A New Resilient Key Management Protocol for Wireless Sensor Networks

Chakib Bekara and Maryline Laurent-Maknavicius

Institut National des Télécommunications d'Evry
Département Logiciel-Réseaux
9 rue Charles Fourier, 91000 Evry Cedex, France

Abstract. Wireless Sensor Networks (WSN) security is an important issue which has been investigated by researchers for few years. The most fundamental security problem in WSN is key management that covers the establishment, distribution, renewing and revocation of cryptographic keys. Several key management protocols were proposed in the literature. Unfortunately, most of them are not resilient to nodes capture. This means that an attacker compromising a node can reuse the node's key materials to populate any part of the network with cloned nodes and new injected nodes. In this article, we present a simple polynomial-based key management protocol using a group-based deployment model without any necessary predictable deployment location of nodes. That solution achieves high resilience to nodes compromising compared with other protocols.

Keywords: WSN Security, Key Management, Nodes Compromising, Intrusion Detection.

1 Introduction

Wireless sensor networks (WSN) are infrastructure-less and self-organizing networks, which can be deployed anywhere, and work without any assistance [1]. These characteristics motivated their deployment, but introduced critical security issues like network control access, authentication, confidentiality and nodes compromising. WSN are very sensitive to those issues since sensors are known to be tamper-vulnerable devices [2], and deployment of them is mostly done in open area that should be assimilated as hostile area for security consideration.

Current WSN security solutions rely on secret keys but today an efficient key management protocol is still needed to generate, distribute, renew and revoke cryptographic keys. In the last few years, several key management protocols for WSN have been defined, but they do not satisfy the protocol efficiency requirements as follows:

1. Low storage, computation, and transmission overheads.
2. Resistance to nodes compromising, so keys established between non compromised nodes remain confidential even in case of nodes compromising.

D. Sauveron et al. (Eds.): WISTP 2007, LNCS 4462, pp. 14–26, 2007.

3. No on-line key management server, that would be the point of failure in case of DoS attacks.
4. Resilience to nodes compromising that prevents attackers to populate the network with clones of compromised nodes or injected false nodes, by reusing their key materials.

Most of the key management protocols [2] [3] [4] [5], satisfy one or more of the three first requirements. Unfortunately the last requirement is rarely or not enough investigated. Most of the key management protocols are poorly resilient to nodes compromising, and the few ones achieving an acceptable level of resilience, either rely on strong assumptions (i.e. nodes are tamper-resistant) [6] [4], introduce heavy overheads (the use of public key cryptography) [6], or require prior knowledge on nodes deployment [4].

In this article, we propose a simple key management protocol for static WSN, based on the well-known polynomial-based key generation protocol of [7] for pair-wise keys establishment, and our proposed group-based deployment model to ensure resilience to nodes compromising. Our protocol requires no prior knowledge on the locations of deployed nodes. It relies on realistic assumptions, and introduces no significant overhead.

The remainder of the paper is organized as follows. In section 2, we introduce the basic version of polynomial-based key generation protocol. In section 3, we present our assumptions, network model, notations and we define our group-based deployment model. In sections 4 and 5, we describe our proposed protocol, and in section 7 we give its detailed security analysis. In section 8, we briefly review related works on key management in WSN, and we compare the resiliency of our protocol to the resiliency of the related works in section 9 , and we conclude our work in section 10.

2 Polynomial-Based Key Generation Protocol

Blundo et al. [7] present a new pair-wise key establishment protocol, based on a symmetric bivariate polynomial. A trusted key server in the network generates a symmetric bivariate polynomial, as follows:

$$f(x,\ y) = \sum_{i,j=0,\ldots,t} a_{ij} x^i y^j \pmod{Q} \tag{1}$$

where Q is a sufficiently great prime number, $1 \leq a_{ij} \leq Q-1$, and t is the degree of the polynomial and a security parameter. Initially, the key server configures each node u with its unique secret polynomial share:

$$f_u(y) = f(Id_u,\ y) = \sum_{i,j=0,\ldots,t} a_{ij} (Id_u)^i y^j \pmod{Q} \tag{2}$$

where Id_u is the unique identifier of node u in the network. Two nodes of the network u and v, can easily establish a unique shared secret key by computing:

$$K_{uv} = f(Id_u,\ Id_v) = f(Id_v,\ Id_u) = K_{vu} \tag{3}$$

As long as the number of compromised nodes in the network is less than $t+1$ nodes, the system is secure. The greater the t-value is, the more secure (resistant to nodes compromising) the system is.

