A Review of Distributed Dynamic Key Management Schemes in Wireless Sensor Networks

Seyed Reza Nabavi[1*], Seyed Morteza Mousavi[2]

[1] Young Researchers and Elite Club, Arak Branch, Islamic Azad University, Arak, Iran.
[2] Department of Computer Engineering, Arak Branch, Islamic Azad University, Arak, Iran.

* Corresponding author. Tel.: +989188629049; email: s.rezanabavi@hotmail.com
Manuscript submitted June 25, 2016; accepted December 28, 2016.
doi: 10.17706/jcp.13.1.77-89

Abstract: wireless sensor networks include small-sized sensor nodes with limited resources (energy, memory, etc.). Considering this limitation and their placement in unsupervised locations, e.g. military areas, and since this type of networks does not rely on a fixed infrastructure, security becomes a complex and considerable issue in these networks. Providing data and communication security requires appropriate encryption protocols. Key management is one of the most important mechanism used to secure encryption keys and their safe distribution among sensor nodes. Recently, many key management schemes have been proposed, each with its own particular strengths, weaknesses, and applications under certain circumstances. This paper investigates special requirements of distributed dynamic key management schemes in wireless sensor networks. Subsequently, modern distributed dynamic key management schemes are compared and finally, several relevant recommendations are made.

Key words: Wireless sensor networks, distributed dynamic key management, security.

1. Introduction

A wireless sensor network consists of a large number of battery supplied sensor nodes, as well as sensor components, data processing, and short-range radio communications [1]. This type of networks has applications from environmental surveillance and intelligent homes to sensitive cases in military or security areas, including surveilling battlefields, targeting, and tracking.

Security in wireless sensor networks is challenging since they do not rely on a fixed infrastructure. These networks include wireless connection, close interaction of sensor nodes, unsupervised operations without human intervention and lack of physical protection. Therefore, wireless sensor networks are vulnerable against an extensive set of attacks at network level, as well as physical damages [2]. Although sensor nodes can possess built-in anti-tampering mechanisms, memory chipsets are vulnerable to various damages of reading their memories.

Key management is a fundamental mechanism to insure security in network services and wireless sensor network applications. Key management can be defined as a set of processes and mechanisms, which support key establishment and key related interactions between authorized nodes, which is based on security policy making [3]. Since sensor nodes have limited computational power and memory capability, security approaches designed for contingency and wired networks are not appropriate for wireless sensor networks. key management in wireless sensor networks aims to solve the problem of creating, distributing, and maintaining a secret key. Therefore, techniques for reliable key distribution and management are highly

important in the security of wireless sensor networks.

Due to their importance, key management systems have always attracted the interest of researchers and an increasing regard in the scientific literature. Recently, many key management schemes have been proposed for wireless sensor networks [4]-[8]. These schemes can be categorized into static and dynamic methods based on their capability in updating the encryption key at runtime (rekeying) [9].

static key management is limited to key pre-distribution and keys are constant during the network`s lifetime. However, if an encryption key is used for a long period, it becomes significantly more likely to be attacked. In contrast, dynamic key management regenerates keys during network`s lifetime. Therefore, dynamic key management is known to be more promising in wireless sensor networks. dynamic key management is a set of processes used to rekeying, whether periodically or per network`s demand. Since the keys of compromised nodes are revoked during the rekeying process, dynamic key management schemes highly improve the survivability and resilience of wireless sensor networks.

Generally, depending on whether a centralized key controller is involved in generating or distributing a new key, almost all dynamic key management techniques can be divided into distributed or centralized schemes. This paper only discusses distributed key management schemes.

2. Basic Requirements and Evaluation Metrics

Dynamic key management can be considered a branch of key management. All key management schemes should satisfy the following traditional security requirements: confidentiality, authentication, freshness, integrity, and non-repudiation. These requirements should be satisfied by dynamic key management schemes as well. Moreover, features and the application environment of the dynamic key management scheme highlight certain evaluation measurements. Therefore, this section defines the most common metrics used to evaluating dynamic key management in wireless sensor networks. In [10], evaluation metrics to manage key pre-distribution are divided into security, efficiency, and flexibility. This categorization is based on the constraints of sensor nodes and networking limitations. Accordingly, basic requirements and evaluation metrics are proposed based on this categorization and the unique features of dynamic key management. We must note that among the following metrics, node revocation, forward and backward secrecy, collusion resistance and key connectivity are only applicable for dynamic key management schemes and other metrics can also be applied to static key management schemes.

2.1. Security Metrics

Dynamic key management schemes should provide secure encryption keys, such that the activities of malicious nodes are prevented within the network. When a compromised sensor node is detected, the current secret key of the compromised sensor node must be revoked and a new key generated and distributed among related nodes. The dynamic key management scheme should possess forward and backward secrecy, as well as collusion resistance between newly joined and compromised nodes. Moreover, resilience should be provided after capturing and replication nodes.

Some security metrics of dynamic key management schemes are as follows:

- Node revocation: when compromised sensor nodes are detected, an effective approach must immediately eliminate (invalidate) them from the network. Such mechanisms are used to prevent compromised nodes from deviating the network`s behavior by injecting false data or manipulating the data of trusted nodes.
- Forward and backward secrecy: forward secrecy is used to prevent a node from using an old key to decrypting new messages [11]. On the contrary, backward secrecy is used to prevent a node with a new key from returning to a previous state and decipher previously received messages, which are encrypted with prior keys [11]. Both forward and backward secrecy are used to defeat

node capturing attacks.

