SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2014; **7:**2003–2014
Published online 10 January 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.914

RESEARCH ARTICLE

Key Predistribution Scheme for Clustered Hierarchical Wireless Sensor Networks based on Combinatorial Designs

Masoumeh Javanbakht[1], Hossein Erfani[2], Hamid Haj Seyyed Javadi[1]* and Parisa Daneshjoo[3]

[1] Department of Mathematics and Computer Science, Shahed University, Tehran, P.O. Box: 18151-159, Iran
[2] Department of Computer, South Tehran Branch, Islamic Azad University, Tehran, Iran
[3] Department of Computer, West Tehran Branch, Islamic Azad University, Tehran, Iran

ABSTRACT

Combinatorial designs are powerful mathematical tools with comprehensive and simple algebraic structures. Recently, many researchers have used combinatorial designs as key predistribution schemes in wireless sensor networks (WSNs). Previous studies on security of WSNs are mainly concentrated on those networks containing nodes with the same capabilities. Further, investigations reveal that high reliability and lifetime on networks can be achieved through hierarchical heterogeneous wireless sensor networks, where a small number of sensor nodes have more energy, memory, and transmission capability. Inspired by scheme due to Lee and Stinson, we propose a key predistribution scheme for a clustered heterogeneous WSN using transversal designs. This proposed scheme assigns key chains to sensor nodes before deployment and separates key pool of each cluster by adding a pseudo-random generated number after deployment. The performance evaluation and security analysis show that our proposed scheme can provide better security with significant reductions on communication overhead and storage space than other key management schemes without compromising connectivity. Copyright © 2014 John Wiley & Sons, Ltd.

KEYWORDS
combinatorial design; security; heterogeneous wireless sensor network; key predistribution; transversal design; resilience

***Correspondence**
Hamid Haj Seyyed Javadi, Department of Mathematics and Computer Science, Shahed University, Tehran, P.O. Box: 18151-159, Iran.
E-mail: h.s.javadi@shahed.ac.ir

1. INTRODUCTION

A wireless sensor network (WSN) consists of many sensors that have very limited storage capacity, power, and computational capabilities [1]. WSNs have many applications such as environmental and habitat monitoring, infrastructure security and counterterrorism applications, vehicle traffic monitoring, and industrial process control [2]. Nodes in WSNs should be able to communicate with each other to store and relay information to the base station. Wireless nature of communications between nodes in WSNs and unsecure environment allow an attacker to eavesdrop communication messages. So, the need to secure transmitted messages is clear. To secure communications in a WSN, all messages should be encrypted with keys distributed between nodes.

Cryptographic methods can be divided into symmetric key and asymmetric key (public key) cryptography. Researchers have demonstrated that public key cryptography imposes computational cost and much processing time [1,3]. So, computational constraints and consumption power of sensor nodes limit the use of public key cryptography in WSNs; accordingly, symmetric key cryptography is recommended.

Because the network security depends on the used mechanism to distribute required keys between sensor nodes, an efficient key establishment method should be designed in order to distribute the cryptographic keys among the sensor nodes. One possible approach is to establish secret keys using public key protocols such as key agreement schemes, but as was mentioned earlier, public key cryptography imposes extensive computational requirements. Key predistribution is another solution to the

problem of key establishment in sensor networks where each sensor node is preloaded with a finite set of keys prior to deployment. A key predistribution scheme (KPS) is a means of specifying which nodes store which keys.

Key predistribution schemes can be random, deterministic, and hybrid. In random schemes, keys are randomly drawn from a key pool and are stored in each sensor node. This approach does not make sure whether every two nodes can communicate directly. If direct communication is not possible then a path needs to be established between two nodes. This path establishment decreasing the speed of communications increases power consumption. In deterministic schemes, deterministic methods are used to design key pool and key chains aiming at providing better key connectivity. In hybrid schemes, both deterministic and random approaches are combined to improve scalability and resiliency.

Eschenauer and Gligor proposed a random KPS for distributed WSNs in [4]. In this scheme, at first, a large key pool is generated, and each sensor node is loaded with the fixed number of keys chosen randomly from this key pool along with their key identifiers. Then in key discovery phase, to find a common key, two nodes in their wireless communication range exchange the list of key identifiers from their own key chains. In case of sharing a common key, they can establish a direct secure communication. Otherwise, two nodes try to communicate with each other through a multi hop path. The proposed scheme in [4] provides good connectivity and resiliency but in shared key discovery phase, because two nodes have to exchange the list of their key identifiers to find the common key, the number of broadcast messages increases. This, in turn, enhances the communication complexity of protocol and decreases the battery's life.