3 Assumptions, Notations and Network Model

3.1 Assumptions

First, we suppose that the Base Station (BS) is a *trusted* and a *powerful* entity in the network, that cannot be compromised.

Second, we suppose that sensors are static, so once they are deployed they do not leave their locations. In many scenarios (i.e. perimeter monitoring), WSN are considered as static, either because sensors are fixed or because sensors are not asked to be mobile for achieving their tasks.

Third, we suppose that sensors are deployed progressively in successive generations (groups). This assumption is adopted in most group-based deployment key management protocols like [3] and [8]. However, unlike the other group-based key management protocols, we do not suppose that nodes of the same generation are deployed in the same neighborhood. In our protocol, nodes of the same generation might be deployed anywhere in the network. Therefore, our protocol is not based on any prior knowledge on deployment location of nodes, but if such information was available, our protocol will achieve better resilience.

Fourth, we suppose that an attacker needs a minimum time T_{comp} in order to compromise a node after it is deployed. T_{comp} is greater than the time T_{est}, which is the maximum time needed by a newly deployed node to establish pair-wise keys with its one-hop neighbors. This assumption is present in several works like [9] and [5], and is likely to be true, because an attacker must first have a physical access to a sensor, and then use some programming tools in order to retrieve sensor's key materials. However, in [9] and [5], deployed nodes are initially loaded with some common secrets that nodes use to establish different keys (pair-wise keys, cluster keys) with their neighbors. In addition, [9] and [5] require that each newly deployed node erases these common secrets after a time T_{est} from its deployment, to prevent that an attacker can get them if it is compromised. In our protocol, no such assumption exists, because each node only needs its unique secret polynomial share for pair-wise key establishment.

Fifth, we suppose that sensors are synchronized with the BS. This could be done through an authenticated beacon periodically broadcasted by the BS, to keep sensor's clocks synchronized with the BS's one. Authentication can be guaranteed using the $\mu Tesla$ protocol [10].

Finally, we suppose that an attacker can only get a partial control over the network. In case of full control on all deployed nodes, security solutions will be inoperative to stop the attacker.

3.2 Notations

Table 1 summarizes the notations used in this paper.

Table 1. The used notations

Notation	Significance
u, v	Two nodes of the WSN
Id_u	4 bytes unique identifier of node u in the network.
N_u	An increasing nonce value generated by node u
f_u	The secret polynomial share of node u
$K_{uv} = K_{vu}$	The secret pair-wise key established between u and v
$MAC_K(M)$	The message authentication code of M using the secret key K
H	A one way hash function
$\|x\|$	The length on bytes of x
$a\|\|b$	a concatenated to b

3.3 Network Model and Security Considerations

The BS deploys nodes in multiple generations numbered successively from 1 to n, where n is the maximum number of deployed generations. If we suppose that $n < 2^{16} - 1$, each generation's number is identified by exactly 2 bytes. The order of deployment must be respected $G_1, \ldots, G_i, G_{i+1}, \ldots, G_n$, where G_i is the i^{th} deployed generation. Each node belongs to a unique generation.

Because nodes are not mobile in our network, it is logical that *only* nodes of the newly deployed generation ask for key establishment with their neighbors, which may belong either to the same generation, or to former deployed generations. Nodes of former generations *can not* request for key establishment, and even if they do request, their requests must be rejected. Based on this assumption, we can state that any key establishment request originates from:

- either a node from the newly deployed generation,
- or a node deployed by an attacker, which is either a node having a false Id, or a cloned node having the Id of a compromised node.