- Collusion Resistance: an intruder may compromising several nodes to attack the network and force them to collude and collaborate to control all systems keys and eventually, the entire network. A good dynamic key establishment technique should resist the collaboration of newly joined and compromised nodes.
- Resilience: resilience is a measure that indicates how much resistance there is against node capturing. It means that this factor measures the impact of a captured node on the rest of the network when an intruder attacks a physical sensor node and tries to recover secret information from its memory. Recovery of a network is high if the intruder cannot affect any node other than the captured one. On the contrary, resilience is low if capturing one node leads to compromise the entire network.

2.2. Efficiency Metrics

The number of exchanged messages to change a key, the number of required encryption keys, the amount of operations, and the size of encryption keys should be as low as possible. This prevents limiting available nodes` energy resources and storage capacity, as well as network size. Since nodes inherently have limited resources, the dynamic key distribution itself should be impose a large load as follows:

- Memory: the mount of required memory to store security credentials, e.g. keys (public or private, pairwise keys), user certificate (e.g. ID), and trusted certificates (e.g. neighboring nodes' reputation).
- Bandwidth: the number and size of messages exchanged in the key generation process, node replenishment, and node eviction.
- Energy: the consumed energy in the process of key agreement, transmission, and data reception, as well as the computational procedure of generating and distributing new keys.

2.3. Flexibility Metrics

Key deployment techniques should be flexible enough to perform well in an extended set of wireless sensor network applications. The most important flexibility metrics include:

- Mobility: most network topologies assume that sensor nodes are stationary. However, mobility of base stations, sensor nodes, or both are required in certain applications [12]. Therefore, key deployment should distribute new keys to moved nodes, so that they can communicate with their new neighbors. Key generation and distribution are more challenging for mobile nodes, since in addition to energy and bandwidth, mobility capacity also becomes an important issue.
- Scalability: the number of sensor nodes deployed in a region may be hundreds to even thousands. Moreover, during the network lifetime, sensor nodes can be joined or removed. Therefore, dynamic key management techniques should be scalable to different network sizes. While, security and efficiency features in small networks should be maintained when applying to larger networks.
- Key connectivity: key connectivity is defined as the probability that two (or more) nodes can deploy a key after rekeying it. Local connectivity refers to the connectivity of each pair of neighbor nodes. In contrast, global connectivity means connecting to the entire network. In order to provide continued security, high key connectivity is required after each rekeying process.

3. Distributed Dynamic Key Management Schemes in Wireless Sensor Networks

This section discusses main distributed dynamic key management schemes, which are proposed to date for wireless sensor networks and specifies security and efficiency specifications. Depending on different

basic encryption operations, based on which existing distributed dynamic key management schemes are proposed, they can be divided into the following three groups [9]. Fig. 1 presents the categorization of distributed dynamic key management schemes.

Distributed dynamic key management is a set of processes in which there is no centralized key controller, such as a base station or third party, involved in the rekeying process by sensor nodes. In contrast, key management is performed by multiple controllers, which can be predetermined or dynamically assigned.

Distributed dynamic key management schemes prevents a single point of failure and allows better network scalability. However, they are prone to design errors, since captured sensor nodes can participate in the node eviction process.

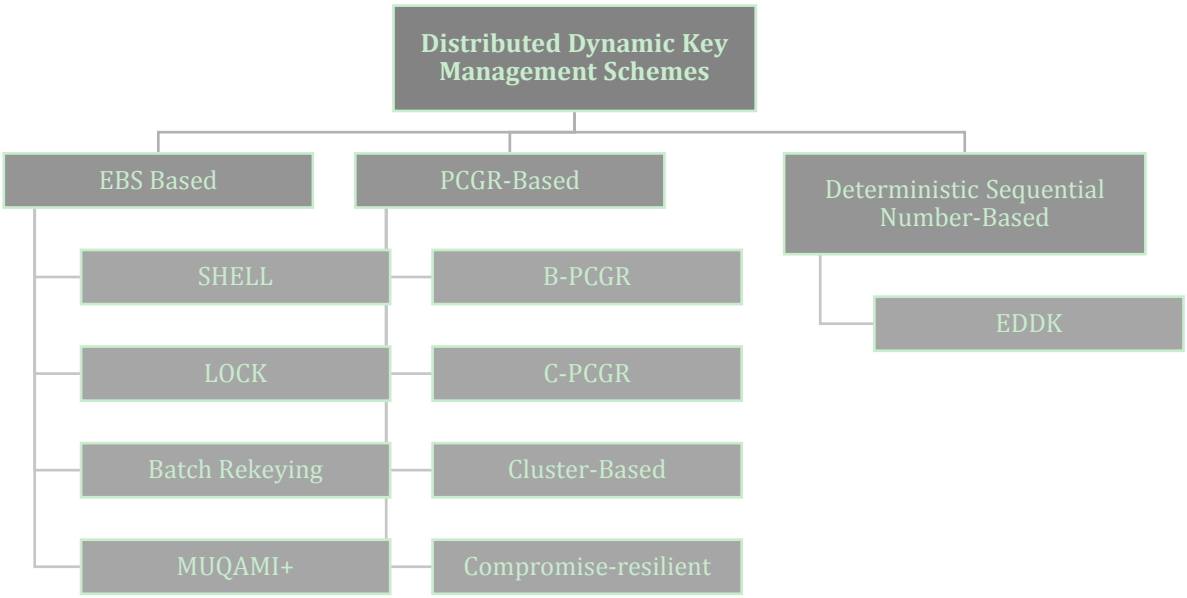

Fig. 1. Distributed dynamic key management schemes.