Combinatorial designs are one of the methods used to design a deterministic KPS. (Combinatorial design theory is interested in arranging elements of a finite set into subsets to satisfy certain properties [5]). Deterministic KPSs based on combinatorial designs have some advantages such as the following:

- Applying combinatorial designs in key predistribution makes the study of metrics for evaluating KPSs easily performed such as resilience and local connectivity. In addition, appropriate choice of parameters in a combinatorial design can increase the maximum connectivity and decrease the length of key path.
- The rich mathematical structure of combinatorial designs result in shared key discovery and path-key establishment phases is carried out in a constructed method so that the complexity of computational and communicational algorithms can be reduced to O(1).

The previously mentioned unique properties have attracted the attention of many researchers, and so, many KPSs based on combinatorial designs have been proposed for homogeneous WSNs in recent years. Some of them are mentioned in the succeeding text:

In 2004, Camtepe and Yener presented a deterministic KPS for a distributed WSN [5]. In their proposed scheme, the specific combinatorial designs namely finite generalized quadrangles and symmetric balanced incomplete block design (BIBD) were employed. Their scheme provided key sharing and resulted high network connectivity and simultaneously encountered the constraints of resiliency and scalability.

In 2005, Lee and Stinson examined set systems as a deterministic KPS. They used the specific class of set systems named transversal design (TD) [6], which provided better resiliency. Other researchers such as Roy and Ruj employed partially BIBD (PBIBD) in 2007 [7], or Dong and Pei used Orthogonal Arrays in 2008 [8], as deterministic KPSs in WSNs.

In all of the previous proposed KPSs, researchers mainly considered homogeneous sensor networks organized in a flat architecture. In such networks all sensor nodes have the same characteristics such as battery's life, computational power, and memory capacity. However, flat ad hoc networks are suitable in terms of efficiency and simplicity for sensor applications, but recent researches have indicated the weakness of these networks in terms of performance and scalability [9,10]. So, recently, heterogeneous networks have been getting more attention.

There are different types of nodes having different levels of capabilities and transmission ranges in heterogeneous sensor networks [11–13]. Hierarchical WSN is a kind of heterogeneous WSNs where there is a hierarchy among the nodes based on their capabilities: cluster heads (CHs) and cluster nodes (CNs). CHs are a large number of powerful nodes, which have different capabilities in terms of communication, computation, energy supply, storage space, and reliability. The rest of the nodes, which have the same capabilities, are called CNs.

In this paper, we develop the proposed scheme in [4] for the mentioned type of heterogeneous WSNs. We utilize a TD to preload keys on predeploymented nodes in such networks. The structure of a TD allows each CN sharing a common key with its own CH, which makes direct communication of each CH with its CNs possible without imposing a high communication overhead [13] or large required storage space [14]. Additionally, in [15], it is demonstrated that a TD is near optimal in terms of key pool size, which tends to maximize the resilience. After sensor nodes are deployed and clustering is carried out, key chains are reconstructed by concatenating a pseudo-random number uniquely generated by each CH. This concatenation provides distinct key pool in each cluster, which greatly decreases the effect of capture node.

The paper is followed as in the succeeding text. In Section 2, we present a brief review of required combinatorial designs. In Section 3, the description of our framework for KPS in a clustered hierarchical WSN is presented. The simulation analysis and discussions are presented in Section 4. The paper focuses on a comparison of proposed

DOI: 10.1002/sec

scheme with the previous works in Section 5, and our conclusion is accessible in Section 6.

2. PRELIMINARIES ON COMBINATORIAL DESIGNS

2.1. Review definitions

In the present section, we briefly review some definitions and notations used through the paper mainly based on [16].