For security reasons, we suppose that any newly deployed node u sets a timer to the value T_{est} straight after deployment. Once the timer expires, node u rejects any key establishment request originating from a node of the same newly deployed generation.

4 Our Proposed Protocol

We propose a resilient key management protocol, based on the use of a symmetric polynomial for secure key establishment, and based on our defined group-based deployment model for achieving resilience to nodes compromising. Our protocol involves three phases: *initialization*, *pair-wise key establishment* and *key-path establishment*.

4.1 Initialization Phase

Initially, the BS generates a random symmetric bivariate polynomial $f(x, y)$ (see (1)). The BS then selects a group of nodes to form the next deployed sensors

generation. The BS loads each node u with a unique secret polynomial share (see (2)) as follows: suppose $u \in G_i$, then the BS loads the node with the secret polynomial share $f_u(y) = f(i||Id_u,\ y)$.

Note that it is impossible that two different nodes can have the same secret polynomial share, so a node can never lie on its real identifier or the real generation number to which it belongs. Indeed, suppose that $u \in G_i$ and $v \in G_j$, with $i \neq j$. $f_u(y) = f_v(y)$ is possible, only and only if $i||Id_u = j||Id_v$. Each generation's number is exactly 2 bytes length, and each node identifier is exactly 4 bytes length, so $\mid i||Id_u \mid = \mid j||Id_v \mid =$ exactly 6 bytes. With our well formatted extended node identifier (2 bytes generation number, 4 bytes node ID), starting from an extended node identifier $i||Id_u$, it's impossible to find another distinct node identifier $j||Id_v$ where $i \neq j$ or $Id_u \neq Id_v$.

4.2 Pair-Wise Key Establishment Phase

Suppose that the BS deployed some previous generations, say the i first generations (1, 2, ..., i), and just deployed generation $i+1$. In our protocol, nodes know the highest deployed generation's number $i+1$ through a mechanism we describe in section 5.

Let $u \in G_j$ a newly deployed node. It is obvious that as a well-behaving node, $u \in G_{i+1}$. Node u tries to establish secure links with its direct neighbors by locally broadcasting a *Hello* message:

$$u \rightarrow * : \qquad Hello, j, Id_u, N_u$$

where N_u is used to guarantee response freshness. Let $v \in G_z$, where $z \leq i+1$, a neighbor node of u receiving its message. For node v to decide serving node u, node v follows two steps:

1. v checks if $j{=}i+1$, to verify whether u belongs to the newly deployed generation. If the verification fails, it simply rejects the request of node u, because u is normally already deployed.

2. If v verifies that $j{=}i+1$ then:
   - If v belongs to generation $z \leq i$, then v computes $K_{vu} = f_v(i+1||Id_u)$ and sends back to node u the following message:

$$v \rightarrow u : \qquad z, Id_v, N_v, MAC_{K_{vu}}(z, Id_v, N_v, N_u)$$

   - If $j{=}z{=}i{+}1$ $(u, v \in G_{i+1})$, then:
     - If the timer set by node v (to the value T_{est}, see section 3.3), did not expire, do the same treatment as the previous case.
     - If the timer expired, reject the request, because node u is suspected to be malicious.

Upon receiving node $v's$ message, node u computes $K_{uv} = f_u(z||Id_v)$, and checks the message authenticity. If the message is authenticated, node u sets

K_{uv} as the shared pair-wise key with v, and sends to v the following message to conclude and mutually authenticate the key establishment process:

$$u \rightarrow v : \qquad ok, MAC_{K_{uv}}(ok, N_v)$$

Upon receiving node $u's$ response, node v checks the message authenticity using K_{vu}, and, if successfully done, node v sets K_{vu} as the shared pair-wise key with u, otherwise (failed authentication, or non received response), it erases K_{vu}.

