The Cloning Attack Vulnerability in WSN Key Management Schemes

Othmane Nait Hamoud
Ecole Militaire Polytechnique
Algiers, Algeria
o_naithamoud@esi.dz

Tayeb Kenaza
Ecole Militaire Polytechnique
Algiers, Algeria
ken.tayeb@gmail.com

Nadia Nouali-Taboudjmat
CERIST
Algiers, Algeria
nnouali@cerist.dz

***Abstract*—The majority of existing key management schemes in wireless sensors networks suffer from the physically compromised of nodes. This vulnerability allows an adversary to reproduce clones from a compromised node and to inject them into several locations on the network to conduct other types of attacks. Moreover, the joining of new nodes to the network (for maintenance), which is an inevitable step to prolong its life or to correct voids, presents the best opportunity to conduct such an attack. In this paper, we review several key management schemes and we highlight their weakness regarding the cloning attack.**

Keywords—WSNs; key management scheme; cloning attack; maintenance of WSNs.

I. INTRODUCTION

Wireless Sensor Networks (WSNs) present an effective solution in many areas (military, environment, etc.) thanks to their low cost manufacturing, their multiple usages and their ad hoc property. However, their deployment in open spaces, their transmission medium and the lack of infrastructure with a robust security, expose them to serious vulnerabilities.

To protect WSNs, several solutions have been proposed in the literature [12], where the central element is the key management scheme. This latter can be classified regarding the used encryption technique as symmetric, asymmetric, or hybrid. Each of these schemes, try to find a compromise between cost and efficiency, where their common goal is to allow each pair of neighboring nodes to establish a symmetric cryptographic key. These keys are then used to ensure the confidentiality of the exchanged data between nodes. However, the random deployment of a huge number of sensors renders unpredictable the resulting network topology. Therefore, one can never predict for each sensor its direct neighbors in order to pre-load it with the adequate Pair-wise Keys (PK), which makes the design of secure key management solutions a challenging task. Another difficulty lies in the authentication mechanism. This can be achieved through asymmetric cryptography, which is known for its requirements in terms of resources consuming. Hence, the convergence of the vast majority of key management approaches to the symmetric cryptography, when authentication is generally based on the pre-loading of sensors with a primary key.

However, these mechanisms suffer from physical sensors compromise. This vulnerability allows an adversary to reproduce clones from a compromised node and inject them in several places in the network (known as cloning or replication attack). These clones will then present themselves as legitimate nodes to infiltrate the network and then conduct other types of attacks [13]. Moreover, the joining of new nodes in the network (for maintenance) is an inevitable step to prolong its life and repair voids (not covered areas) or isolation of subnets that can arise both immediately after the initialization of the network or during its operational phase, presents the best opportunity to lead the cloning attack.

The remainder of this paper is organized as fellows. We introduce in Section 2 the cloning attack vulnerability in distributed key management schemes. In Section 3 we review some of these schemes where we highlight their weakness regarding the cloning attack. Section 4 concludes this paper.

II. CLONING ATTACK IN WSNS

To minimize the impact of the physical compromise of sensors in primary key based schemes, three alternatives have been proposed:

1. The primary key must depend on the geographic location of new nodes (localization based schemes) [5], [11]. In this case, the compromise of this key will not allow an adversary to inject its clones in locations other than those in which he compromised them.
2. The primary key must be checked by a trusted third party before joining a new node (centralized schemes) [6] [10].
3. The primary key must depend on time (time-based schemes) and/or on the use of one-way hash function [1], [2], [3], [4].

Generally, in localization based schemes, deployment is done in such way that each sensor has the coordinates of its location (known previously or given by a Global Positioning System (GPS)) or has its relative position regarding its neighbors in the case of deterministic deployment (e.g., grid deployment [7]). Therefore, information about the position of a sensor can be involved into the process of key establishment, which prevents from (or greatly limits) cloning attack in this category. In centralized schemes, the centralized control of maintenance operations can significantly reduce the impact of the cloning attack. However, these two categories present, respectively, several constraints regarding the deployment and the scaling.

