$ broadcasts a similar message that is, $S_i \rightarrow *$: $BS \rightarrow *: \{HELLO, S_i, 1, BS, MAC_{Kr}(S_i, 1, BS)\}_{Kr}$, in its neighbouring hood to allow other nodes to join the diagram and so on until all the nodes join the diagram.

When receiving other ***HELLO*** messages, each node uses them to set its list of sons in the diagram and computes common keys with them or simply constructs its list of neighbours. ***Algorithm*1** describes the detailed actions achieved by a sensor node when receiving a ***HELLO*** message.

***Algorithm 1***

Recieve
$\{HELLO, Sender\_ID, Sender\_Level, MAC_{Kr}(Sender\_ID, Sender\_Level, Sender\_Father)\}_{Kr}$

**If** the message is received for the first time **Then**
    $FATHER_i := Sender\_ID$; /* Father receives the identifier of the sender. */
    $Level_i := Sender\_Level + 1$;
**If** ($Father_i \neq BS$) **Then**
/*Compute a common key with the father */
$K_{Si}$, $Sender\_node := H_{Kr}(S_i \| Sender\_node \| Level_i)$ ;

/* if the father is the *BS* a shared key already exists.
$S_i \rightarrow *$: $\{HELLO, S_i, Level_i, Father_i, MAC_{Kr}(S_i, Level_i, Father_i)\}_{Kr}$;

**Else**

**If** ($Level_i = Sender\_Level - 1$) **AND** ($Sender\_Father = S_i$) **Then**
Add the node *Sender_ID* to the list of sons;
**If** $S_i <> BS$ **then**
Compute a shared key with the son *Sender_ID*:
    $K_{Si\ Sender\_ID} := H_{Kr}(S_i \| Sender\_ID \| Level_i + 1)$;

**EndIF**
**Else**
Add the node $S_i$ to the list of neighbours;
**End IF**

Figure 3 shows some steps of the construction of the recovery diagram of the WSN depicted on Figure 1. Figure 3(a) shows the first step where the neighbours of the base station join the diagram. In Figure 3(b), we only show the step during which neighbours of nodes S5 join the diagram. Finally, the complete diagram is depicted in Figure 3(c).

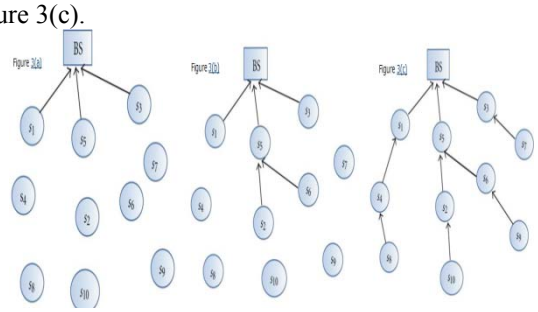


Figure 3. Diagram construction.

*3.3. Diagram Maintenance and Rekeying*

Once the diagram is constructed, each node shares a symmetric key with the base station, a symmetric key with its father in the Diagram, and the key $K_r$ with the whole network. A rekeying process is launched by the base station periodically to refresh $K_r$ as follows.

(1)The ***BS*** sends to each son $S_i$ a ***Refresh*** message encrypted with their common key $K_{BS,Si}$ $\{REFRESH, BS, New\_K_r, null, MAC_{KBS,Si}(BS, new\_K_r)\}_{KBS,Si}$. Note that in a periodic refresh ***Mal_ID*** is set to ***null***. However, if the base station issues the ***REFRESH*** because of a captured node attack, it includes the ***ID*** of this node in the message.

(2)When a son node of the base station receives the Refresh message, it updates key $K_r$ by $new\_K_r$, *then* it forwards the ***Refresh*** message of the base station to its own sons. The message is encrypted with the symmetric key shared between father and son. So, the ***Refresh*** message goes downward within the Diagram till reaching all the sensor nodes. Thus, all sensor nodes get the new global key.

A rekeying process may also be trigged if malicious/ captured nodes are detected in the network. Because they are resource constrained, sensor nodes cannot implement mechanisms for detecting malicious nodes. For example, putting a sensor node in promiscuous mode is not feasible. Detecting captured nodes is conceivable only if the captured node issues attacks such as sending unnecessary messages to its neighbours to cause energy depletion. When a node $S_i$ suspects one of its neighbours, say $S_m$, to be captured, two situations are possible.