At the end of this phase, either a pair-wise key is established between two valid nodes, or the pair-wise key establishment fails in case one of the two nodes is suspected of being a clone or a false node. The described protocol guarantees that any served key establishment request, originates from a node belonging to the newly deployed generation $i+1$. However, the current version of the protocol fails to detect two particular attempts of false key establishment. The first attempt is when an attacker compromises a newly deployed node of generation $i+1$ and deploys clones in the neighborhood of nodes of older generations, and the second attempt is when an attacker compromises an older deployed node, and tries to respond to the *Hello* messages of the newly deployed nodes. Solutions to these two particular attempts are presented in section 7.

4.3 Key-Path Establishment Phase

In our scheme, two non neighboring nodes might attempt to establish a secret key. The two nodes might belong to the same generation, or to two different generations. Moreover, the node initiating the establishment might be from a higher generation, same generation, or lower generation than the target node. This raises a problem because we supposed that only newly deployed nodes are eligible for requesting key establishment. We rely on the previously established pair-wise keys in order to overcome this problem, and to guarantee each node that the other node is a valid node and not a cloned node or a false injected node.

Let $u \in G_i$ and $v \in G_j$ be two distant nodes that need to establish a secret key, and consider that u initiates the communication, and u knows Id_v and that $v \in G_j$. First, u must find a secure path to node v, formed by previously established secure links. Once the path is found, node u sends to v the following *Key Establishment Request* (KER) message:

$$u \rightarrow v : \qquad i, Id_u, N_u, MAC_{K_{uv}}(i, Id_u, N_u)$$

where $K_{uv} = f_u(j||Id_v)$. The KER message is sent in a secure path, where each node in the path, including u and v, authenticates the message with the key it shares with the previous node in the path. Upon receiving the KER message, node v computes $K_{vu} = f_v(i||Id_u)$, and then checks the authenticity of the message. If the KER message is authenticated, then v sends back to u the *Key Establishment Confirmation* (KEC) message:

$$v \rightarrow u : \qquad ok, MAC_{K_{vu}}(ok, N_u)$$

to conclude the key-path establishment. Node u checks the authenticity of the KEC message, and in case of unsuccessful authentication, it erases K_{uv}.

Note that thanks to the secure path, each node is ensured that the other node is neither a cloned node nor an injected node. Indeed, suppose that the path is going through nodes: u, w_1, ..., w_i, w_{i+1}, ..., w_r, v. According to section 4.2, each pair-wise key (secure link) in the network is established between two valid nodes, which are neither cloned nodes nor false injected nodes. As a consequence, the fact that u found a secure path and received the KEC message means that node v is a valid node, and the fact that node v received the KER message through the path, means that node u is a valid node.

5 Determining the Highest Deployed Generation

Now let describe how nodes know the number of the highest deployed generation, as seen in section 4.2.

The BS initially defines a static scheduling for generations deployment. The BS considers deploying the first generation G_1 at instant $T_1 = 0$, which serves also as a reference within deployed nodes for synchronization and time counting. If a period T is defined between each generation deployment, each node of generation i needs only be loaded with its generation's deployment time T_i, and the period time T.

After nodes deployment, when a node $u \in G_i$ asks for key establishment, a neighbor node v of an older generation j verifies that u is a node of the newly deployed generation G_i, by verifying that $0 < t_{current} - (i-1) * T < T$, where $t_{current}$ is the current time in node v. If this inequality is not verified, v rejects the request of u, because u is not a node from the highest deployed generation.

6 Computation and Memory Costs

Now, let consider the computation and memory costs of our protocol. For the memory cost, each node stores its extended identifier (generation number, node ID), its polynomial share and the established pair-wise keys. An extended identifier is 6 bytes length (see section 4.1). A polynomial share is represented by *t+1* coefficients, plus the modulo Q. If we choose a modulo Q of 8 bytes, as in [11], and *t=100*, each node needs 816 bytes memory to store its polynomial share. In addition, each established pair-wise key needs 8 bytes of memory.