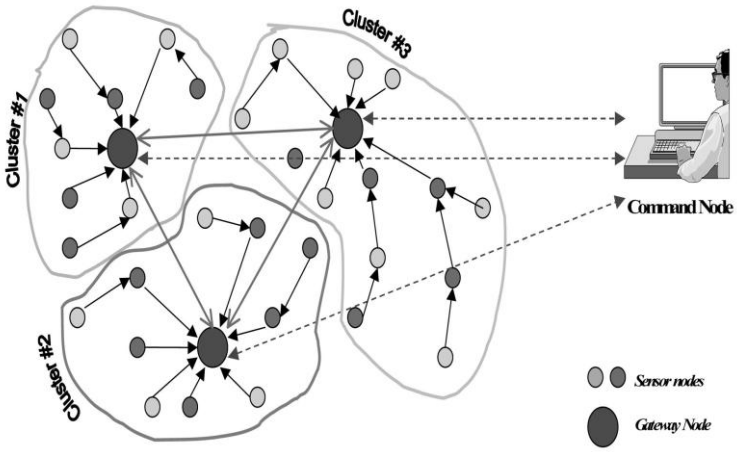

Fig. 2. SHELL key management scheme.

3.1. EBS Based Schemes

Exclusion Basis System (EBS) [13] is a combinatorial formulation of the key group management problem in wireless sensor networks. In EBS based schemes, k keys of a set with size $P = k + m \ \ (m < n. 1 < k)$ is assigned to each node n_i, where n is the number of sensor nodes in the network. Rekeying is called periodically or when one or more keys are captured (or are suspected of being captured). In the rekeying process, replacement keys are generated and encrypted with all the m keys, which are unknown to the

captured node. Finally, these keys are distributed to other nodes. In total, these nodes are aware of these m keys [14].

An EBS based distributed key management scheme which is scalable, hierarchical, efficient, location-aware, and light [15]. SHELL groups sensor nodes in clusters C_i ($0 < i \leq n$, where n is the number of nodes in a cluster). Rekeying is only performed within these clusters. The network considered in SHELL consists of command nodes, cluster heads (CHs), gateways, and sensor nodes. Fig. 2 presents this scheme.

It is assumed that the command node is resources-rich and cannot be compromised. After bootstrapping the gateway, SHELL determines how many administrative keys are required for a cluster. Subsequently, the list of nodes and the EBS table, which includes key combinations, are sent to the command node. Command node determine a number of gateways for each cluster C_i to generate administrative keys. After generation, the key generation gateway encrypts these keys and sends them to head cluster $G_{ch}[i]$. The head cluster decrypts these messages and sends their content to the cluster member nodes.

In order to refresh communication keys, cluster head $G_{ch}[i]$ sends new communication keys to its two key generation gateways. These gateways encrypt these keys with the administrative keys of sensor nodes and send the encrypted message to $G_{ch}[i]$. Finally, $G_{ch}[i]$ forwards new communication keys to the nodes of its cluster. Administrative keys can also be changed in a similar way. When a gateway is compromised, regeneration can be managed by whether deploying a new gateway or redistributing cluster nodes with the compromised gateway among safe clusters. If a sensor node fails or compromised in a cluster, data keys should be changed for the entire cluster. The EBS-based rekeying process to eliminate compromised sensor nodes. However, EBS-based key management is vulnerable to collusion attacks [16]. This is particularly true when the value of m is selected to be relatively small (to minimize rekeying traffic). SHELL proposes a heuristic key assignment to prevent collusion and increase the number of colluding nodes to capture the entire network. Nodes with physically smaller distances are assigned key combinations with lower Hamming distance. In order to prevent assigning key combinations with a large Hamming distance to two neighbor nodes with the same parent, key combinations are exchanged between the considered node and any node with predetermined combinations.

Despite the successful rekeying and collusion prevention of, SHELL also poses some weaknesses. First, the structure and operations are very complex, since there are various node operations and several types of keys (about 7) are used. Due to complexity, energy consumption and encryption overhead is higher than those of other schemes. Second, it is assumed that nodes have specific physical locations. However, location information in high density networks are not necessarily available. Third, by swapping key combinations, it is more likely to have two neighbor sensor nodes with the same parent to have the same key combinations, since these combinations have minimum Hamming distance. This increases the number of colluding neighbor nodes. Therefore, capturing a small number of nodes exposes most keys and leads to capturing the entire network [7]. Hamming distance based dynamic key management is proposed to solve this problem. Finally, SHELL exploits a centralized key generation gateway to perform rekeying. therefore, capturing key generation gateways provides a larger number of sensor nodes to the intruder in comparison to when ordinary sensor nodes are captured.

Localized Combinatorial Keying (LOCK) is another EBS-based dynamic key management scheme. This wireless sensor network model consists of a three level hierarchy, the base station at the top, then cluster leaders (CLs), and finally, regular sensor nodes. Fig. 3 presents the LOCK key management scheme.

LOCK employs no location information in generating new keys. In the initialization phase, sensor nodes of each cluster create a set of backup keys, which are shared with the base stations and are hidden from the cluster-leader. When a node is captured, other nodes of that cluster rekeying their keys by a local rekeying

mechanism, such that the compromised node cannot communicate with them. If the compromised node is a cluster head, the base station initiates rekeying at the cluster head level. Similarly, nodes in a cluster governed by the compromised cluster leader are rekeyed with the base station. In contrast to other dynamic key management schemes, in LOCK, capturing a node (including the cluster leader) has no effect on the common operations of other clusters. Since, nodes locally rekeyed, LOCK can reduce time delay and energy consumption used for the cluster key generation and renewal.

