Defending Against Nodes Replication Attacks on 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
{*chakib.bekara, maryline.maknavicius*}*@int-edu.eu*

Wireless Sensor Networks (WSN) are subject to several kind of threats and attacks, like eavesdropping, intrusion, battery exhausting, packets reply and nodes compromising. While most of threats can be dealt with them through cryptographic materials provided by key management protocols, some other threats, like nodes replication attacks, can still go undetectable. Nodes replication attacks are one of the most redoubtable attacks, where an attacker compromising a node, uses its secret cryptographic key materials to successfully populate the network with clones of it. Few nodes replication defending protocols were proposed in the literature, but unfortunately, they require either a high computation, transmission and energy overheads, or that nodes know their exact location coordinates, which limits their usability in most WSN scenarios. In this paper, we present a replication detection protocol for static WSN, which requires no knowledge of nodes deployment locations, and introduces no significant overhead on the resource-constrained sensors.

Keywords: WSN Security, Intrusion Detection, Replication Attacks, Key Management Protocols

1 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) [MOT], which is in general neither rechargeable nor replaceable [ASS02]. In addition, sensors are unshielded devices [KW03], work unattended [KW03], and are generally deployed in remote locations assimilated as hostile areas [KW03] [PSWCT01]. 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 [PPG05] [DCL04] [LMC06], but unfortunately, most of them are not suitable for WSN, because of there heavy computation and transmission overheads due to the use of public key cryptography, and due to there 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 detection of nodes replication attacks on static WSN, where any deployed cloned node can be detected once it attempts to establish pair-wise keys with legitimate nodes of the network, thus protecting legitimate nodes from communicating with it 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 present 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.

2 Related Work

2.1 Parno et al. protocols

In [PPG05], *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), store 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 [BHE00] 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. First, signature generation is energy consuming, and the transmission of a signature over the network is highly energy consuming. 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 [WGEGS05], 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.

2.2 Dutertre et al. protocol

Dutertre et al. [DCL04] propose a protocol for pair-wise key establishment, which provides a degree of defense against nodes replication attacks. The protocol suppose that nodes are deployed in n successive

generations, and can not be compromised in a period of time less than $T_{comp} > T_{est}$, where T_{est} is the maximum time a newly deployed node needs to establish pair-wise keys with its neighbors, and T_{comp} is the minimum time an attacker needs to compromise a node of the network. Each generation is loaded with a unique two master keys, used respectively for authentication and key generation between the nodes of the same generation. Once a deployed node successfully establishes secure links with its neighbors of the same generation, it permanently 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 of a newly deployed generation i to establish secure links with previously deployed generations $j = 1,\ldots,i-1$. GK_i is also deleted at the end of the key establishment phase. 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 protocol provides a low defense against nodes replication attacks, as an attacker compromising a node u of generation i, can populate the network with clones of u and establish secure links with nodes of the future deployed generations, using the compromised secret keys Su_j. Moreover, if an attacker succeed in compromising 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.

2.3 Liu et al. protocol

Liu et al. [LMC06], propose a distributed location-aware key establishment protocol that achieves a high protection against nodes replication attacks under the assumptions made by the authors. 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. Services 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), and they remain static. After deployment, each service node creates a distinct t-degree secret bivariate polynomial (see 4.1), and then securely initializes each neighbor node with its secret polynomial share, using the unique location coordinates of the node. Two nodes initialized by the same server and being in the transmission overhead of each other, can directly establish a pair-wise key. As long as the service nodes are not compromised, the protocol guarantees that an attacker compromising an arbitrary number of sensors in the network, with at most t nodes initialized by the same server node, can not launch nodes replication attacks. However, if a service node is compromised, which is a current threat because a service node is just a non tamper resistant sensor, an attacker can inject clones and new nodes with new positions, and deploy them in the neighborhood of the compromised service node, to establish pair-wise keys with other nodes initialized by the same compromised server node. In addition, the protocol suppose that nodes know their exact location once deployed. This means that nodes have either GPS receivers, which is highly energy consuming, or must trust some GPS-equiped nodes to provide them with their location, which introduces some trust issues.

3 Background

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 a group-based deployment of nodes, where sensors are deployed progressively in successive generations (groups). This assumption is adopted in several previous works like [LND05] and [YG05], that used a group-based deployment model. However, unlike [LND05] and [YG05], 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 [PPG05], 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 [DCL04], [ZSJ03] and [DK05], 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 [PSWCT01].

3.2 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 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.

3.3 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.

3.4 Notations

For clarity, we list the symbols and notation used throughout the paper below:

4 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 [BSHKVY92] and our defined group-based

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

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.

4.1 Initialization

Initially (before nodes deployment) the BS generates a random symmetric bivariate polynomial [BSHKVY92]

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

where Q is a prime number, $1 \leq a_{ij} \leq Q-1$, and t is the degree of the polynomial and a security parameter, and $\mid x \mid = \mid y \mid = 4$ bytes.

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

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

A t-degree polynomial has the property of t-resistant, which means that an attacker needs to compromise at least t+1 nodes in-order to recover the bivariate polynomial the BS creates.

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 $Id_u \neq Id_v$.

4.2 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 5.

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 3.2), 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 6.

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 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.

Tab. 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*

6 Security analysis of our protocol

In our protocol, and under our assumptions of section 3, 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 5 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 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 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 4.2 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 5, 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.

6.1 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.

7 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 [LN03], 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 [LN03], 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 [MOT], 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 48 bits (6 bytes) length (see section 4.1), requires $4 \times t$ modular additions, and $8 + 24 \times (t-1)$ modular multiplications.

One can easily notice, that our protocol have less memory, computation and even transmission overhead, then *Parno et al.* protocols [**?**] that are based on the use of public-key cryptography.

8 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, and 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, 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, 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.

References

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

[ASS02] I. F. Akyildiz, W. Su and Y. Sankarasubramaniam. Wireless sensor networks: a survey. Computer Networks (38), pp. 393–422, 2002

[KW03] 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

[PSWCT01] 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, pp. 189-199, 2001

[PPG05] 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, pp. 49–63, 2005

[DCL04] 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

[LMC06] F. Liu, J. M. Major Rivera and X. Cheng. Location-aware Key Establishment in Wireless Sensor Networks. In Proc. of the International Conf. on Wireless Communications and Mobile Computing. pp. 21–26, 2006

[BHE00] N. Bulusu, J. Heidemann and D. Estrin. GPS-less low-cost outdoor localization for very small devices. IEEE Personal Communication Magazine, 2000

[WGEGS05] 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, pp. 324–328, 2005

[LND05] 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, pp. 11–20, 2005

[YG05] Z. Yu and Y. Guan. A Robust Group-based Key Management Scheme for Wireless Sensor Networks. IEEE Wireless Communications and Networking Conference, pp. 1915–1920, 2005

[ZSJ03] S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient Security Mechanisms for Large Scale Distributed Sensor Networks. In Proc. of the 10th ACM Conf. on Computer and Communications Security, pp. 62–72, 2003

[DK05] 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, 2005

[BSHKVY92] 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. Springer-verlag, UK, pp. 471–486, 1992

[LN03] 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, pp. 52–61, 2003