For the computation cost, each node needs to evaluate its polynomial share for each pair-wise key establishment. As described in [11], evaluating a polynomial share requires t modular multiplications and t modular additions in a finite field F_Q. However, because a sensor's CPU does not manipulate words of 64 bits (8 bytes), and the more powerful of them, like MOTEIV [12], handles words of 16-bit only (2 bytes), more modular multiplications and modular additions are needed. Consequently, in a 16-bit CPU processor, evaluating a t-degree polynomial share $f_u(y)$ over a finite field F_Q, where Q is a prime number of 64 bits,

and y is an extended identifier (see section 4.1) of 48 bits (6 bytes), requires $4 \times t$ modular additions, and $12 + 28 \times (t-1)$ modular multiplications.

7 Security Analysis of Our Solution

The proposed security analysis of our protocol focuses on the resilience to nodes compromising, and other features.

7.1 Resilience to Nodes Compromising

Now, let consider the resilience to nodes compromising, which refers to the capability of an attacker to inject cloned nodes and new nodes in the network, using the key materials it gets from the compromised nodes.

Injecting Nodes with False Ids. It is clear that as long as an attacker compromises fewer than $t+1$ nodes, new nodes with non-existing Ids can not be injected in the network. In our protocol, each node u possesses a unique polynomial share bound to its identity $f_u = f(i||Id_u,\ y)$, where $u \in G_i$. After compromising node u, an attacker can not create node u', with a new identity $Id_{u'}$ and a new polynomial share $f_{u'}(y) = f(i||Id_{u'},\ y)$. In addition, an attacker can not use the polynomial share f_u, because node u' will fail to establish secret keys with this polynomial share using the new identity $Id_{u'}$.

As a conclusion, our protocol is resilient to the injection of false nodes with non-existing identifiers in the network.

Injecting Cloned Nodes. In our protocol, and under our assumptions of section 3, an attacker is highly unlikely to deploy cloned nodes, and convince his neighbors of the validity of the clones.

Table 2 summarizes the different scenarios for key establishment. Four possible cases of key establishment can be differentiated according to the generations to which the *requesting node* and the *responding node* belong. Next the behavior of the attacker is analyzed according to the role of the attacker which is either a requesting node, or a responding node.

First, let see how our protocol handles the two last cases $Old-New$, and $Old-Old$, where a node u of an older deployed generation asks for key establishment with a node of a newer generation, or an older generation. Remember that only nodes of the newly deployed generation are able to establish secure links with their neighbors. As a consequence, an attacker compromising an older deployed node $u \in G_i$, can not initiate key establishment with another deployed node $v \in G_j$ where $j > i$. In addition, the mechanism described in section 5 guarantees that all nodes of the network have the same view of the number of the highest deployed generation, so a node $u \in G_i$ of an older generation can not ask for key establishment with a node $v \in G_j$ where $j \leq i$.

Second, let see how our protocol handles the first case $New-New$, where an attacker compromises a newly deployed node and asks for key establishment with another newly deployed node of the same generation. By limiting the duration

Table 2. Identified scenarios for key establishment according to the generation of concerned nodes

Requesting node	Responding node
New	*New*
New	*Old*
Old	*New*
Old	*Old*

of key establishment phase for newly deployed nodes to T_{est}, even if an attacker compromises a newly deployed node in a period of time T_{comp} ($T_{comp} > T_{est}$), he cannot establish secure links with other nodes of the same generation, simply because the responding nodes will reject the request, as described in section 4.2.

Now let see how our protocol handles the second case, where a node of the newly deployed generation asks for key establishment with a node of an older generation. Again, two cases are distinguished:

- First, the newly deployed node (requesting node) is a cloned node of a compromised node that belongs to the newly deployed generation.
- Second, the responding node is a cloned node of a compromised node that belongs to an older generation.