(i)If the suspected node $S_m$ is the son of $S_i$, node $S_i$ ignores the messages of $S_m$ and removes it from the list of sons. Then it generates a nonce $N_i$ and sends $\{REFRESH\text{-}REQ, S_i, <S_m>_{Ki\text{-}BS}\}$to its father on the Diagram. This message goes upward until it reaches the ***BS***. This last broadcasts a refresh message $\{REFRESH, BS, New\_K_r, S_m, MAC_{KBS}, S_i(BS, new\_K_r)\}_{KBS,Si}$ in the Diagram. With this message, $K_r$ is updated and all sensor nodes are aware that $S_m$ is malicious.

(ii)If the suspected node $S_m$ is the father of $S_i$, node $S_i$ must first contact one of its neighbours in the list $neighbours_i$ to find another father and by the way another path to reach the base station. If it succeeds in this task, it issues a ***REFRESH-REQ*** as in the precedent case.

In both cases, it is the base station that decides if it issues a key refresh or not. We suppose that the base station has more means and resources to detect malicious/captured nodes. Captured nodes may also send ***REFRESH-REQ*** messages only to consume energy and to evict honest nodes from the network. If the base station detects this behaviour, it may issue a ***REFRESH*** message that evicts the node that sends the ***REFRESH-REQ*** from the network.

Adding a new node $S_n$ is achieved as follows.

(1) $S_n$ generates a nonce $N_n$ and broadcasts a join message as follows: $S_n \rightarrow *:\ \{JOIN, S_n, N_n, MAC_{Kr}(S_n, N_n)\}_{Kr}$, The ***JOIN*** message is encrypted with the current value of $K_r$ , the key shared by all the sensor nodes. So, the new sensor node must be launched with this key at deployment. Note here that deploying a new sensor must be synchronized with the periodic rekeying process to avoid deploying a new node with an obsolete key.

(2)When receiving the ***JOIN*** message, every node $S_i$ in the transmission range of the new node generates a nonce $N_i$ and responds with the following message:

$$\{JOIN\text{-}Ack, S_i, Level_i, N_n, N_i, MAC_{Kr}(S_i, Level_i, N_i)\}_{Kr} \quad (2)$$

(3)The new node declares the source of the first received message as a father and diffuses the following message:

$$\{Father, S_n, S_i, N_i\}_{Kr}. \quad (3)$$

(4)The father node adds the new son node in its sons list. The surrounding nodes that heard the ***Father*** message add the new node to their list of neighbours.

(5)The father and the son compute their shared keys.

A sensor node may fail or consume totally its energy. In this case, nodes that are attached to him, especially its sons, must search for another path to the base station. So, they ask the nodes in their list of neighbours to take the place of the failed father.

## IV EVALUATION

*4.1.Comparison with Some Existing Solutions*

In this section, we evaluate the performance of our solution and compare it with some existing ones. We use the following metrics to achieve this evaluation.

(i)**Memory complexity**: memory needed to store keys.

(ii)**Communication Complexity**: number of messages exchanged for key management.
(iii)**Key connectivity**: the probability that two nodes (or more) share a key.
(iv)**Resilience against node capture**: this metric measures the impact of a node compromise on the security of the rest of the network. We quantify this metric with the three following values:
(a)Good resilience: the compromised node affects only its neighbours (local influence),
(b)Weak resilience: the compromised node affects its neighbours and also some non neighbouring nodes,
(c)Very weak resilience: if the compromise of one node leads to compromise the whole network.
(v)Scalability: this metric measures the flexibility of the protocol with the size of the network. In other words, the metric shows how the cost of the protocol, that is memory and message overhead, varies when the network becomes larger. Scalability is a very imp-ortant metric to consider when distributed algorithms are proposed, especially for dense WSNs. To quantify scalability, we use the following values:
(a)Very good: the protocol does not induce further costs when the number of nodes in the network increases,
(b)Good: the protocol induces reasonable costs when the number of nodes increases,
(c)Medium: the cost of the protocol depends on the number of nodes.
Table 2 presents the results of com-paring key management protocols described in Section 2 with the pro-posed solution. The different notations used in the table are described in Section 2. In the proposed scheme, a node needs to store initially three keys before deployment. After deployment, each node computes a number of keys that is the function of the number of its neighbours. If a node $S_i$ has $d$ neighbours, where $d'$ are sons, the set of keys stored by the node $S_i$ is two keys (with the base station) + one key (shared by the whole network) $+d'$ keys. The memory complexity of our solution is acceptable for nowadays sensors.