Ref. [17] proposed a batch key scheme, which defines three operations for member nodes, including join, leave with collusion-resistance, and leave with collusion-free.

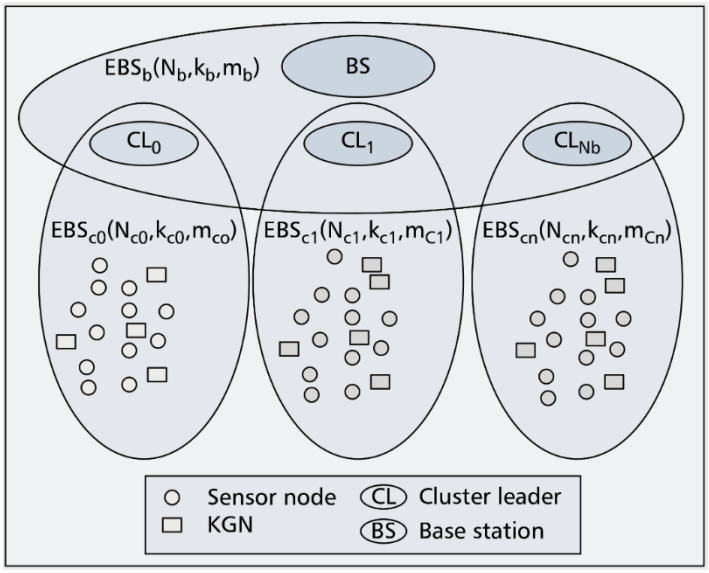

Fig. 3. The LOCK key management scheme.

3.2. PCGR Based Schemes

Polynomial Secret-Sharing-Based Rekeying (PCGR) [18] is a family of collaboration-based and pre-distribution group rekeying, which was proposed to solve the problem of node compromise. In PCGR-based scheme, sensor nodes are randomly assigned to several groups and the nodes of each group share a unique key. Accordingly, the two Basic PCGR-based (B-PCGR) and Cascading PCGR-based (C-PCGR) rekeying schemes were proposed. In B-PCGR, neighbors in a one-hop distance must collaborate to protect the polynomials of their group`s key. However, if a node like u and a number of its one-hop neighbors based on a specific threshold ($u + 1$, where u is a parameter picked by u) are compromised, the group key polynomial is revealed. In the C-PCGR, the share of a node is distributed among its multi-hop neighbors, was proposed to address the above mentioned limitation of B-PCGR and achieve higher resistance against node compromise.

In the B-PCGR, before the deployment of sensor nodes, a setup server determines the total number of groups and every node is preloaded with the group key polynomial (a unique s-degree univariate g-polynomial) of its group. After detecting neighbors, sensor u randomly picks a bivariate e-polynomial (called encryption polynomial, e.g. $e_u(x.y)$) to encrypt its g-polynomial, (e.g. $g(x)$), which result in the encrypted g-polynomial: g'-polynomial (e.g. $g'(x)$). Encryption is performed in form of $g'(x) = g(x) + e_u(x.y)$. After encryption, node u distributes shares of $e_u(x.y)$ to its direct neighbors v_i, $(0 \leq i < n$ where n is the number of neighbors). Specifically, each neighbor v_i receives a share $eu(x.v_i)$ from node u. after the shares distribution process, each node u keeps $g'(x)$ in its memory and removes $eu(x.y)$ and $g(x)$.

The PCGR is assumed that to be loosely time synchronized and group rekeying is lunched by each node

periodically. Each node keeps a variable called the current group key version (denoted as c), which is initialized as 0 when the node is deployed. When it is time to update group keys, each innocent node v_i increases its current group key version by one, and returns its share $e_u(c.v_i)$, to each trusted node u.

After receiving $\mu + 1$ shares, which are denoted as $\{\langle v_i.e_u(c.v_i)\rangle.i = 0.\ldots.\mu\}$, u constructs a unique μ-degree polynomial, and finally, computes the new group key $g(c) = g'(c) - e_u(c.u)$.

The C-PCGR was proposed based on B-PCGR-based, however, it differs from B-PCGR in the encryption and distribution of the polynomial shares and key updating phase. In the C-PCGR, the e-polynomial shares of a sensor node u is distributed to its multi-hop neighbors, while e-polynomial shares are distributed or collected in a cascading way. By using a polynomial-based signature and authentication [18], PCGR can effectively detect false shares injected by an adversary.

The PCGR schemes have five important weaknesses. First, each node has to store the secret shares of its direct neighbors e-polynomials, PCGR may not be applicable to networks with a dense deployment of sensor nodes. Second, although PCRGR can reduce the amount of traffic and relieve traffic congestions by employing a return probability for its neighbors` share [18], the communication consumption is still too high to be applied to large-scale networks. Third, if a certain number of compromised sensor nodes continuously return false polynomial shares to a sensor node $u$, a DoS attack can be lunched. Fourth, if a large fraction of a sensor node`s neighbors are compromised and new sensor nodes are not deployed around it, this sensor node may not be able to update its group key and thus be isolated. Fifth, in order to allow nodes to send information during the updating process, the previous group key is still used to communication among sensor nodes during a threshold time interval after the rekeying operation. In other words, a revoked node can still use the previous group key to decrypt new messages. Therefore, forward secrecy is not guaranteed.