For the first case, unfortunately the algorithm in section 4.2 does not handle this situation. This case is difficult to detect, because the cloned node looks like a node belonging to the highest deployed generation. One solution could be that an older deployed node accepts establishing secure links with nodes of any newly deployed generation only during a period of time T_{max} after the deployment of any new generation, where $T_{est} < T_{max} < T_{comp}$. According to section 5, nodes know the static scheduling of generations deployment, so each deployed node sets a timer to the value T_{max} when the time of deployment of a new generation is reached. Because an attacker needs at least a time T_{comp} in order to compromise a newly deployed node, we are practically ensured that an attacker compromising a newly deployed node, can not establish secure links with nodes of older generations, because these nodes will reject his request.

The second case is also difficult to detect, because the newly deployed node is asked for key establishment, and it has no way to check whether the responding node is a cloned node. The problem is even more difficult if the clone stays inactive or silent until a newly node is deployed. At this time, the clone might become active and establish a secure link with it by simply responding to its request. In this scenario, because the cloned node does not ask its neighbors for key establishment, it cannot be detected, so the newly deployed node cannot be prevented. One solution to this problem is that deployed nodes which are neighbors of both the newly deployed node and the cloned node, detect that a neighbor node exists but they have no secure links with it, so they conclude that

the node is a malicious node. As a consequence, an informative message is sent by them to the newly deployed node which erases any established key with the cloned node.

7.2 Node Revocation and Intrusion Detection

As described above, in the four possible cases of key establishment, an active attack is always detected. Moreover, a silent attacker (intruder) is also detected when he tries to respond to key establishment requests from newly deployed nodes, and the newly nodes are then notified. As a consequence, the identity of the compromised node is known, so the neighboring nodes of the cloned node can either launch a distributed revocation against it, or notify the BS which broadcasts a revocation message in the network for revoking both the compromised node and its clones.

8 Related Works

Liu et al. [4] propose a distributed location-aware key establishment protocol, based on bivariate symmetric polynomials. The protocol assumes that the network is formed by simple nodes, and some sufficiently dedicated nodes called the service nodes, which are elected amongst sensors after deployment. These nodes play the role of *trusted* key servers in the network and are assumed to be *non-compromised*. The protocol also assumes that once nodes are deployed, they know their exact location coordinates (x, y). After deployment, each service node creates a distinct t-degree bivariate polynomial, and then securely initializes each neighbor node with its secret polynomial share, using the unique location coordinates of the node. The protocol is resistant and resilient to node compromising, as long as the service nodes are not compromised, and there are fewer than $t+1$ compromised nodes initialized by the same service node. However, if a service node is compromised - which is a current threat because a service node is just a non tamper resistant sensor node - an attacker can inject clones and new nodes with new positions, deploy them in the neighborhood of the compromised service node, and establish secure links with any nodes of the network.

Dutertre et al. [9] suppose that nodes are deployed in n successive generations, and can not be compromised in a period of time less than $T_{comp} > T_{est}$ (see section 3.1). Each generation is loaded with a unique two master keys, used respectively for authentication and key generation between the nodes of the generation. Once a deployed node successfully establishes secure links with its neighbors of the same generation, it erases these two keys to prevent from attacks. In order to establish secret keys between nodes of two different generations, each generation i is also loaded with a unique secret group key GK_i that enables nodes to establish secure links with previously deployed generations $j = 1, \ldots, i-1$. In addition, each node u of generation i is loaded with a unique random value R_u, and a secret key $Su_j = H(GK_j, R_u)$ for each future generation $j = i+1, \ldots, n$, allowing it to establish secure links with nodes of newly deployed generations.

The group key GK_i is also deleted at the end of the key establishment phase. The protocol is poorly resilient to nodes compromising, as an attacker compromising a node of generation i, can establish secure links with nodes of the future deployed generations, using the compromised secret keys Su_j. Moreover, if an attacker compromises a node u of generation i in a time period less than T_{est}, he might retrieve the master keys of generation i, and the group key GK_i. As a consequence, he can deduce secure links established between nodes of generation i, and can inject cloned nodes and false nodes anywhere in the network.