TABLE 2.
COMPARISON WITH SOME EXISTING SOLUTION.

| Metrics / Schemes | memory Complexity | communication Complexity | Key connectivity | Resilience against node capture | Scalability |
|---|---|---|---|---|---|
| Key Infection [5] | Depends on the number of one hop neighbors ($d$) | For each node : $2 * d$ | 100 % | Weak | Good |
| BROSK [6] | 1 | $2 * d$ | 100 % | Very weak | Very good |
| Lightweight Key Management System [7] | $4+2g$, where g is : number of group in network | $2 * d$ | 100 % | Very weak | Very good |
| Blom Scheme [8] | $\lambda(\lambda + 1)$ | $d + 1$ | 100 % | $\lambda$- secure | Medium |
| Polynomial scheme [9] | $\lambda+1$ | $d + 1$ | 100 % | $\lambda$- secure | Very good |
| SPINS [10] | 3 + the chain list of keys used by µTESLA | $3*n/2$ | 100 % | Weak | Medium |
| Random key pre-distribution [11] | Key pool size ($m$) + keys identifier s | $d + 1$ | Probability that two nodes share a key, say $p_1$ | Depends on $m$ and $p_1$ | Good |
| Q-composite [12] | $2 * m$ | $d + 1$ | Probability that two nodes share a key say $p_2 > p_1$ | Depends on $m$ and $p_2$ | Medium |
| Key management using deployment knowledge [13] | $d$ - 1 | $d + 1$ | Probability that two nodes share a key, say $p_3$ | Depends on $d$ and $p_3$ | Good |
| Dynamic key management [14] | $k$ keys + keys' identifiers | $d + 1$ | Probability that two nodes share a key, say $p_4$ | Depends on $k$ and $p_4$ | Good |
| LEAP [15] | $(3 * d) + 2$ + keys chain of µTESLA | $(2 * d) + 1$ | 100 % | Very good | Good |
| Location-based keys [16] | $2 * d + 1$ | $2 * d$ | Probability that two nodes share a key, say $p_5$ | $\lambda$- secure | Good |
| Our solution | 3 + number of sons | $d + 1$ | 100 % | Good | Good |

The analysis of the communication complexity for the construction of Diagram is measured by the number of messages received and issued by each node. Each node sends a message and receives $d$ messages from its neighbours, that is, $d + 1$ *message* by each node. Our solution has a low-complexity communication over other proposed solutions [11, 4-8], moreover, it is deterministic; key connectivity is equal to one (no concept of probability).

Preserving energy in WSNs is very important and having a low communication complexity means that the **solution minimizes energy**.

### *4.2. Simulation Results*

To validate the results presented by table 2, we also measured the performance of our protocol by simulation. In this section, we provide an overview of our simulation model and some of the results we obtained. We implemented our scheme within the MATLAB framework (MATLAB for Matrix Laboratory is a matrix-based system for scientific and engineering calculation). We considered WSNs with 500 sensor nodes deployed randomly in an area of 250 * 250 meters square. Each sensor node has a signal range of 15 meters.

We first focused on evaluating the average number of messages received by a sensor node after a random deployment. This number of messages is equal to the number of neighbours of the sensor node as explained earlier. Multiple runs were conducted. Each run corresponds to a particular random deployment of the 500 identified sensor nodes. During each run, we measured the number of neighbours of each sensor node. We to-ok the average of the values registered at the level of each sensor node. Figure 4 illustrates the obtained results. It appears that our random deployments generate an average number of neighbours at most equal to 12. If the value of energy consumed by each type of message is known, we can compute the total energy consumed during Diagram construction.

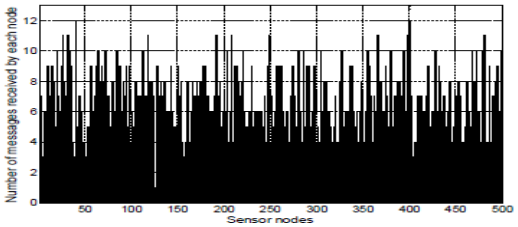


Figure 4 Average numbers of messages for each node in a network of 500 sensor nodes.

We then considered the resilience of our scheme to node capture. We compute the resilience to node capture by the fraction of total communication links that are compromised by the capture of $x$ nodes. We assume that sensor nodes are randomly captured by an attacker. Figure 5 presents the obtained results. It shows that for a network of 500 sensor nodes, an attacker must capture at least 100 sensor nodes to reach all communications in the network. This corresponds to 20% of nodes in the network.

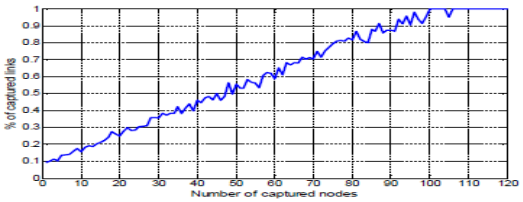


Figure 5. Percentage of corrupted links versus number of captured nodes.

We also carried out multiple experiments on networks of size ranging from 500 to 1000 nodes. In each experiment we computed the maximum number of messages that can be received by a node. We took the average number of this value for each network size. Figure 6 summarizes these results. It shows how the number of messages received by a sensor node evolves when the size of the network becomes larger. The curve is almost logarithmic which is very acceptable.

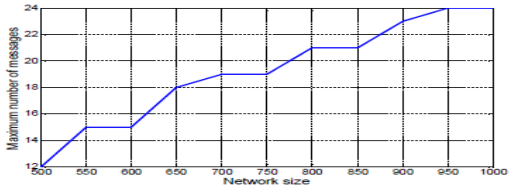


Figure 6. Maximum number of messages received by a node versus network size.

### *4.3. Security Analysis*

In this section, we analyze the security of our solution. As mentioned in the assumptions (cf. Section 3.1), the base station will not be compromised. An outsider attacker, who does not know the key $K_r$, cannot discover the meaning of messages diffused by nodes after deployment. Nevertheless, an attacker can compromise one or more nodes, so he becomes an insider attacker. The keys of the compromised node can be used for forging wrong messages (reading message, for example,) and also consume nodes' energy by sending useless messages to his sons and father. If the base station can detect an abnormal behaviour of a node, it can lunch a rekeying to refresh $K_r$ and revoke this node. In the following, we analyze the behaviour of our solution for three types of attack.

(1)*HELLO* flood attack: in our proposal, nodes discover their neighbours by sending a ***HELLO*** message encrypted with the key $K_r$. An attacker without knowing the key $K_r$ could not launch a ***HELLO*** Flood attack.
(2)Sybil attack: in the algorithm, an MAC of the node's identifier, its level, and its father's identifier is calculated to authenticate the sender and the receiver. Therefore, a node cannot play a role of other nodes.
(3)Node capture attack: when a node is captured, this does not affect its neighbours. In fact, after a node is captured, what can an attacker do? Since he has the key shared with the base station, he may send wrong information (lectures) to that base station. The latter may have a mechanism to verify the behaviour of sender nodes. The attacker also has access to the keys of the sons of the victim enabling it thereafter to send unnecessary messages to these sons in order to consume their energy and cause battery depletion. Nodes that detect these useless messages can suspect the captured node and inform the base station with a ***REFRESH-REQ*** message.

## V CONCLUSION

Security is a necessity for most applications using WSNs, especially if the sensor nodes are deployed in unsafe areas, such as battlefields, strategic places (airports, critical buildings ...). These sensor nodes operating in difficult access places, without protection and without possibility of recharging their batteries, may be subject to disruptive and malicious actions. Therefore, it is important to provide to them an acceptable security level. The primary objective of WSN nodes is to collect data and transmit them to a decision center. So, this must be done in a trustworthy and safe way. In this paper, we presented a key management solution to WSN that deals with one of the hardest attack: node capture. The main idea of the solution is to quickly and cheaply build a recovery diagram that serves to refresh the shared key with minimum costs. The solution is scalable and uses little memory.

As a perspective of our present work, we plan to use the NS3 simulator to compare the performance of our solution with other solutions from the literature and particularly those of [6-7-8]. We also work on **energy** and a mobile version of our scheme.

## REFERENCES

[1] C. Buratti, A.Conti, D. Dardari and R. Verdone, An Overview on Wireless Sensor Networks Technology and Evolution. www.mdpi.com/ journal/sensors, 6869-6896 (2009).
[2] T Kavitha, D Sridharan, Security vulnerabilities in wireless sensor networks: a survey. Journal of Information Assurance and Security 5, 31–44 (2010)
[3] Y Xiao, VK Rayi, B Sun, X Du, F Hu, M Galloway, A survey of key management schemes in wireless sensor networks. Computer Communications 30(11-12), 2314–2341 (2007).
[4] Suman Bala, Gaurav Sharma and Anil K. Verma: Classification of Symmetric Key Management Schemes for Wireless Sensor Networks. International Journal of Security and Its Applications, Vol. 7, No. 2, March, 2013, 117-138.
[5] Tahira Laskar, Debasish Jena : A Survey on Key Management Issues in WSN. International Journal of Engineering and Innovative Technology (IJEIT), Volume 1, Issue 5, May 2012,74-77, ISSN: 2277-3754
[6] Dahai Du, Huagang Xiong, and HailiangWang, An Efficient Key Management Scheme for Wireless Sensor Networks. International Journal of Distributed Sensor Networks, Vol 2012, Article ID 406254, 14 pages
[7] Kamaljit Singh, Lalit Sharma, Hierarchical Group Key Manage-ment using Threshold, International Journal of Computer Applications (0975 – 8887), Volume 63–No.4, February 2013
[8] Saewoom Lee, Youngtae Noh, and Kiseon Kim, Key Schemes for Security Enhanced TEEN Routing Protocol in Wireless Sensor Networks. International Journal
[9] RJ Watro, D Kong, S-F Cuti, C Gardiner, C Lynn, P Kruus, TinyPK: securing sensor networks with public key technology. Proceedings of the 2004 ACM Workshop on

Security of Ad Hoc and Sensor Networks (SASN '04), October 2004, 59–64.

[10] P Ning, A Liu, TinyECC: elliptic curve cryptography for sensor networks (http://discovery), csc.ncsu.edu/ software/ TinyECC/ website

[11] R Anderson, H Chan, A Perrig, Key infection: Smart trust for smart dust. Proceedings of the 12thIEEE International IEEE International Conference on Network Protocols (ICNP '04), October 2004, 206–215.

[12] R Blom, An optimal class of symmetric key generation systems. Proceedings of the Eurocrypt 84 Workshop on Advances in Cryptology: Theory and Application of Cryptographic Techniques, 1985 (Springer), pp. 335–338.

[13] C Blundo, AD Santix, A Herzberg, S Kutten, U Vaccaro, M Yung, Perfectly-secure key distribution for dynamic conferences. Proceedings of the 12th Annual International Cryptology Conference on Advances in Crypto-logy, 1992, Berlin, Germany (Spring), pp. 471–486.

[14] B Lai, S Kim, I Verbauwhede, Scalable session key construction protocol for wireless sensor networks. Proceedings of the IEEE Workshop on Large Scale RealTime and Embedded Systems (LARTES '02), 2002.

[15] B Dutertre, S Cheung, J Levy, Lightweight key management in wireless sensor networks by leveraging initial trust. SDL Technical Report SRI-SDL-04-02 (2004).

[16] A Perrig, R Szewczyk, V Wen, D Culler, JD Tygar, SPINS: security protocols for sensor networks. Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, July 2001 (ACM Press), pp. 189–199.

[17] S Zhu, S Setia, S Jajodia, LEAP: efficient security mechanisms for large-scale distributed sensor networks. Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03), October 2003, 62–72.

[18] D Liu, P Ning, Location-based pairwise key establishments for static sensor networks. Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (CCS '03), October 2003, 72–82.

[19] L Eschenauer, VD Gligor, A key-management scheme for distributed sensor networks. Proceedings of the 9th ACM Conference on Computer and Communications Security, November 2002, 41–47.

[20] D Liu, P Ning, LI Rongfang, Establishing pairwise keys in distributed sensor networks. ACM Transactions on Information and System Security 8(1), 41–77 (2005).

[21] W Du, J Deng, YS Han, S Chen, PK Varshney, A key management scheme for wireless sensor networks using deployment knowledge. Proceedings of the IEEE International Conference on Computer Communications (IEEE INFOCOM '11), March 2004, 586–597.

[22] M Eltoweissy, M Moharrum, R Mukkamala, Dynamic key management in sensor networks. IEEE Communications Magazine 44(4), 122–130 (2006). of Distributed Sensor Networks, Volume 2013, Article ID 391986, 8 pages

[23] Xiaobing He, Michael Niedermeier and Hermann de Meer Dynamic Key Management in Wireless Sensor Networks: A Survey Journal of Network and Computer Applications, Volume 36, Number 2, pp. 611-622, 2013.