In [19], proposed a cluster-based key management scheme. Instead depending on the collaboration of neighboring nodes to acquire the group key, this scheme lets CHs generate and distribute the group key to the nodes within the cluster. The total number of groups is determined by the sink node. For each group, a unique 2t-degree bivariate polynomial $g(x.y)$ is constructed, and each node (including the CH) u is loaded with its personal secret $g(u.y)$. After deployment, the algorithm proposed in [20] is adopted to help with CH election and cluster formation. Each CH uses an one-way hash function and the identifier (ID) of its member nodes to construct hierarchical keys, which are then unicasted to its corresponding member nodes. When a CH wants to derive the current group key, it broadcasts a request message to the nodes within its cluster. When this request message is received, each node u, willing to trust the CH returns its personal secrets $g(u.y)$, where y is the current group key version. After successfully receiving t personal secrets, CH derives the current group key by following the threshold secret sharing scheme [21]. Subsequently, CH unicasts the current group key to its members using the hierarchical keys.

When a compromised CH is detected, the sink node informs the members of the compromised cluster to begin a new CH selection process. If a normal node A is compromised, first its CH deletes the hierarchical key shared with it, then, the CH of the compromised node sends a group rekeying message to other CHs in the same group to invoke the key update process. [19] addressed the problem of node isolation suffering in [18] and [22]. When a CH cannot reconstruct the group polynomial, it sends a cluster dissolving message to the sink node and its innocent members. Subsequently, it deletes all its hierarchical keys with its members, and then unicasts a joining message (including all IDs of its innocent members) to other CHs. However, when a new node is added, it may use an absolute group key, as the new node is not synchronized with the group rekeying process. On the other hand, without time-synchronization, the group key used for communication between two arbitrary nodes may be inconsistent. Moreover, it does not indicate how to select the parameter t appropriately to achieve the desired level of security and reliability. Furthermore, a

normal node under the control of an adversary may return false shares to the CH.

Using the analysis results of the rekeying protocol [22], [23] proposed a robust pairwise rekeying protocol for hierarchical wireless sensor networks to prevent node capture attacks. The characteristic of the perturbation polynomial [24] is exploited for the proposed rekeying scheme. In [23], the rekeying protocol can be divided into three phases, system initialization, pre-distribution of perturbed polynomials and key establishment and rekeying.

In the system initialization phase, a large prime number q and the minimal integer l, which satisfy $2^l > q$, are chosen based on the size of the network. An offline authority arbitrarily constructs a bivariate symmetric polynomial $f(x.y)$. Subsequently, the method described in [24] is used to construct the legitimate ID set S for sensor nodes and generate the perturbation polynomial set Φ. Regarding a positive integer r $(r < l)$, any polynomial $\Phi(x)\epsilon\phi$ must satisfy:

$$\forall x \in S.\phi(x) \in 0.\ldots.2^r - 1 \tag{1}$$

In the pre-distribution of perturbed polynomials phase, before deployment, each cluster head l needs to be loaded with a unique ID, $CH_l \in S$ and a perturbed polynomial:

$$\overline{g}_{CH_l}(y) = f(CH_l.y) + \phi_{CH_l}(y) = g_{CH_l}(y) + \phi_{CH_l}(y) \tag{2}$$

Similarly, each sensor node (SN) m is also preloaded with a unique ID $SN_m \in S$ and a perturbed polynomial:

$$\overline{g}_{SN_m}(y) = f(SN_m.y) + \phi_{SN_m}(y) = g_{SN_m}(y) + \phi_{SN_m}(y) \tag{3}$$

In addition to the unique ID and the perturbed polynomial, all sensor devices (SNs or CHs) are equipped with an one-way hash function, $H^k(x)$, the returned hashed value of which is based on the most significant k bits of x.

In [23], the main pairwise key establishment is treated as a rekeying process. After the construction of the hierarchical network, CH_l randomly generates a new t-degree univariate rekeying polynomial function $P_{CH_l}(y)$, at the begin of each rekeying phase. For each of its member sensor nodes SN_m, CH_l updates the corresponding pairwise key $K_{CH_l.SN_m} = H^{l-r}(P_{CH_l}(SN_i))$. At the same time, CH_l uses the newly generated $P_{CH_l}(y)$ and the preloaded polynomial $\overline{g}_{CH_l}(y)$ to construct a master polynomial:

$$w_{CH_l}(y) = P_{CH_l}(y) + \overline{g}_{CH_l}(y) \tag{4}$$

Subsequently, the CH broadcasts its ID and the master polynomial $w_{CH_l}(y)$ to all of its member nodes.

After receiving the broadcast message, each SN_m evaluates the preloaded polynomial $\overline{g}_{SN_m}(y)$ and the received $w_{CH_l}(y)$ separately. During the evaluation, three candidate keys are calculated [23]. An encoded message $E(msg.K_{CH_l.SN_m})$ is sent using piggybacking from CH_l to SN_m. For SN_m, one of the previously calculated candidate keys that can successfully decode this message will be determined as the new pairwise key.

In comparison with traditional polynomial based rekeying protocols, the proposed scheme is achieve higher robustness against node capture attacks. Performance analysis results in this work show that their proposed protocol outperforms [22] in terms of the total computation complexity and total communication overhead. Compared to [22], the compromised-resilient pairwise rekeying scheme achieves both forward and backward secrecy. However, it does not indicate a method to revoke captured or compromised nodes, especially for the CHs. When a CH is captured, it may lunch a revocation attack, resulting in significant traffic congestion and energy depletion. When the pairwise keys are updating, nodes may not be able to

send data to their CH, since the pairwise keys known to them (nodes and their CH) may be inconsistent. Moreover, this scheme does not indicate how to establish pairwise keys between newly deployed nodes and the existing ones.