Bhuse et al. propose a key distribution protocol based on the use of the Hughes's variant of the DH protocol with encrypted key exchange (HDH-EKE) [13], and based on the assumption that nodes can not be compromised, and even if they are, then they self destroy without revealing their secret cryptographic materials. All nodes are initially loaded with a common password P used for authentication, and after deployment, nodes self organize into clusters, and elect one of them to act as a key server. The key server of each cluster generates a cluster key, and securely distributes the key to each one-hop neighbor using the HDH-EKE protocol, which in turn will pursue the distribution of the key to its neighbors in the same manner, until all nodes of the cluster posses the cluster key. The cluster key is used for encrypting messages and authenticating them inside the cluster. In order to send packets between two different clusters, boarding nodes, which posses the cluster keys of two or more clusters will act as a gateway by decrypting/encrypting messages from the source cluster to the target cluster. The key server periodically updates the cluster key, by sending a random counter value used along with the secret password, and the current cluster key to produce the new cluster key. The main problem of this protocol is its high computation overhead due to the use of modular exponentiations (public key cryptography), its weak authentication mechanism. The protocol is resilient against nodes compromising as long as an attacker cannot retrieve the secret common password P. However, it's unlikely that sensors can be tamper-resistant [10] [14], where memory containing the secret cryptographic materials is hardware-protected, because this will increase significantly the cost of sensors, and sensor nodes are intended to be very inexpensive.

9 Comparison with Previous Work

As we have seen in the description of some previous works done in the literature, most of them lack to provide resilience to nodes compromising, and those providing some degree of resilience rely on some assumptions, that can not be met easily. For essence, [6] and [4] suppose that nodes are tamper-resistant devices, so they can not be compromised or they self destroy when they detect that they are under attacks, and [4] relies also in the assumption that nodes know their locations coordinates, in-order to tie each node's secret key material (i.e. its secret polynomial share) to its location coordinates, so even if in attacker succeeds into compromising a node and creates some cloned nodes, it can not deploy them anywhere in the network. Someone can suppose that the future generation of sensors will be tamper-resistant. However, tamper-resistant

devices are expensive, and constructors tendency may be to keep sensors at lower prices, with an increase of memory and computation capacities, instead of making them tamper-resistant. Localization in WSN is still under research, and the actual solutions assume that either nodes are equipped with GPS receivers, or the existence of some trusted nodes (at least three) on the perimeter of the network, that provide nodes with their locations coordinates. The first solution is unlikely to be deployed in sensors, and is energy consuming, and the second solution requires multiple trusted entities, and the resulting location coordinates are prone to errors. In our protocol, we don't assume that nodes are tamper-resistant devices, and we don't consider nodes locations.

Some other works like [5], [9] and [15], suppose that nodes share some common secret key(s) they use for key establishment, which will be erased from their memory straightforward after they finish key establishment when they are first deployed. These protocols suppose, as we do, that nodes can not be compromised in time less then T_{comp}, and that any newly deployed node needs at most a time $T_{est} < T_{comp}$ to establish keys with its neighbors. However, in the previous protocols, if an attacker succeeds to compromise a node in time less then T_{est}, it will have access to all its secret key materials, especially the common secret key(s), consequently the entire network security can be compromised, because established keys between non-compromised nodes can be retrieved, and future established keys can also be computed, and evidently cloned nodes and new nodes with non-existing identifiers can be easily injected. In our protocol, only if an attacker compromises a newly deployed node (which belongs to the newly deployed generation) in a time less than T_{est}, it will be able to deploy cloned nodes in the network and establish pair-wise keys with them, but the attacker can not compute any pair-wise key established in the network between two non-compromised nodes. If an attacker compromises an old deployed node (which belongs to an old deployed generation), it can not deploy cloned nodes of it, and even if the cloned nodes launch a silent attack (see section 7.2) they will be detected.