Note that we are particularly interested in the third category (time-based schemes), since their schemes are the most cited in the literature. These schemes offer unlimited scalability, a relatively secure connectivity, and a decentralized management.

Unfortunately, these schemes are still subject to the cloning attack, where the impact is very important regarding to the effort made by an attacker to conduct such attack. Indeed, to infiltrate a WSN and to carry out more attacks, we need just to compromise a single sensor, extract the secret information from its memory and reproduce several clones, and finally inject them in several locations of the network [13]. These clones will be able to legitimately communicate with old nodes as new nodes, and to conduct the attack that we call "direct cloning" or, wait for the deployment of new nodes to present themselves to the latter as old nodes, and thus lead the attack that we call "indirect cloning" (Fig. 1).

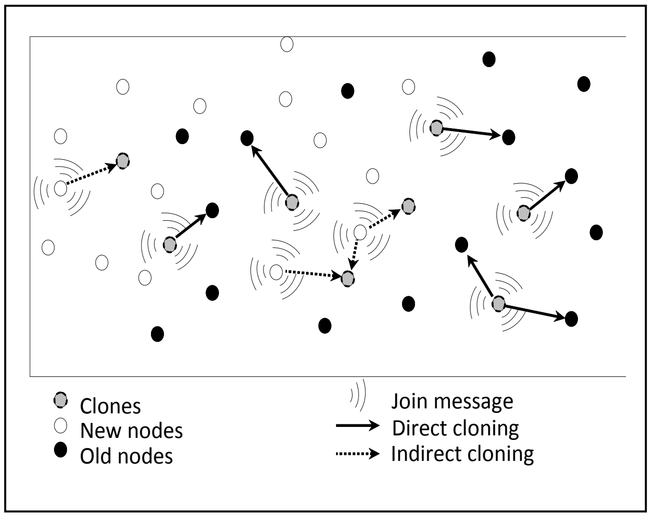

Figure 1. Direct and indirect cloning attack.

Time-based schemes, although they claim resisting to the cloning attack, they finally resist only to the direct cloning attack. Note that, to our knowledge, the indirect cloning has never been discussed in the literature.

III. Review Of Some Existing Time-Based Schemes

In this section, we will review some time-based key management schemes, where we particularly highlight their weakness regarding the cloning attack and its impact on the security of WSNs.

A. LEAP

Localized Encryption and Authentication Protocol (LEAP) [1] is one of the most referenced protocols in the literature. It presents a comprehensive key management scheme based on a primary key, under the assumption that this key will be erased after the network initialization. It has been designed to support multiple communication modes: unicast, broadcast, local and global broadcast.

In order to bypass the use of the BS in each operation of key establishment or authentication between nodes, authors of LEAP have based their scheme on a primary key K_{IN}, which is pre-loaded in nodes.

They assume that there is a time limit T_{min}, needed to compromise a node, and another time limit T_{est} required by a newly deployed sensor to discover its immediate neighbors, to establish PK with each of them and to erase K_{IN}. Zhu et al. [1] have proposed two variants of this scheme:

- Basic scheme

In this scheme, it is assumed that an attacker cannot recover K_{IN} before T_{min} (i.e., $T_{est} < T_{min}$). Before deployment, the BS pre-loads the primary key K_{IN} on a node u, which is then used to derive a primary key $K_u = f_{K_{IN}}(u)$ where f is a pseudo-random function. Once deployed, the node u broadcasts a message containing its identity u and initiates a timer, which will be triggered when T_{min} expires. The response of a node v includes its identity and the MAC (calculated using K_v) of u and v:

$$u \rightarrow * : u \qquad (1)$$
$$v \rightarrow u : v \mid MAC(K_v, u|v) \qquad (2)$$

The response of v can be only authenticated with the primary key K_v, which is derived as follows: $K_v = f_{K_{IN}}(v)$. As the node u has K_{IN}, it can also derive and verify the identity of v. Thus, the PK will be calculated as follows: $K_{uv} = f_{K_v}(u)$. Once T_{min} is expired, the node u erases K_{IN} and all primary keys of its neighbors, while it keeps its K_u.

- Extended scheme

In this scheme, authors assume that an attacker can recover K_{IN} before T_{est} (i.e., $T_{est} > T_{min}$). The basic idea in this case is to remove the dependency on K_{IN}, which will be replaced by a series of primary keys corresponding to time intervals (T_1, T2, ..., T_M) of new nodes joining. Before deployment, the BS pre-loads a node u to be deployed in the interval Ti with the primary key K_{IN}^i, through which it derives its primary key $K_u^i = f_{K_{IN}^i}(u)$. In addition, the node u is pre-loaded with primary keys $K_u^j = f_{K_{IN}^j}(u)$ for all $i < j \leq M$. Once deployed, the node u starts the neighbor discovery and the establishment of PK. In the neighbor discovery, a node u begins by sending a message containing its identity u and the current interval i:

$$u \rightarrow * : u \mid i \qquad (3)$$

As response, a neighbor v sends an acknowledgment Ack authenticated with the master key K_v^i of the current interval i:

$$v \rightarrow u : v \mid MAC(K_v^i, u|v) \qquad (4)$$

Since u has K_{IN}^i, it can calculate K_v^i and check the Ack. Now, nodes u and v can calculate their PK : $K_{uv} = f_{K_v^i}(u)$. Once T_{min} is expired, the node u erases K_{IN}^i and all primary keys of its neighbors, while it keeps its K_u^i and primary keys of future intervals j.

Discussion

LEAP authors address the cloning attack only in the case when a new node compromise would be made before T_{min}, while the cloning attack can be done even if the compromise was made after T_{min}. In the basic scheme, the maintenance operation exposes K_{IN} to compromise, which may annul the security of the entire network. Moreover, even if K_{IN} is not disclosed, the indirect cloning attack is possible against new nodes. Indeed, the primary key of a compromised node can be directly used by clones to authenticate theirs responses (2) to messages sent by new nodes during the neighboring discovery (1).

In the extended scheme, if the attacker compromises a node u after T_{est}, the primary key K_{IN}^{i} of a node u and its neighbors primary keys K_{v}^{i} are deleted, which prevents from the direct cloning attack. However, the attacker can recover primary keys K_{u}^{j} that correspond to future intervals j. Thus, he can clone the node u, and all copies will be able to present themselves as legitimate old nodes (4) to new nodes to be deployed in future intervals (indirect cloning).

For example, a node u is deployed at T_2, it is pre-loaded with K_{IN}^{2} (Through which it derives its primary key K_{u}^{2}), and M-2 master keys $K_{u}^{3}, \dots, K_{u}^{M}$, corresponding to intervals T3, ..., Tm. After T_{min}, the node u do not delete master keys K_{u}^{2} , K_{u}^{3} , …, K_{u}^{M} in order to authenticate nodes deployed in T_2,T_3, ..., T_M. So, the compromise of node u results clones with all these master keys. Consequently, any clone u*, deployed across the network will be able to authenticate itself to all sensors deployed at T_2, T_3, ..., T_M. Therefore, clones can easily establish PK and infiltrate the network (indirect cloning).

Note that LEAP authors have proposed a solution against the cloning attack in the case of a compromise before T_{est}. In this solution, they rely on the broadcast by the BS of an authenticated list of added nodes, and this using μTesla [6]. In this case, even if the network nodes establish PK with clones, they can revoke them by receiving the list of added nodes.

Although LEAP uses the BS as a trusted third party that broadcasts the list of added nodes, the vulnerability of the cloning attack still exists. The origin of this vulnerability is the lack of mutual authentication between nodes. Indeed, when an attacker compromises a node all constructed clones will have the same identity of the compromised node. In addition, the establishment of PK is based on the identity of nodes; it cannot be done with a clone that has an identity x and keys of node y. Therefore, the broadcast of the list of added nodes identities eventually brings nothing. In other words, as the compromised node is legitimate, so their identity and keys are legitimate also, except that clones are everywhere in the network. The problem is that the network will never identify them.

B. OTMK

Like LEAP, Opaque Transitory Master Key (OTMK) [2] is based on a primary key. However, the PK established between two nodes will not depend on the primary key; it will be randomly generated by both nodes. This ensures the security of existing links in case of the compromise of the primary key. To do this, each node is pre-loaded with a primary key K_{IN}, and the PK establishment is done as follows:

$$u \rightarrow * : \mathrm{Join}|E_{K_{IN}}(u|n_u) \tag{5}$$

$$v \rightarrow u : \mathrm{Reply}|E_{K_{IN}}(v|n_u + 1|K_{uv}) \tag{6}$$

For the maintenance operation, Deng et al. proposed a mechanism, which is also based on the decomposition of the life of the network into a number of time intervals. Each node is pre-loaded with an authentication key H. This key is obtained by applying a chain of one-way hash function $< H_k, H_{k-1}, \dots H_1, H_0 >$, where each key H_i corresponds to a time interval i. Nodes deployed in the same interval establish their PK as described previously (5 and 6), where the authentication key H_i plays the role of the primary key K_{IN}. Subsequently, each node v calculates its primary key $K_v = MAC(H_i, v)$ and calculates $H_{i-1} = f(H_i)$ and then removes H_i.

Let u be a new node and v an old node. The node u is pre-loaded with H_j and the node v with H_i (with j> i). When the node v receives the message JOIN from the node u, it responds with a message containing its identity v, a nonce n_v, the index i of the primary key and the associated MAC:

$$u \rightarrow * : \mathrm{Join} \mid n_u \mid u \tag{7}$$

$$v \rightarrow u : v \mid n_v \mid i \mid MAC(K_v, n_u|n_v|v) \tag{8}$$

Node u can compute H_i from H_j and the index i. Thus, it can generate K_v with H_i and check the MAC, and therefore authenticate v. Since v is authenticated, node u must also authenticate itself to v:

$$u \rightarrow v : u \mid E_{K_v}(K_{uv}) \mid MAC(H_j, n_v|u|K_{uv}) \tag{9}$$

Once PK is established, u broadcasts its primary key H_j. Therefore, v can authenticate u:

$$u \rightarrow * : H_j \tag{10}$$

Discussion

Although an adversary who compromises K_{IN} cannot compromise already established PK, he can intercept reply messages (6) and then disclose PK. Moreover, this protocol is not secure since listening to the traffic allows an adversary to save the messages exchanged during a time interval j (9), and then, decrypt them after the disclosure of the key H_j (10), which will cause the loss of the confidentiality in the network.

Regarding the cloning attack, an attacker can carry out a direct cloning attack against existing nodes through clones obtained by the compromise of a new node before it erases the primary key H_j. In addition, even if the compromise would be done after clearing H_j, the attacker can (through the index i and the corresponding primary key) conduct an indirect cloning attack against nodes added in the future time intervals T_i (i>j), and this, by replaying the same authentication (8).

C. TBMK

The probabilistic scheme Time-Based Key Management protocol for WSN (TBMK) [3] assumes that the time T_{est}

required to establish keys between nodes is greater than T_{min}, the time required to compromise a node. To reduce the impact of compromising the primary key K_I, authors have decompose the lifetime of the network in P time intervals T_i (corresponding to maintenance phases) when for each time interval corresponds a primary key. In addition, authors used a probabilistic distribution in the pre-loading of sensors, as used in [8].

To do this, the BS pre-loads nodes with a primary key K_i^k corresponding to their deployment time interval k, and a set of m random primary keys corresponding to the future intervals i ($K_{ui} = f_{K_{Ii}}(u)$). After deployment, like LEAP, the first key establishment (corresponding to the time interval T_1) is done through the key K_{I1}. A node u computes its key and broadcasts a message containing its identity and a nonce n_u. A node v responds with a message containing its identity and the MAC of $(n_u|v)$:

$$u \rightarrow * :\ u \mid n_u \tag{11}$$
$$v \rightarrow u :\ v \mid MAC(K_{v1}, n_u|v) \tag{12}$$

Having the key K_{I1}, a node u can generate the primary key of v and thus authenticate it. The PK is calculated as follows: $K_{uv} = f_{K_{v1}}(u)$.

For new nodes joining, those deployed at the same time will be able to establish PK as they have the same primary key. They can also establish PK with those deployed in previous time intervals if old nodes have among the randomly pre-loaded primary keys, a primary key derived from the key of the current interval. Fig. 2 presents an example given in [3]. N_n represents a group of nodes deployed at the time interval T_n. The group N_1 is pre-loaded with the primary key K_{I1} and 3 randomly selected primary keys (K_{U2}, K_{U5}, K_{U7}). They can establish PK between them, because they all have K_{I1}. As they can establish PK with the

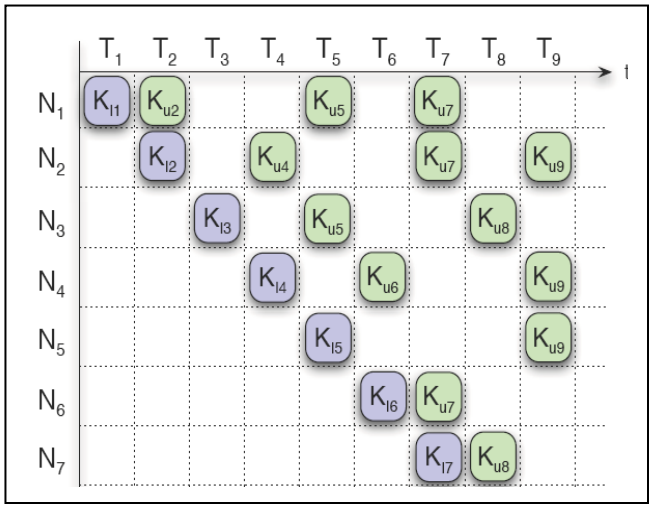

Figure 2. Example of probabilistic primary key pre-loading.

groups N_2, N_5 and N_7 because they have K_{U2}, K_{U5} and K_{U7}, respectively. Groups N_2, N_5 and N_7 can authenticate nodes belonging to the group N1 thanks to their respective primary key K_{I2}, K_{I5} and K_{I7}.

Discussion

Regarding the cloning attack, the compromise of a sensor that belongs to N4 can recover the primary key of the current interval K_{I4} and the primary keys K_{U6} and K_{U9} derived from the primary key K_{I6} and $K_{I9,}$ respectively. Thus, clones can, thanks to the primary key K_{I4}, establish PK with nodes deployed during the same time interval (i.e., nodes of N4) as well as those deployed during the interval T_2 (i.e., nodes of N2), which represent a direct cloning attack.

Even if the compromise was made after the erase of the primary key K_{I4}, primary keys K_{U6} and K_{U9} will never be erased, which will allow clones to establish PK with nodes to be deployed later during the time interval T_6 and T_9 (i.e., nodes of groups N_6 and N_9), which represents an indirect cloning attack.

D. TEKM

Towards Enhanced Key Management (TEKM) scheme [4] is based on the fact that a node u can live up to G_w generations (generation window), which correspond to the addition of new nodes. Before deployment, a node u belonging to the generation j is pre-loaded with a set key KR_j called "KeyRing" containing the primary key IK_j of the generation j and (Gw-1) hidden primary keys $K_{j,l}$ of the future generations ($KR_j = \{IK_j, K_{j,l}\}$). The hidden primary keys are calculated using a secure hash function:

$K_{j,l} = H(IK_l|j)$.

Such as : $j + 1 \leq l \leq j + G_w - 1$.

After deployment, nodes of a generation j have the same primary key IK_j, so it will be used (before being deleted) to establish the PK ($K_{u,v}^j$).

$$u \rightarrow * : u \mid j \mid n_u \tag{13}$$
$$v \rightarrow u\ : v \mid MAC\left(K_v^j, u|v\right) \tag{14}$$

where: $K_v^j = f_{IK_j}(v)$ (15)

$$K_{u,v}^j = \begin{cases} f_{K_u^j}(v) \text{ si } u < v \\ f_{K_v^j}(u) \text{si } u > v \end{cases} \tag{16}$$

For nodes that do not belong to the same generation, PK is calculated differently. Indeed, let u and v be two nodes that belong to the generation g and h respectively ($1 \leq g < h \leq g + Gw - 1$). After deploying the generation h, the PK K_{uv}^{gh} between u (old) and v (new) will be calculated as follows: $K_{uv}^{gh} = f_{K_{gh}}(u|v)$. The node u is already pre-loaded with the key K_{gh}, since its "KeyRing" is as follows: $KR_g = \{IK_g, K_{g,g+1}, \dots, K_{g,h}, \dots, K_{g,g+Gw-1}\}$. For the node v, K_{gh} is the hidden primary key of the group deployed in the

TABLE I. COMPARISON BETWEEN THE STUDIED KEY MANAGEMENT SCHEMES

	Using primary key in		Security			Performance		
	Authentication	PK establishment	Resilience	Cloning		Memory	MAC	Transmission
				Direct	Indirect			
Basic LEAP	Yes	Yes	Good Bad*	Yes* No	Yes	1 key (K_{IN})	Yes	2 local transmissions * d + 1 broadcast from the BS
Extended LEAP	Yes	Yes	Good Bad *	Yes* No	Yes	1 key (K_{IN}^{i})+ (M-i) keys	Yes	
OTMK	Yes	No	Good Bad *	Yes* No	Yes	1 key (Hi)	Yes	2 local transmissions * d
TBKM	Yes	Yes	Good Bad *	Yes* No	Yes	1 key (K_{IN}^{i}) + m random keys	Yes	2 local transmissions * d + Proxy
TEKM	Yes	Yes	Good Bad *	Yes* No	Yes	G_w keys	Yes	2 local transmissions * d

*. When the primary key is compromised. d. the average number of neighbors.

previous generation g, and it's not in its "KeyRing". However, it can get it by calculating $K_{gh} = H(IK_h|g)$ as it holds the key IK_h of its generation.

Discussion

When the attacker compromise a new node v belonging to the generation h, and if he recovers the initial key before it is erased, he can use it to calculate the hidden primary key ($K_{xh} = H(IK_h|x)$) of a future generations x, and so to calculate PK of nodes belonging to previous generations (direct cloning).

Moreover, even if the primary key is erased, it can use the hidden primary key contained in its memory to calculate PK of nodes of future generations (indirect cloning).

IV. GENERAL DISCUSSION

The physical compromise of sensors can be exploited in two ways: (1) by disclosing the shared secret used by network nodes to setup PK, thus the existing links can be compromised (e.g., the K_{IN} in basic LEAP is used to calculate all PK) or (2) by reproducing clones that will allow an attacker to infiltrate the network and then to conduct other types of attacks. Both exploits seriously degrade the resilience of a key management scheme. Recall that a perfect resilience refers to the ability of a key management solution to prevent an attacker to not compromise any further communication link other than those involving compromised nodes.

Table 1 compares the discussed protocols in the previous section with respect to security and performance metrics. All schemes use a primary key in the authentication process, so if this key will be compromised, this will allow the attacker to insert malicious nodes. So, all these schemes are vulnerable to the cloning attack, especially the indirect cloning attack. In addition, in all schemes (except OTMK), the PK are also based on the primary key. So, its compromise leads to disclose of all PK already established, which weighs heavily on confidentiality. In LEAP if an attacker succeeds to compromise a node deployed in a time interval before T_{min}, then confidentiality is not guaranteed since the attacker can compute master keys of nodes deployed in the same time interval. In addition, if the attacker compromises a node after T_{min}, it can recover master keys of the future time intervals. Thus, it can clone it and all copies will be able to present themselves as legitimate nodes to new nodes even if they cannot communicate with the existing nodes of the same time interval.

Although the protocol OTMK is very complex, it is not secure, since listening to the traffic allows an attacker to back up the encrypted messages exchanged during a time-slot j and decipher them after disclosure of the key Hj. On the other hand, it can lead an attack cloning by replaying the same authentication through its own nodes in several places in the network.

The problem of time-based key management schemes, regarding the cloning attack, lies on the authentication mechanism between the nodes. This mechanism is generally based on some secret information pre-loaded on sensors to build confidence between nodes. Unfortunately, this latter is set to fail by the physical compromise of nodes, allowing the disclosure of the shared secret. Being aware of this vulnerability and wanting to keep the distributed aspect of their key management schemes, works [1], [2], [3], [4], [14], [9] introduced techniques such as time intervals, hash function, digital signature, etc. to limit the impact of the compromising of sensors. However, these solutions are still vulnerable to the cloning attack as shown in the previous section, even if their assumptions are satisfied.

V. CONCLUSION

In this paper, we presented the weaknesses of time-based protocols, especially regarding the maintenance operation, which is a critical phase in the life of a WSN. The problem lies in the legitimacy of joining new nodes to the network on one hand, and on the other hand, to the physical compromise of sensors that allows an attacker to have clones and inject them into the network. Therefore, the problem is difficult, since new nodes might need to legitimately join the network at times, raising the question of how to perform authentication while defending against cloning attacks. Thus, it is essential to involve the BS or to depend the primary key to the geographic position of nodes to ensure perfect resilience during the maintenance phase.

REFERENCES

[1] S. Zhu, S. Setia, and S. Jajodia, “Leap+: Efficient security mechanisms for large-scale distributed sensor networks,” ACM Trans. Sen. Netw., 2002, 2(4), pp. 500-528.

[2] J. Deng, C. Hartung, R. Han, and S. Mishra, “A practical study of transitory master key establishment for wireless sensor networks,” Proc. of the 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm 05), Athens, Greece, September 2005. pp. 289–299.

[3] J. Jang, T. Kwon, and J. Song, “A time-based key management protocol for wireless sensor networks,” in Information Security Practice and Experience, 2007, pp. 314–328.

[4] B. Tian, S. Han, L. Liu, S. Khadem, and S. Parvin, “Towards enhanced key management in multi-phase ZigBee network architecture,” in computer communication, 2012. 35(5): pp. 579-588.

[5] A. Fanian, M. Berenjkoub, H. saidi, and T. A. Gulliver, “A high performance and intrinsically secure key establishment protocol for wireless sensor networks,” Comput. Netw, 2011, vol. 55, pp. 1849-1863, doi:10.1012/j.comnet.2011.01.012.

[6] A. Perrig, R. Szewczyk, V, Wen, D, Cullar, and J. D. Tygar, “SPINS: Security protocols for sensor networks,” Proc. of the 7th annual ACM/IEEE international conference on mobile computing and networking, July 2001, pp. 189–199.

[7] Y. S. Han, W. Du, S. Chen, and P. K. Varshney, “A key management scheme for wireless sensor networks using deployment knowledge,” Proc of IEEE INFOCOM04. Hong Kong: IEEE Press; 2004. pp. 586–597.

[8] L. Eschenauer and B. Gligor, “A key-management scheme for distributed sensor networks,” Proc of the 9th ACM conference on computer and communication security, Washington, DC, USA, 2002. pp. 41–47.

[9] S. Blackshear, M. Rakesh, and Verma, “R-LEAP+: Randomizing LEAP+ key distribution to resist replay and jamming attacks,” in: SAC’10 March 22, 2010, Sierre, Switzerland.

[10] D. Manivannan and pp. Neelamegam, “WSN: Key issues in key management schemes-A review,” Research Journal of Applied Sciences, Engineering and Technology 4(18): 3188-3200, 2012.

[11] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili, “A pairwise key pre-distribution scheme for wireless sensor networks,” ACM T. Inform. Syst. Se., 2005. 8(2): pp. 228-258.

[12] J, Zhang, and V. Varadharajan, “WSN key management and taxonomy,” Journal of Network and Computer Application 33 (2010) pp. 23-75.

[13] B. Parno, A. Perrig, and V. Gligor, “Distributed detection of node replication attacks in sensor networks,” Proc. of the 22th IEEE symposium on security and privacy (S&P’05); May. 2005. p. 49–23.

[14] O. Delgado-Mohatar, A. Fúster-Sabater, and J. M. Sierra, “A light-weight authentication scheme for wireless sensor networks,” Elsivier, Ad Hoc Networks 9 (2011) pp. 727–735.