3.3. Deterministic Sequence-Number Based Schemes

After analyzing OMTK [5], which is vulnerable to resource exhausting attacks and DoS attacks, [25] proposed an energy-efficient distributed deterministic key management scheme (EDDK) to securely establish and maintain the pairwise keys and the local cluster key. It assumes each node A, before deployment, is loaded with a network-wide pseudorandom function f and a network-wide initial key K_l. Also, node A has a local cluster key which is shared with all its neighbors. The initial key K_l is used to compute the node's individual key, from which a separate encryption key and a message authentication code (MAC) key are derived. In EDDK, each node not only needs to store the pairwise keys and the local cluster keys but also needs to keep its own public and private keys. In addition, each node also needs to store a neighbor table to maintain the keys (including the pairwise key and the local cluster key) and sequence numbers. EDDK has three phase: key establishment, data transfer and key maintenance. The last phase includes key update, compromised key revocation, new node joining and mobile node joining.

In the key establishment phase, each node A computes its individual key K_a generates a sequence number SN_a and broadcasts a network joining message to determine its neighboring nodes. Finally, the pairwise key K_{ab} between neighboring nodes A and B is established with the individual keys (e.g. K_a and K_b) and the random sequence numbers (e.g. SN_a and SN_b):

$$K_{ab} = f(K_a \oplus K_b . SN_a \oplus SN_b) \tag{5}$$

In the key maintenance phase, the key update process occurs when one or more nodes are compromised or when the lifetime of a pairwise key is reached. The node whose ID is smaller would generate a pairwise key rekeying message and invoke the key update process. When the local cluster key of a node reaches its lifetime, the node broadcasts a similar key update message.

When a new node N wants to setup pairwise keys with its neighbors, it broadcast a new join message. Upon hearing the broadcasted new join message, each new node W and existing node B verifies the elliptic curve digital signature algorithm (ECDSA) [26] signature of node N. After verifying the legality of node N, for two new nodes, they can follow the key establishment phase to calculate the pairwise key between them. For the pairwise key negotiation between a new node and an existing node, the existing node B broadcasts a response message. Subsequently, with their own private key and the other's public key, both nodes N and B could independently compute the pairwise key $K_{nb} = s_n P_b = s_b P_n$. If a node A moves to a new location, it will broadcast a rejoin message. After verification, all of the three kinds of pairwise key (pairwise key between two mobile nodes, pairwise key between a mobile node and a new node and pairwise key between a mobile node and an existing node) can be computed using one's private key and the other's public key.

In EDDK, as long as the ID of a node and the sequence number field are in the neighbor table, reply attacks will fail at the receiver node. The Sybil attacks and node replication attacks will also fail because the attacker does not possess all information (the sequence numbers and pairwise key) required for message authentication. However, as one sensor node must establish the pairwise keys with all its neighbor and keep a neighbor table, EDDK may not be applicable in dense networks where each sensor node has many neighbors. Moreover, since the number of entries in a neighbor table is limited, there may be cases in which a trusted mobile node is unable to establish pairwise keys with its new neighbors and, as a result, cannot join the network. In order to defend against selective forwarding attacks, a promiscuous mode is used for each node to overhear data transmission among its neighboring nodes. However, higher energy consumption is associated with the promiscuous listening mode, which may influence the lifetime of the

network.

4. Comparisons

Table 1 compares the reviewed distributed dynamic key management schemes according to the security metrics. In Table 1, regarding resilience, "High" means that the compromised node cannot affect non-compromised nodes, "Medium" refers to the compromised node only affects his neighbors, and "Low" denotes the compromise of one node leads to the compromise of the whole network.

Table 1. Comparison of Distributed Dynamic Key Management Schemes According to Security Measures

Scheme	Forward and backward secrecy	Collusion resistance	Resilience	Node revocation
SHELL [15]	Both	Partial	Dependent on k and the probability that two nodes share a key	Revoke a compromised CH through cluster reorganization or revoke a non-CH node locally
LOCK [13]	Both	Partial	Dependent on k and the probability that two nodes share a key	Revoke a compromised CH with the BS or revoke a non-CH node locally
B-PCGR [18]	Backward	At most μ	μ-secure	Revoke the group key
C-PCGR [18]	Backward	At most μ	μ-secure	Revoke the group key
Cluster-based [19]	Both	At most t	t-secure	Remove the hierarchical key of the compromised node and revoke the group key
Compromise-resilient [23]	Both	Yes	High	N/A
EDDK [25]	Both	Yes	High	Remove the keys for revoke ID and revoke the local cluster key

Table 2. Comparison of Distributed Dynamic Key Management Schemes According to Efficiency and Flexibility Measures

Scheme	Memory	Processing	Mobility	Scalability	Key connectivity
SHELL [15]	k keys + key identifier	m Enc / Dec	No	Low	Probability that two nodes share a key, say p_1
LOCK [13]	k keys + key identifier	m Enc / Dec	No	Medium	p_1
B-PCGR [18]	g' Polynomial + shares of neighbor's e-polynomial	2n Enc / Dec + $o((n+1)s^2+\mu^3)$ Mul / Div	No	High	100%
C-PCGR [18]	g' Polynomial + shares of neighbor`s 0 level and 1-level e-polynomial	2n Enc / Dec + $o\big((2n+1)s^2+2\mu^3\big)$ Mul / Div	No	High	100%
Cluster-based [19]	2 + 1 cluster identifier + 1 personal secret	2t Enc / Dec + $O(t\ log^2 t)$ polynomial evaluation	No	High	100%
Compromise-resilient [23]	1+ 1 secret value + 1 t-degree perturbed polynomial	$O((n_a-n_c+1)t)$ Mul + (n_a-n_c+3) hash functions	No	N/A	100%
EDDK [25]	5 + neighbor table	1 Enc / Dec + 1 pseudorandom function	Yes	Low	100%