10 Conclusion

Our proposed solution improves considerably resilience to nodes compromising compared with other protocols, and does not require any prior knowledge relative to nodes deployment, and any common secret key pre-establishment between nodes. Moreover, the solution does not rely on non-realistic assumptions like supposing that compromised nodes do not divulge their secret keys or that some nodes in the network can not be compromised.

Our protocol uses t-degree polynomial-based key generation protocol for achieving resistance to nodes compromising, and the proposed group-based deployment scheme for resilience against nodes compromising, where only nodes of the newly deployed generation ask for key establishment. In addition, the proposed mechanism for determining the highest deployed generation, guarantee that nodes will respond only to the newly deployed nodes' requests, and limiting the duration of

key establishment makes it practically impossible for an attacker to succeed in establishing keys in the network. Moreover, our protocol supports detection of silent attackers (intruders) and can be enhanced to achieve a distributed revocation. In a future work, we'll implement our protocol to evaluate its real resiliency to nodes compromising, and extend it with a distributed revocation mechanism.

References

1. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y.: Wireless sensor networks: a survey. Computer Networks 38 (2002) 393–422
2. Chan, H., Perrig, A., Song, D.: Random Key Predistibution Schemes for Sensor Networks. In IEEE Symposium on Security and Privacy. Okland California USA (2003) 197–213
3. Liu, D., Ning, P., Du, W.: Group-Based Key Pre-Distribution in Wireless Sensor Networks. In Proc. of the 4th ACM workshop on Wireless security. Cologne Germany (2005) 11–20
4. Liu, F., Major Rivera, J. M. , Cheng, X.: Location-aware Key Establishment in Wireless Sensor Networks. In Proc. of the International Conf. on Wireless Communications and Mobile Computing. Vancouver Canada (2006) 21–26.
5. Zhu, S., Setia, S., Jajodia, S.: LEAP: Efficient Security Mechanisms for Large Scale Distributed Sensor Networks. In Proc. of the 10th ACM Conf. on Computer and Communications Security. Washington DC USA (2003) 62–72
6. Bhuse V., Gupta A., Pidva R. (2003) A Distributed Approach to Security in Sensornets. IEEE Vehicular Technology Conference: 3010-3014.
7. Blundo, R., Suntis, A. D., Herzbeg, A., Kutten, S., Vaccaro, U., Yung, M.: Perfectly secure key distribution for dynamic conferences. In Proc. of the 12th Annual International Cryptology Conference on Advances in Cryptology. Springer-verlag, UK (1992) 471–486
8. Yu, Z., Guan, Y.: A Robust Group-based Key Management Scheme for Wireless Sensor Networks. IEEE Wireless Communications and Networking Conference. New Orleans USA (2005) 1915–1920
9. Dutertre, B., Cheung, S. , Levy, J.: Lightweight Key Management in Wireless Sensor Networks by Leveraging Initial Trust. SDL Technical Report SRI-SDL-04-02, SRI International (2004)
10. Perrig, A., Szewczyk, R., Wen, V., Cullar, D., Tygar, J. D.:"Spins: Security protocols for sensor networks". In Proc. of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking. Rome Italy (2001) 189-199
11. Liu, D., Ning, P.: Establishing Pairwise Keys in Distributed Sensor Networks. In the Proc. of the 10th ACM Conference on Computer and Communication Security. Washington DC USA (2003) 52–61.
12. TmoteSky wireless sensor module. http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf
13. DH Key-Excchange Protocols. https://www.cs.tcd.ie/courses/baict/bass/4ict11/Cou sewo k/4ICT11MT6.2.pdf
14. Karlof, C., Wagner, D.: "Secure routing in wireless sensors networks: attacks and countermeasures". Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols (2003)
15. Dimitriou T., Krontiris I. (2005) A Localized, Distributed Protocol for Secure Information Exchange in Sensor Networks. In Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium.