A New Protocol for Securing Wireless Sensor Networks Against Nodes Replication Attacks

Chakib Bekara and Maryline Laurent-Maknavicius

Institut National des Télécommunications d'Evry
Département Logiciel-Réseaux
9 rue Charles Fourier, 91000 Evry Cedex, France
Email: {Chakib.Bekara, Maryline.Maknavicius}@int-evry.fr

***Abstract*—The low-cost, unattended nature and the capability of self-organizing of sensors, yield the use of Wireless Sensor Networks (WSN) very popular to day. Unfortunately, the unshielded nature of sensors, their deployment in remote open (hostile) areas, and the use of wireless transmission medium, make them subject to several kind of threats and attacks, like eavesdropping, intrusion, Deny of Services (DoS) attacks and nodes compromising. While most of threats and attacks can be prevented using cryptographic materials (i.e. shared pair-wise secret keys, certificates, etc.) provided by key management protocols, some other threats, like nodes replication attacks, can still go undetectable. Nodes replication attacks are harmful attacks, where an attacker compromising a node, uses its secret cryptographic key materials to successfully populate the network with several clones of it, in-order to gain the control over the network or disturb the normal operation of the network. Several nodes replication detection protocols were proposed in the literature, but unfortunately, they require either a high computation, transmission and energy overheads, or that nodes know their exact locations coordinates, which limits their usability in most WSN scenarios. In this paper, we present a new protocol for securing and preventing against nodes replication attacks in static WSN, which requires no knowledge of nodes deployment locations, and introduces no significant overhead on the resource-constrained sensors.**

***Index Terms*—WSN security, Nodes replication attacks, Key management protocols, Pubic key cryptography**

I. Introduction

The spread deployment of WSN during the few last years, introduces several security considerations. Sensors are resource-constrained tiny devices, with small memory storage capacities (10 KB of RAM and 48 KB of ROM), low computation capacities (16-bit and 8 MHZ CPU), and extremely limited energy supply (3.6 V) [1], which is in general neither rechargeable nor replaceable [2]. In addition, sensors are unshielded devices [3], work unattended [3], and are generally deployed in remote locations assimilated as hostile areas [3] [4]. All these facts, yield WSN target to different attacks like eavesdropping, compromising, intrusion, deny of service (DoS), and nodes replication attacks. In replication attacks, an attacker first compromises a sensor from the network, and then populate the network with clones of it, using the secret materials (secret cryptographic keys, etc.) it retrieves from the compromised node. The aim of such attack is to have the control over the network, by compromising only few legitimate nodes.

Different protocols for nodes replication detection were proposed in the literature [5] [6], but unfortunately, most of them are not suitable for WSN, because of their heavy computation and transmission overheads due to the use of public key cryptography, and due to their dependency on nodes location coordinates, which means that sensors are either GPS-enabled, or that they must trust some beacon nodes placed on the perimeter of the network to compute their exact locations.

In this article, we present a new protocol for securing and preventing against nodes replication attacks on static WSN, where any deployed cloned node can be detected once it attempts to establish pair-wise keys with legitimate neighbor nodes of the network, thus protecting legitimate nodes from communicating with it or relaying its packets, and protecting our network from being penetrated. Our protocol doesn't rely on nodes locations for achieving detection of replication attacks, it uses only symmetric cryptography, and so keeps computation and transmission overheads as low as possible.

The remainder of the paper is organized as follows. In section 2, we overview some works on detection of nodes replication attacks. In section 3 we describe our assumptions, network deployment model, adversary model and the adopted notations. In section 4 and section 5 we describe our proposed protocol, and we give a detailed security analysis of it in section 6. In section 7 we give the memory and computation overheads of our protocol, and we conclude our work in section 8.

II. Related Work

In what follows, we describe two main protocols of the literature, which deal with the problem of nodes replication attacks. The first protocol we describe, achieves a localized detection of nodes replication attacks, where the second protocol achieves a totally distributed detection of nodes replication attacks. Unfortunately, the two protocols are based on the knowledge of sensors locations, and the use of public key cryptography.

A. A Location-based mechanism to detect nodes replication attacks

In [6], *Zhang et al.* propose the use of location-based keys to thwart and defend against several attacks, including nodes replication attacks. Their protocol is based on the use of a bilinear map [7] along with the use of Identity-Based Cryptography [8], in addition to the capability of sensors to retrieve their exact location coordinates once they are deployed.

Initially, the BS which is a widely trusted entity in the network, defines a bilinear map *e* over two cyclic groups G_1, G_2 of the same prime order *Q*. We define over G_1 an additive operation, while we define over G_2 an multiplicative operation. *e*: $G_1 \times G_1 \rightarrow G_2, \forall R, S, \in G_1$ and $a, b \in Z$, we have

$$e(aR, bS) = e(R, abS) = e(abR, S) = e(R, S)^{ab} \quad (1)$$

In general G_1 represents the set of points of an elliptic curve defined over *GF(Q)*, where G_2 represents a set of integer of a prime order *Q*.

Then, the BS chooses a master secret key $k \in \{1, ..., Q-1\}$ and two public hash functions, $H_1 : \{0,1\}^* \rightarrow G_1$, and a second hash function H_2 like SHA-1.

During the initialization phase before nodes deployment, the BS loads each sensor *u* with a unique identity-based private key $IK_u = kH_1(Id_u)$, where Id_u is both the identity and the public key of u. Like any other public key cryptography, it's mathematically infeasible to retrieve the identity-based private key IK_u from the public key Id_u.

Once nodes are deployed, some trusted mobile robots, either ground or flying, are dispatched over the deployment area of sensors. The mobile robots are trusted by all nodes of the network, are able to compute the private identity-based key of any node on the network, they have GPS capabilities, as well as more computation and communication capacities than regular sensors. Mobile robots are capable of collectively or independently determining the exact geographic location of each individual sensor. Each sensors *u* in the network, receives from the nearest trusted mobile robot an encrypted message containing its unique geographic position $Pos_u =< x_u, y_u >$ and its unique location-based key $LK_u = kH_1(Pos_u)$. The message is encrypted using node *u*'s identity-based key IK_u. Henceforward, each sensor *u* of the network will use its position Pos_u as its unique identifier (instead of Id_u) and its public key, and use its location-based key LK_u as its private key, where it's mathematically infeasible to deduce LK_u from Pos_u.

Before establishing a pair-wise key, a pair of neighboring nodes *u* and *v* must first pass through a phase of location-based node-to-node authentication. Node *u* sends to node *v* its identifier and a nonce value, $Pos_u =< x_u y_u >$ and N_u respectively. Node *v* checks that *u* in its transmission range, by verifying that

$$(x_v - x_u)^2 + (y_v - y_u)^2 \leq R^2 \quad (2)$$

where $Pos_v =< x_v,\ y_v >$, and R is the communication range of any node in the network. If (2) is satisfied, v sends to u its position Pos_v, a nonce value N_v and an authenticator V_v to authenticate itself to u, calculated as

$$V_v = H_2(e(LK_v, H_1(Pos_u)||N_u||N_v||0) \quad (3)$$

Upon receiving v's replay, *u* checks first that *v* is within its communication range (a neighbor), and then authenticates v using the characteristic of the bilinear map (see (1)), by verifying that

$$V_v = H_2(e(H_1(Pos_v), LK_u)||N_u||N_v||0) \quad (4)$$

If equality (4) is verified, node u authenticates itself to v, by sending its authenticator V_u calculated as

$$V_u = H_2(e(H_1(Pos_v), LK_u)||N_u||N_v||1) \quad (5)$$

Upon receiving u's reply, node v can easily verify that u is an authenticated neighbor by verifying the following equality

$$V_u = H_2(e(LK_v, H_1(Pos_u))||N_u||N_v||1) \quad (6)$$

If both equalities (5) and (6) are verified, nodes u and v will locally compute a symmetric secret pair-wise key

$$K_{uv} = e(H_1(Pos_u), LK_v) = e(LK_u, H_1(Pos_v)) \quad (7)$$

As we can see, *Zhang et al.* protocol achieves a strong security, by preventing an attacker from injecting any node in the network, either a cloned node, or a node with a false identity. To prove its identity, which is also its unique location coordinates, a node must have possession of the corresponding location-based key, otherwise it can never pass the location-based node-to-node authentication phase, as seen in (3), (4), (5) and (6). However, the protocol rely highly on the trust of the mobile robots that provide sensors with their position and location-based keys, and rely also on the properties of bilinear map and identity based cryptography, which are mathematically based on elliptic-curves cryptography (ECC) [9]. Deploying mobile robots is not evident especially if the covered area is large, and their compromising is possible resulting in the broken of the entire network security. In addition, implementing a bilinear map protocol and identity-based cryptography protocol based on elliptic curve cryptography is memory and energy consuming, even if ECC is less energy consuming then traditional public key. Indeed, according to *Malan et al.* work [10], implementing an ECC-163 bits protocol, requires approximately 34 MB of ROM, and 1,1 MB of RAM.

B. Distributed detection of nodes replication attacks

In [5], Parno et al. presented two protocols for distributed detection of nodes replication attacks in WSN, called respectively Randomized Multicast, and Line-Selected Multicast. The two protocols rely on the assumptions that each node of the network knows it geographic coordinates location, and that sensors are able to perform public key cryptographic operations (generate signatures). Each node in the network possesses a pair of private/public keys generated by a trusted entity in the network (BS), where each node's public key is certified by the public key of the trusted entity.

In Randomized Multicast protocol, each node *N* in the network generates a signed message, using its private key, called location claim containing its identity and its location coordinates, and locally broadcasts it to its *d* one-hop neighbors. Each neighbor, with probability *p*, checks the authenticity of the message, and then verifies that *N* is within its communication range, and finally forwards the location claim to *g* randomly selected nodes in the network, called the witness nodes. Upon receiving node *N*'s signed location claim message, the witness nodes verify the authenticity of the message, and store it if verified. A replication attack is detected whenever a collision (two distinct position claims for the same node identity) happens at a witness node, which will flood the entire network with the two conflicting location claims in-order to revoke the compromised node and all its clones. The authors showed that if *d*=20, *g*= 100, the probability *p*= 0.05, and a compromised node is replicated twice, the replication attack will be detected with a probability greater than 95%.

The Line-Selected Multicast protocol, uses the fact that sensors act as routers to reduce the communication overhead of the previous protocol. In this protocol, nodes in the path from each neighbor forwarding the location claim of a node *N*, to the randomly chosen witnesses (as described above), stores the location claim of *N*. As consequence, each path from a neighbor of *N* to a witness node (for *N*) constitutes a line. If a conflicting location claim for a node *N* crosses a line, the node of intersection, which belongs to two paths towards two whiteness nodes for *N*, will detect it, and consequently floods the network with the two conflicting claims. In this scheme, conflicting location claims can be detected on road (the point of intersection of two lines), and not only at the witness nodes as in the randomized multicast protocol, thus reducing transmission overhead.

As we can see, Parno et al. protocols are based on the assumptions that nodes are able to know their geographic location coordinates, and are able to generate digital signatures. To know their geographic locations, nodes (or some of them) must have either GPS receivers, or must rely on trusted base stations placed on the perimeter of the network [11] in order to compute their geographic coordinates. The first solution is not too realistic and is highly energy consuming, and the second solution requires trust on at least three BSs (triangulation process), and the resulting localization is prone to errors. The capability of sensors to handle public key cryptography is severely critiqued [4] [3]. First, signature generation is energy consuming [4], and the transmission of a signature over the network is highly energy consuming [4]. Second each node must send its certificate along with its location claim, in order to allow its related whiteness nodes and the nodes in the paths to verify the authenticity of the location claim. As described in [12], an X.509 RSA-1024 bits certificate is at least 262 bytes, and an ECC-160 bits certificate is at least 86 bytes. Consequently, transmitting a certificate is highly energy consuming, and is memory consuming too, especially that each node can be chosen to be a witness for several nodes in the network.

III. Background

In what follows, we describe our assumptions, our network model, the used notations, and the supported adversary model.

A. 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 a group-based deployment of nodes, where sensors are deployed progressively in successive generations (groups). This assumption is adopted in several previous works like [13] and [14], that used a group-based deployment model. However, unlike [13] and [14], we do not require that nodes of the same generation to be deployed in the same neighborhood. In our protocol, nodes of the same generation might be deployed anywhere in the network (randomly scratched). Therefore, our protocol is not based on any prior knowledge on deployment location of nodes. Note here that unlike [5], in our protocol, nodes don't need to have any knowledge about their geographic location coordinates.

Fourth, we suppose that once a node is deployed in the network, it needs at most a time T_{est} to establish pair-wise keys with its one-hop neighbors. Moreover, an attacker needs at least a time T_{comp} in order to compromise a node after it is deployed in the network, with $T_{comp} > T_{est}$. This assumption is present in several key management protocols for WSN like [15], [16] and [17], and is likely to be true, because an attacker must first have a physical access to a sensor, connect to it, and then use some programming tools in order to extract sensor's secret key materials.

Finally, 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 [4].

B. Network deployment model

The BS deploys nodes in multiple generations numbered successively from 1 to n, where n is the maximum number of deployed generations. We take $n < 2^{16} - 1$, so each generation number is exactly two bytes length. 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. A generation $j + 1$ is deployed after that the nodes of the previous deployed generation j have finished the establishment of their pair-wise keys.

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

TABLE I
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, with an output length of 4 bytes
$a\|\|b$	a concatenated to b
$\mid x \mid$	The length on bytes of argument x

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 a cloned node having the Id and the cryptographic secret key materials 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 stops key establishment process with any deployed node of an older generation, and rejects any key establishment request originating from a node of the same newly deployed generation.

C. Adversary model

We consider that an adversary has the capability of capturing and compromising a limited number of legitimate nodes of the network. After compromising a legitimate node, the adversary clone the node by loading the node's cryptographic secret key materials on multiple generic sensor nodes, and deploy the cloned nodes in some strategic locations in the network. Once cloned nodes are deployed by the adversary, they first try to establish secure links with their neighbors, in-order to sent packets, and participate in the network operations as any other legitimate node in the network. Once the cloned nodes are integrated in the network, the adversary and the cloned nodes can collaborate to lunch different attacks against the network. We consider also, that any cloned node has at least one or more legitimate nodes in his neighborhood.

D. Notations

For clarity, we list the symbols and notation used throughout in Table 1.

IV. OUR PROPOSED PROTOCOL

Now we describe our protocol for detection of nodes replication attacks. The basis of our protocol, is the use of symmetric polynomial for pair-wise key establishment [18] and our defined group-based deployment model. The main idea of our protocol, is to tie each deployed node to the unique generation (or group) to which it belongs, through the use of symmetric polynomial, so that even if cloned nodes are created, the cloned nodes also belong to the same generation as the compromised node. In our protocol, only newly deployed nodes (which belong to the newly deployed generation) are able to establish pair-wise keys with their neighbors, and all nodes in the network know the number of the highest deployed generation. As consequence, an attacker compromising an old deployed node (which belongs to an old deployed generation), cannot succeed to populate the network with cloned nodes of it, because the cloned nodes will fail to establish pair-wise keys with their neighbors.

A. Initialization

Initially (before nodes deployment) the BS generates a random symmetric bivariate polynomial [18]

$$f(x,\ y) = \sum_{i,j=0,\ldots,t} a_{ij} \times x^i y^j \pmod{Q'} \qquad (8)$$

where Q' is a large prime number, $1 \leq a_{ij} \leq Q' - 1$, and t is the degree of the polynomial and a security parameter. We suppose that $\mid x \mid = \mid y \mid = 4$ bytes.

For any newly deployed generation G_i, the BS loads each node $u \in G_i$ with its unique secret polynomial share:

$$f_u(y) = f(H(i||Id_u),\ y) \qquad (9)$$

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 $H(i||Id_u) = (j||Id_v)$, which means $i||Id_u = j||Id_v$. 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 $Id_u \neq Id_v$.

B. Protocol description

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 V.

Let $u \in G_j$ a newly deployed node. It is obvious that as a legitimate 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 $K_{vu} = f_v(H(i+1||Id_u))$ 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 III-B), 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 $K_{uv} = f_u(H(z||Id_v))$, and checks the message authenticity. If the message is not authenticated, node v simply rejects the message. 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 cloned node or a false node with non-existing identifier. 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 VI.

V. Determining the highest deployed generation

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

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 to 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.

TABLE II
Identified scenarios for key establishment according to the generation of concerned nodes

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

VI. Security analysis of our protocol

In our protocol, and under our assumptions of section III, an attacker is highly unlikely to deploy cloned nodes, and convince their neighbors of their validity.

Table 2 summarizes the different scenarios for attempts of deployment of a cloned node. As a cloned node must first establish pair-wise keys with its neighbors in-order to integrate the network, four possible cases of attacks can be defined depending on the generation to which belongs the cloned node, and if the cloned node is a *requesting node* or a *responding node* in the pair-wise key establishment process.

First, let see how our protocol handles the two last cases $Old - New$, and $Old - Old$, where a cloned node u of an old deployed generation asks for key establishment with a node of a newer generation then it, or an older generation. Remember that only nodes of the newly deployed generation are able to ask for key establishment with their neighbors. As a consequence, a cloned node u of an old deployed generation ($u \in G_i$), can not initiate key establishment with another deployed node $v \in G_j$ where $j > i$. In addition, the mechanism described in section V guarantee that all nodes of the network have the same view of the number of the highest deployed generation, so the cloned node $u \in G_i$ 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 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}, where $T_{comp} > T_{est}$, the attacker cannot establish secure links with other nodes of the same generation, simply because the responding nodes will reject the request, as described in section IV-B.

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 deployed 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 IV-B does not handle this situation. This case is difficult to detect, because the cloned node looks like a legitimate 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 V, nodes know at any moment, the number of the highest deployed generation, and when a new generation will be deployed. Consequently, each already deployed node sets a timer to the value $T_{max} < T_{comp}$ 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 or not. The problem is even more difficult if the cloned node stays inactive or silent until a newly node is deployed. At this time, the cloned node might become active and establish a secure link with the newly deployed node 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 (cloned 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.

A. 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.

VII. 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 IV-A). A polynomial share is represented by *t+1* coefficients, plus the modulo *Q'*. If we choose a modulo *Q'* of 8 bytes, as in [19], 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 [19], 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 [1], handles 16-bit words 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 64-bit prime number of , and y is 48-bit (6 bytes) length (see section IV-A), requires $4 \times t$ modular additions, and $8 + 24 \times (t-1)$ modular multiplications.

Due to the use of symmetric key cryptography only, our protocol has less computation overhead when performing encryption or authentication operations then *Parno et al.* protocols [5], which based on the use of public-key cryptography, and even less then *Zhang et al.* [6] protocol which is based on elliptic curves cryptography. Even when generating a pair-wise key, the polynomial-based key generation protocol has less computation overhead, because we have no modular exponentiation as in traditional public key cryptography, and we have no scalar point multiplication as in elliptic curve cryptography.

For the memory cost due to the initially stored key materials, our protocol may have some additional cost then the other protocols. Indeed, as described above, each node needs to store a key materials (polynomial share) of 800 hundred bytes. In *Parno et al.* protocols, and in the case we use public key elliptic curves cryptography, each node needs to store at least 163-bit (21 bytes) private key, the prime modulo *Q* which is at least 163-bit length too, its certificate of 86 bytes length, and the BS's certificate which is 86 bytes length too. However, Parno et al. protocols consume heavy energy during transmission, because each node must broadcast in the network its certificate of 86 bytes along with its location claim to allow the verification of its signature, and the signature is approximately 42-byte length. In *Zhang et al.* protocol, each node stores a prime modulo *Q* of 163 bits (over which the elliptic curves are defined), its 42-byte identity-based private key, its location-based private key of 42 bytes, and its position, which the length depends on the type of the used coordinates.

VIII. Conclusion

We proposed in this paper, a new protocol for detection of nodes replication attacks, which do not assume that sensors know their geographic location coordinates, which do not consider trusted entities other then the BS, and which do not assume that the tiny constrained-resource devices are able to perform public-key cryptographic operations.

Our protocol uses t-degree polynomial-based key generation protocol for pair-wise keys establishment, and the proposed group-based deployment scheme for protection against nodes

replication attacks, where only nodes of the newly deployed generation ask for key establishment. In addition, the proposed mechanism for determining the highest deployed generation, guarantees 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, as consequence, an attacker is unable to deploy cloned nodes 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 performances, and extend it with a distributed revocation mechanism.

ACKNOWLEDGEMENTS

Authors are thankful to French ANR (Agence Nationale de la Recherche) agency which funded RNRT project CAPTEUR (2005-2008).

REFERENCES

[1] TmoteSky wireless sensor module. http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf

[2] I. F. Akyildiz, W. Su and Y. Sankarasubramaniam, "Wireless sensor networks: a survey", Computer Networks (38), pp. 393-422, 2002

[3] C. Karlof, and D. Wagner, "Secure routing in wireless sensors networks: attacks and countermeasures", Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols, 2003

[4] A. Perrig, R. Szewczyk, V. Wen, D. Cullar and J. D. Tygar, "Spins: Security protocols for sensor networks", In Proc. of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking, Rome, Italy, pp. 189-199, 2001

[5] B. Parno, A. Perrig and V. Gligor, "Distributed Detection of Node Replication Attacks in Sensor Networks", Proceedings of the IEEE Symposium on Security and Privacy, Berkley, California, USA, pp. 49-63, 2005

[6] W. Zhang, W. Liu, W. Lou and Y. Fang, "Securing sensor networks with location-based keys", IEEE Wireless Communication and Networking Conference, New Orleans, USA, pp. 1909-1914, March 2005

[7] J. Baek, J. Newmarch, R. Safavi-Naini and W. Susino, "A Survey of Identity-Based Cryptography". http://www.math.uiuc.edu/ du-ursma/Math595CR/KimJ.pdf

[8] D. Boneh and M. Franklin, "Identity-based encryption from Weil pairing", SIAM Journal of Computing, vol.32, no.3, pp. 586-615, 2003.

[9] D. Hankerson, A. J. Menzes and S. A. Vanstone, "Guide to ellipric curve cryptography", Springer professional computing, Springer, New york, ISBN: 0-387-95273-X.

[10] D. J. Malan, M. Welsh and M. D. Smith, "A Public-Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography", IEEE SECON, Santa Clara, CA, USA, October 2004

[11] N. Bulusu, J. Heidemann and D. Estrin, "GPS-less low-cost outdoor localization for very small devices", IEEE Personal Communication Magazine, 2000

[12] A. S. Wander, N. Gura, H. Eberle, V. Gupta and S. C. Shantz, "Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks", Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications, Kauai Island, Hawai, pp. 324-328, March 2005

[13] D. Liu, P. Ning and W. Du, "Group-Based Key Pre-Distribution in Wireless Sensor Networks", In Proc. of the 4th ACM workshop on Wireless security, Cologne, Germany, pp. 11-20, Septembre 2005

[14] Z. Yu and Y. Guan, "A Robust Group-based Key Management Scheme for Wireless Sensor Networks", IEEE Wireless Communications and Networking Conference, New Orleans, USA, pp. 1915-1920, March 2005

[15] B. Dutertre, S. Cheung and J. Levy, "Lightweight Key Management in Wireless Sensor Networks by Leveraging Initial Trust", SDL Technical Report SRI-SDL-04-02, SRI International, 2004

[16] S. Zhu, S. Setia, S. Jajodia, "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, pp. 62-72, October 2003

[17] T. Dimitriou and I. Krontiris, "A Localized, Distributed Protocol for Secure Information Exchange in Sensor Networks", In Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium, Denver, Colorado, USA, April 2005

[18] R. Blundo, A. D. Suntis, A. Herzbeg, S. Kutten, U. Vaccaro and M. Yung, "Perfectly secure key distribution for dynamic conferences", In Proc. of the 12th Annual International Cryptology Conference on Advances in Cryptology, Lecture Notes in Computer Science, vol. 17, Springer-verlag, pp. 471-486, 1992

[19] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks", In the Proc. of the 10th ACM Conference on Computer and Communication Security, Washington DC, USA, pp. 52-61, October 2003