Table 2 summarizes the efficiency and flexibility of the reviewed schemes. Here, memory refers to the memory consumption per normal node. processing describes operations required per key generation and distribution. Speaking of scalability, "high" to that no further cost (storage, communication and computation) is induced when one node is added to the network, "Medium" implies the cost induced is reasonable, while "Low" means that high costs are generated with the addition of one node.

Security and efficiency are two contradicting objectives, which should be realized by the distributed

dynamic key management scheme. On the one hand, powerful security protocols usually require a larger amount of memory and energy. On the other hand, the limitation in hardware resources of the sensor platform does not allow equipping sensor nodes with powerful security contributes. Tables (1) and (2) summarize the results. There is no solution, which is always the best for distributed dynamic key management in wireless sensor networks. however, there are several cases, where distributed dynamic key management effectively uses the limited resources of sensor nodes. A clear solution to improve the efficiency and scalability of most schemes is to reduce the amount of information exchanged between nodes. The cluster-based key regeneration scheme is one of these schemes [27]. In this scheme, it is not required to transfer the key of member nodes when generating a new key cluster head key. PCGR and cluster-based key management schemes can be improved in this way. Node communications cost more than processing or storage in current sensor node platforms [6].

Another observation regarding security is using public key encryption, which supports end-to-end security. This review shows that only a small number of approaches are based on public key encryption. A promising key management scheme in wireless sensor networks combines the capabilities of both symmetric and public key encryption techniques. In this scheme, each node has a public key system, which is used to establish end-to-end symmetric keys with other nodes, similar to EDDK [25]. Another critical issue that arises from this approach is the development of more efficient public key algorithms and their implementations which can be widely used on current sensor nodes [28].

In most existing schemes, the revocation of a compromised node requires revoking a large number of keys [25], [29], [30], which causes serious problems in network connectivity.

Future research should particularly find techniques to discover compromised nodes and develop efficient methods to revoke nodes. In distributed dynamic key management, it is expected that keys are periodically updated. However, none of the existing protocols have proposed a decision or analysis about the interval of updating keys. this parameter can depend on node failure, selecting compromised nodes, data traffic volume, and extra processing load incurred by all nodes. Moreover, dynamic key management in specific wireless sensor networks, e.g. sparse WSNs [31] and mobile [32] WSNs are still open research areas. Furthermore, the authentication delay introduced in key-chain-based broadcast authentication mechanisms cannot satisfy real-time applications. There are many potential ways to disrupt the time-synchronization required in broadcast authentication techniques. Therefore, the development of time-synchronization independent broadcast authentication mechanisms is another promising domain for researchers.

5. Conclusion

This paper reviewed modern distributed dynamic key management schemes in wireless sensor networks. With the extensive and vast application of wireless sensor networks, key management has become a fundamental security issue considered by industrial researchers for which many schemes have been proposed so far. This paper discussed the basic requirements of distributed dynamic key management in wireless sensor networks, investigated proposed schemes for these environments, and highlighted the security and performance strengths and weaknesses. Finally, these schemes were compared based on the considered evaluation metrics. Accordingly, it is not possible to find a flawless scheme, which can perform well regarding all evaluation metrics and each pose certain strengths and weaknesses and are appropriate for certain applications. Therefore, an appropriate key management scheme should be selected based on the circumstances and environments. The main contribution of this research was to encourage more researchers to develop and improve potential proposals for distributed dynamic key management in wireless sensor networks.

References

[1] Akyildiz, L. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). A survey on sensor networks. *IEEE Communications Magazine, 40(8)*, 102-114.

[2] Kavitha, T., & Sridharan, D. (2010). Security vulnerabilities in wireless sensor networks: A survey. *Journal of Information Assurance and Security, 5*, 31-44.

[3] Menezes, A., P. van Oorschot, & Vanstone, S. (1999). *Handbook of Applied Cryptography*, 2nd ed. CRC Press.

[4] Eschenauer, L. & Gligor, V. D. (2002). A key-management scheme for distributed sensor networks. *Proceedings of the 9th ACM Conference on Computer and Communications Security*, New York, USA.

[5] Deng, J., Hartun, C., Han, R., & Mishra, S. (2005). A practical study of transitory master key establishment for wireless sensor networks. *Proceedings of the First Conference on Security and Privacy in Communication Network*s.

[6] Desnoyers, P., Ganesan, D., & Shenoy, P. (2005). TSAR: A two sensor storage architecture using interval skip graphs. *Proceedings of the Third International Conference on Embedded Networked Sensor Systems*.

[7] Divya, R., & Thirumurugan, T. (2011). A novel dynamic key management scheme based on hamming distance for wireless sensor networks. *International Journal of Scientific and Engineering Research, 2*, 1-12.

[8] Du, W., Deng, J., Han, Y., Varshney, P., Katz, J., & Khalili, A. (2005). A pairwise key predistribution scheme for wireless sensor networks. *ACM Transactions on Information and System Security, 8(2)*, 228-258.

[9] Michael, N., *et al.* (2013). Dynamic key management in wireless sensor networks: A survey. *Network and Computer Applications, 36*, 611-622.

[10] Simplicio, M. Jr, Barreto, P., Margi, C., & Carvalho, T. (2010). A survey on key management mechanisms for distributed wireless sensor networks. *Computer Networks, 54(15)*, 2591-2612.

[11] Mishra, S. (2002). Key management in large group multicast. Department of Computer Science, University of Colorado, Colorado.

[12] Ye, F., Luo, H., Cheng, J., Lu, S., & Zhang, L. (2002). A two-tier data dissemination model for large–scale wireless sensor networks. *Proceedings of the Eighth ACM/IEEE International Conference on Mobile Computing and Networking*, New York.

[13] Eltoweissy, M., Moharrum, M. & Mukkamala, R. (2006). Dynamic key management in sensor networks. *IEEE Communications Magazine*, 122-130.

[14] Eltoweissy, M., Heydari, M., Morales, L., & Sudborough, I. (2004). Combinatorial optimization of group key management. *Journal of Network and Systems Management*, *12(1)*, 33-50.

[15] Younis, M., Ghumman, K., & Eltoweissy, M. (2006). Location-aware combinatorial key management scheme for clustered sensor networks. *IEEE Transactions on Parallel and Distributed Systems, 17(8),* 865-882.

[16] Moharrum, M., Mukkamala, R., & Eltoweissy, M. (2004). TKGS: Verifiable threshold-based key generation scheme in open wireless ad hoc networks. *Proceeding of the 13th International Conference on Computer Communications and Networks*.

[17] Lo, C., Huang, C., & Chen, S. (2009). An efficient and scalable EBS-based batch rekeying scheme for secure group communications. *Proceedings of IEEE Military Communications Conference*, Boston, MA.

[18] Zhang, W., Zhu, S., & Cao, G. (2009). Predistribution and local collaboration-based group rekeying for wireless sensor networks. *Ad Hoc Networks*, *7(6)*, 1229-1242.

[19] Zhang, W., Shen, Y., & Lee, S. (2010). A cluster-based group key management scheme for wireless sensor networks. *Proceeding of the 12th Asia-Pacific Web Conferenc*e, Busan.

[20] Kim, J., & Cho, T. (2008). A-based key tree structure generation for group key management in wireless

sensor networks. *Computer Communications, 31(10)*, 2414-2419.
[21] Shamir, A. (1979). How to share a secret. *Communications of the ACM, 22(11)*, 612-613.
[22] Chadha, A, Liu, Y., & Das, S. (2005). Group key distribution via local collaboration in wireless sensor networks. Proceedings of IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Network.
[23] Guo, S., & Qian, Z. (2010). A compromise-resilient pair-wise rekeying protocol in hierarchical wireless sensor networks. *Smart Wireless Sensor Networks, 25(6)*, 1-18.
[24] Zhang, W., Tran, M., Zhu, S., & Cao, G. (2007). A random perturbation-based scheme for pairwise key establishment in sensor networks. *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing*, New York.
[25] Zhang, X., He, J., & Wei, Q. (2011). EDDK: Energy-efficient distributed deterministic key management for wireless sensor networks. *Wireless Communications and Networking*, *12*.
[26] Rivest, R., Hellman, M., Anderson, J., & Lyons, J. (1992). Responses to NIST's proposal. *Communications of the ACM*, *35(7)*, 41-54.
[27] Wang, G., Kim, S., Kang, D., Choi, D. & Cho, G. (2010). Lightweight key renewals for cluster sensor networks. *Journal of Networks,* 300-312.
[28] Hu, W., Tan, H., Corke, C., Shih, W., & Jha, S. (2010). Toward trusted wireless sensor networks. *ACM Transactions on Sensor Networks*, 1-25.
[29] Messai, M.-L., Aliouat, M., & Seba, H. (2010). Tree based protocol for key management in wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 1-59.
[30] Wang, Y., Ramamurthy, B., Zou, X., & Xue, Y. (2007). An efficient scheme for removing compromised sensor nodes from wireless sensor networks. Department of Computer Science and Engineering, University of Nebraska Lincoln.
[31] Anastasi, G., Conti, M., & Francesco, M. (2009). An analytical study of reliable and energy efficient data collection in sparse sensor networks with mobile relays. *Proceeding of the Sixth European conference on Wireless Sensor Networks*.
[32] Chuang, I.-H., Su, W.-T., Wu, C.-Y., Hsu, J.-P., & Kuo, Y.-H. (2007). Two-layered dynamic key management in mobile and long-lived cluster-based wireless sensor networks. *Proceedings of IEEE wireless Communications and Networking Conference*.

Seyed Reza Nabavi received his B.S. and M.S. degree in computer engineering from Iran. He is currently Ph.D. student of computer engineering. He was the leader of multiple research projects, the translator of multiple textbooks in Persian, multiple journal and conference papers. He is a member of young researchers and elite club of Islamic Azad University, Arak Branch. His research interests include sensor networks, cryptography, load balancing algorithm, real time, distributed, parallel and fault-tolerant systems.

Seyed Morteza Mousavi received his B.S., M.S. and Ph.D. degree in computer system architecture from Iran. He has been working as a lecturer and a faculty member with the department of computer engineering, Islamic Azad University, Arak Branch. He was leader of multiple research projects. He has also published many papers on international conferences and journals. His research interests include sensor networks, network security, computer architecture and cryptography.