A set system is a pair $(X, \mathcal{A})$, where X is a set of v elements (points) and $\mathcal{A}$ is a finite set of subsets of X called blocks. The degree of a point $x \in X$ is the number of blocks containing x, and $(X, \mathcal{A})$ is regular of degree r, if all points have the same degree, r. The rank of a set system is the size of the largest block, and $(X, \mathcal{A})$ is said to be uniform of rank k if all blocks have the same size k.
A set system is called a group divisible design $GDD(n, p^g)$, if $v = gp$ and in addition to set of blocks of size n, there exists a partition $\mathcal{H}$ of X into g groups of size p such that

- For every group $H \in \mathcal{H}$ and block $A \in \mathcal{A}$, $\mid H \cap A \mid \leq 1$.
- Every pair of elements of X from different groups occurs exactly in one block in $\mathcal{A}$.
- Two elements of one group do not occur together in one block.

A transversal design, $TD(n, p)$, is a group-divisible design $GDD(n, p^n)$ where every group $H \in \mathcal{H}$ intersects each block $A \in \mathcal{A}$ in precisely one element. In a transversal design $TD(n, p)$, any two distinct blocks intersect at most in one element and every element belongs exactly to p blocks.

A method to construct a transversal design $TD(n, p)$ is summarized in the following theorem, which is extracted from [17]. For the sake of completeness, we also include the complete proof.

Theorem 2.1. *Suppose that p is prime and* $2 \leq n \leq p$ *then there exists a* $TD(n, p)$.

Proof. Let $X_1 \subseteq \mathbb{F}_p, \mid X_1 \mid = n$, then we define $X = X_1 \times \mathbb{F}_p$. For $x \in X_1$, define $H_x = \{x\} \times \mathbb{F}_p$, and for every ordered pair $(i, j) \in \mathbb{F}_p \times \mathbb{F}_p$, let $A_{i,j} = \{(x, ix + jmodp) : x \in X_1\}$. Then, $\mathcal{A}$,$\mathcal{H}$ are defined as follows:

$$\mathcal{A} = \{A_{i,j} : (i, j) \in \mathbb{F}_p \times \mathbb{F}_p\},$$
$$\mathcal{H} = \{H_x : x \in X_1\}.$$

We prove that $(X, \mathcal{H}, \mathcal{A})$ is a $TD(n, p)$. Consider two points $(x_1, y_1), (x_2, y_2)$ from different groups. Find $i, j \in \mathbb{F}_p$ such that $y_1 = ax_1 + b$ and $y_2 = ax_2 + b$. It's easy to see that these two equations have a unique solution $i, j \in \mathbb{F}_p$. □

To understand how the blocks and groups are constructed in a $TD(n, p)$ based on previous theorem, consider the following example.

Example 2.1. *Suppose that* $n = 4$, *and* $p = 5$, *with* $X_1 = \{0, 1, 2, 3\}$. *Then, object set X, groups, and blocks are as follows:*

$$X = \{0, 1, 2, 3\} \times \{0, 1, 2, 3, 4\}$$
$$H_0 = \{(0, 0)(0, 1)(0, 2)(0, 3)(0, 4)\}$$
$$H_1 = \{(1, 0)(1, 1)(1, 2)(1, 3)(1, 4)\}$$
$$H_2 = \{(2, 0)(2, 1)(2, 2)(2, 3)(2, 4)\}$$
$$H_3 = \{(3, 0)(3, 1)(3, 2)(3, 3)(3, 4)\}$$
$$A_{00} = (0, 0)(1, 0)(2, 0)(3, 0) A_{01} = (0, 1)(1, 1)(2, 1)(3, 1)$$
$$A_{02} = (0, 2)(1, 2)(2, 2)(3, 2) A_{03} = (0, 3)(1, 3)(2, 3)(3, 3)$$
$$A_{04} = (0, 4)(1, 4)(2, 4)(3, 4) A_{10} = (0, 0)(1, 1)(2, 2)(3, 3)$$
$$A_{11} = (0, 1)(1, 2)(2, 3)(3, 4) A_{12} = (0, 2)(1, 3)(2, 4)(3, 0)$$
$$A_{13} = (0, 3)(1, 4)(2, 0)(3, 1) A_{14} = (0, 4)(1, 0)(2, 1)(3, 2)$$
$$A_{20} = (0, 0)(1, 2)(2, 4)(3, 1) A_{21} = (0, 1)(1, 3)(2, 0)(3, 2)$$
$$A_{22} = (0, 2)(1, 4)(2, 1)(3, 3) A_{23} = (0, 3)(1, 0)(2, 2)(3, 4)$$
$$A_{24} = (0, 4)(1, 1)(2, 3)(3, 0) A_{30} = (0, 0)(1, 3)(2, 1)(3, 4)$$
$$A_{31} = (0, 1)(1, 4)(2, 2)(3, 0) A_{32} = (0, 2)(1, 0)(2, 3)(3, 1)$$
$$A_{33} = (0, 3)(1, 1)(2, 4)(3, 2) A_{34} = (0, 4)(1, 2)(2, 0)(3, 3)$$
$$A_{40} = (0, 0)(1, 4)(2, 3)(3, 2) A_{41} = (0, 1)(1, 0)(2, 4)(3, 3)$$
$$A_{42} = (0, 2)(1, 1)(2, 0)(3, 4) A_{43} = (0, 3)(1, 2)(2, 1)(3, 0)$$
$$A_{44} = (0, 4)(1, 3)(2, 2)(3, 1).$$

In the following two sections, first we describe the notations and network model for the clustered hierarchical WSN. Then, we explain how designs are used to generate key chains for the sensors in a hierarchical WSN.

2.2. Network model

As we discussed a bit earlier, two types of nodes can be considered in clustered hierarchical WSNs. In one type of these nodes energy, computing power and memory are high and in others low. Because in this framework, key chains are preloaded before node's deployment, it is necessary that CHs be specified before nodes are deployed in the environment. So, nodes with high energy are considered as CHs, and other nodes play the role of CNs.

CHs can communicate with BS and with each other directly, as well. Also, in each cluster, CNs can communicate with their associated CHs straightforward. BS is a trusted station, which is never compromised. We assume that nodes have unique identifiers (Section3.1), are randomly deployed in the environment, and do not have any mobility. The notations used in present paper are summarized in Table I.

2.3. Mapping from transversal design to key distribution

For a hierarchical WSN with c clusters and N cluster nodes, a transversal design $TD(n, p)$ can be used as a KPS as illustrated in Table II.

Table I. List of used notations.

Notation	Definition
BS	Base Station
N	Total number of cluster nodes
c	Number of clusters
n_j	Number of nodes in each cluster
R	Radius covered by a sensor node
α_j	Generated challenge by cluster head CH_j
K_i i=1,2,...,n	Key chain of a cluster node before deployment
K_j j=1,2,...,p	Key chain of a cluster head before deployment
CH_j	Cluster head of j-th cluster for j$\in \{1,2,,c\}$
CN_i^j	i-th cluster node of j-th cluster
CN_i	Cluster node (sensor nodes with low energy)
$K_i^{'}$	Key chain of a cluster node after deployment
$K_j^{'}$	Key chain of a cluster head after deployment

Let the CNs and CHs be denoted $CN_1,,CN_N$ and $CH_1,,CH_c$, respectively. Consider the sets $X = \{x_u : 1 \leq u \leq np\}$ and $\mathcal{A} = \{A_i : 1 \leq i \leq p^2\}$ and $\mathcal{H} = \{H_j : 1 \leq j \leq n\}$ of $TD(n,p)$. Following [17], the np points in X acts as key identifiers, which are associated with a set of np keys such that for each key identifier x_u, the key L_t (where $t = x_u$) is selected randomly from some specified key pool, say L (e.g., $L = \{0,1\}^{128}$). Then, for $1 \leq i \leq N$ each cluster node CN_i and for $1 \leq j \leq c$ each cluster head CH_j is associated with a set of keys as follows:

$$K_i = \{L_t : t = x_u \in A_i\} \rightsquigarrow CN_i,$$
$$K_j = \{L_t : t = x_u \in H_j\} \rightsquigarrow CH_j.$$

In this way, the block A_i and group H_j of $TD(n,p)$ including key identifiers are used to associate key chain to the cluster node CN_i and cluster head CH_j, respectively.

Note that the number of cluster nodes, N should be equal or less than the number of blocks, p^2, also clusters' number would be equal or less than the number of groups, n in $TD(n,p)$. If $N < p^2$, in feature, we can add new nodes to the network if desired. In this case, additional blocks from the same key-chain space can be used to assign keys to new nodes, which are added to network. So, we can provide scalability in the proposed scheme.

From a transversal design $TD(n,p)$, we obtain a KPS with the following properties for the hierarchical WSN:

- There is no common key between CHs.
- Each group intersects each block in exactly one element, it means that each CH shares a common key with each CN.
- In a TD, each pair of blocks intersect at most in one element, it means that each pair of CNs share zero or one common key.

3. PROPOSED FRAMEWORK

In the following, we explain how to distribute the required keys and key chains on sensor nodes prior deployment. This process is accomplished in key setup phase; then in shared key discovery phase, we express how two sensor nodes can discover a common key for their secure communications.

3.1. Predeployment phase

Before sensor nodes are randomly deployed in an environment, they are preloaded with key chains generated by a trusted server. As illustrated in Section 2, key chains are generated on the basis of transversal design $TD(n,p)$ (Theorem 2.1). Each block $A_{(m_i,n_i)}$ in $TD(n,p)$ is used to associate key chain K_i to cluster node CN_i for $1 \leq i \leq N$, and each group H_j is used to associate key chain K_j to cluster head CH_j, for $1 \leq j \leq c$. The index of employed blocks and groups in this phase is stored as ID of CN_i and CH_j, respectively. Consequently, prior to deployment, each CN_i is preloaded with a key chain K_i, and every CH_j is preloaded with a key chain K_j.

Because in this phase each group H_j is used to associate key chain K_j to cluster head CH_j, and these groups do not share any element (Section2.3), we use a BIBD design [14] to establish secure communications between CHs. In [14], it is mentioned that BIBD is a fully connected KPS, meaning that each two nodes can communicate directly. So, direct communication between each two CHs is possible if a BIBD is employed to assign cryptographic keys to them. Although, high storage in BIBD limits its employment in large WSNs, its usage in small WSNs does not encounter us with such problems. So it is the number of CHs in a WSN is small, the choice of BIBD for establishment of CHs communication keys can be perfect.

Note that according to what was mentioned, if two CHs which have definitely a shared key, can not communicate directly for some reasons (i.e. not being in each others communication range), they will be able to communicate easily by use of their common key through a two hop path established by BS.

3.2. Key setup phase

After preloading sensor nodes with required keys, they are randomly deployed in an environment, and clustering is performed. Clusters can be formed on the basis of various criteria such as location, communication range, and resource and energy capabilities [18]. The method of clustering is not discussed here, but it is expected that CHs be deployed within the field such that each sensor node belongs to one cluster. In order to show which cluster node belongs to which cluster, hereafter we denote CNs with CN_i^j instead of CN_i, where j indicates the index of cluster in which cluster node is.

After clusters formation, each cluster head CH_j generates a pseudo-random number, which we call it *challenge* α_j, and broadcasts the list $\{E_{L_t}(\alpha_j \parallel \textit{hello message}) : L_t \in K_j\}$ to all its cluster nodes, where *hello message* is shared key's identifier between CH_j and CN_i^j, which is discovered by CH_j based on determined

DOI: 10.1002/sec

Table II. Mapping from transversal design to key distribution.

Key distribution	TD(n,p)
Key pool	Object set (points) =X
Number of keys in a cluster node's key chain	Number of objects in a block = n
Total number of cluster nodes in all clusters	Number of blocks = p^2
Number of keys in the key chain of a cluster head	Number of objects in a group = p
Maximum number of clusters	Number of groups = n
Number of key chains that a specific	Number of blocks that an object is
Key is contained in	Contained in = p

Table III. Reconstructing key chains.

Prior deployment	After deployment
$K_i = \{L_t : t = x_u \in A_i\}$	$K_i^{'} = \{L_t^{'} : L_t^{'} = h(L_t \parallel \alpha_j)$ $s.t. L_t \in K_i\}$
$K_j = \{L_t : t = x_u \in H_j\}$	$K_j^{'} = \{L_t^{'} : L_t^{'} = h(L_t \parallel \alpha_j)$ $s.t. L_t \in K_j\}$

method in Section3.3. Each cluster node upon receiving the list, after discovering shared key, decrypts each $E_{L_t}(\alpha_j \parallel$ *hello message*). After decryption, if *hello message* is equal to identifier of shared key, CN_i^j confirms the validity of *challenge* α_j; if not, CN_i^j discards this message. In this way, no malicious node can forge the *challenge* α_j. Each cluster node CN_i^j and each cluster head CH_j use α_j to reconstruct their key chains by concatenating *challenge* α_j to the keys of their key chains as summarized in Table III.

Joining *challenge* α_j to the sensor nodes key chain makes CNs in different clusters not to share any common key. In other words, key space that was common at first for all nodes is separated for different clusters because *challenge* α_j is unique for each cluster. In this way, intersection of clusters key space will be null. So capturing nodes in a cluster having no effect on the other clusters, which makes the resilience of scheme to node, compromise attack increased.

It is worth mentioning again that *challenge* α_j does not have any role in cryptographic communications between CNs and CHs directly; it is just used to partition the overall key space into unique spaces for each cluster. So exposing *challenge* α_j alone does not risk any secure communication in the network. A communication will be in risk when both *challenge* α_j and key chain K_i are exposed.

3.3. Shared-key discovery

After completing key setup phase, each node needs to discover whether it shares any key with the other nodes or not. This shared-key discovery should be performed in each cluster between cluster nodes, between CNs and their relevant CHs, and among CHs.

Because $TD(n, p)$ is used to assign key chains to sensor nodes, shared-key discovery is easily performed. Discovery of shared keys between CNs is accomplished using proposed method in [17]. In this manner, unlike shared-key discovery phase in random KPSs, there is no need to exchange key identifiers between sensor nodes that increases communication complexity and decreases battery's life, just very little information needs to be broadcasted. In the following, we explain how this method will be generalized in our proposed scheme. It should be noted that each cluster node CN_i^j has a unique ID, (m_i, n_i), which is the index of corresponding block from predeployment phase. Similarly, j, the index of corresponding group from predeployment phase, is the identifier of cluster head CH_j.

- Consider two CNs CN_i^j, $CN_{i'}^j$ of one cluster. Following [17], these two nodes can independently determine if they share a common key as follows, CN_i^j compares its own ID with $CN_{i'}^j$s ID:

  (1) If $m_i = m_{i'}$ (so $n_i \neq n_{i'}$) then CN_i^j, $CN_{i'}^j$ do not share any common key.
  (2) Otherwise, CN_i^j computes the value of

  $$z = (n_{i'} - n_i)(m_i - m_{i'})^{-1} \bmod p$$

  if $0 \leq z \leq n-1$, then $L'_{(z,(m_i * z + n_i) \bmod p)}$ is the common key between CN_i^j and $CN_{i'}^j$. If $z \geq n$, two nodes do not share any common key.

  So, if CN_i^j and $CN_{i'}^j$ have a common key, it can be easily detected with this method. If two nodes do not share any common key, it will be shown in Section 4 with probability $Pr2$ that there is a third node CN_h^j, in the intersection of the neighborhood of the two nodes such that it shares a common key with each of CN_i^j and $CN_{i'}^j$ individually. Thus, two nodes can communicate through a two hop path.
- Now consider cluster head CH_j and cluster node CN_i^j, that can independently determine their common key easily by computing $(j, (m_i * j + n_i) \bmod p)$, which yields the shared key $L'_{(j,(m_i * j + n_i) \bmod p)}$.
- As mentioned in Section 3.1, a BIBD [14] is used to establish secure communications between CHs. In a BIBD, any two blocks intersect precisely in one element, which results in any two CHs, can communicate easily via the shared keys in their key chains.

4. DISCUSSIONS

In this section, we analyze the performance and security of our scheme. We present our analytical results on the following two metrics:

- Network connectivity: It is defined as the probability that two nodes can communicate with each other securely and efficiently. If two nodes are neighbors, it is called local connectivity; otherwise, it shows global connectivity. Because the global connectivity is strongly dependent on physical topology of nodes, it can be difficult to assess on the basis of the properties of KPS [19]. So, only the local connectivity is discussed here.
- Resiliency: When sensor nodes are distributed in a hostile environment, some of them may be compromised by an adversary. In this case, we assume that all of keys or information stored on captured nodes can be detected by the adversary. So, revealed keys cannot be used more for secure communications between nodes. Also links containing revealed keys will be in risk. The resiliency of a network is defined as the probability of a link between two uncompromised nodes is broken, when s nodes are captured, which is denoted by fail(s).

4.1. Network connectivity

Two neighboring nodes can communicate directly when they share a common key. Let $Pr1$ shows the probability of common key existence between two neighboring nodes. In [17], $Pr1$ is computed as follows:

$$Pr1 = \frac{n}{p+1} \tag{1}$$

However, in our proposed scheme, as mentioned in Section 2.3, at first, a transversal design TD(n,p) is used as a KPS, but the key sharing probability can not be computed as simply as proposed in [17]. Because after deployment and clustering nodes, those properties of TDs, which help us to compute (1) such as belonging each object to p blocks, may not be satisfied in each cluster. So, to compute local connectivity in proposed key management, at first local connectivity for each cluster ($Pr1_j$) is computed as summarized in (2), then connectivity of entire network is obtained with computing weighted average on all clusters in (3).
Let S_i^j denote that all of the nodes are in the communication range of CN_i^j, and A_i indicates all of the nodes in the cluster j that share a common key with CN_i^j:

$$A_i = \left\{ CN_{i'}^j : A_{m_{i'},n_{i'}} \cap A_{m_i,n_i} \neq \emptyset \right\},$$

Then, $Pr1_j$ is estimated as in the succeeding text:

$$Pr1_j = \frac{\sum_{i=1}^{n_j} \mid A_i \cap S_i^j \mid}{\sum_{i=1}^{n_j} \mid S_i^j \mid} \tag{2}$$

Where n_j is the number of CNs in the cluster j. Eventually, $Pr1$ will be equal to:

$$Pr1 = \frac{\sum_{j=1}^{c} n_j \times Pr1_j}{N}. \tag{3}$$

Two neighboring nodes sharing no common key may be able to communicate through a common neighbor, which has a common key with each of two nodes. The probability that such two nodes can communicate through a two hop path (denoted by $Pr2$) is estimated as follows:

For $1 \leq i \leq N$, let η_i denote all of the nodes in the communication range of CN_i^j that share no common key with CN_i^j, and β_i indicates the set of all nodes in the communication range of CN_i^j that there exists an intermediate node in the intersection of their neighborhoods such that every pair of adjacent nodes have a common key,

$$\eta_i = \left\{ CN_l^j \in S_i^j : A_{m_l,n_l} \cap A_{m_i,n_i} = \emptyset \right\},$$
$$\beta_i = \left\{ CN_l^j \in S_i^j : \exists CN_h^j \in S_l^j \cap S_i^j \right\}$$

where, $A_{m_h,n_h} \cap A_{m_i,n_i} \neq \emptyset$ and $A_{m_h,n_h} \cap A_{m_l,n_l} \neq \emptyset$. So, the probability $Pr2$ is estimated as

$$Pr2 = \frac{\sum_{i=1}^{n_j} \mid \eta_i \cap \beta_i \mid}{\sum_{i=1}^{n_j} \mid S_i^j \mid} \tag{4}$$

Note that (2), (3), and (4) describe our approach to compute $Pr1$ and $Pr2$ in our implementations. In fact, a straightforward observation shows that given equations for $Pr1$ and $Pr2$ in [6] satisfy in our scheme too.

Experimental values of $Pr1$, $Pr2$, and $Pr1 + Pr2$ for 20 runs are illustrated in Figures 1, 2, and 3. Implementations have been performed on the basis of (3) and (4) for a network with $N = 1369$ nodes and $c = 10$ clusters, where nodes are distributed randomly in square area of length 100. Let sensor node SN_i be located at (x[i],y[i]), where $0 \leq x[i], y[i] \leq 100$ and the radio radius of each

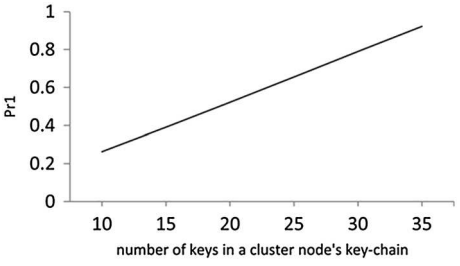

Figure 1. Probability of existence a one hop path between two cluster nodes versus the number of keys in the key chain of cluster nodes.

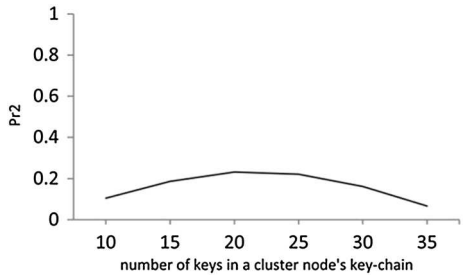

Figure 2. Probability of existence a two hop path between two cluster nodes with no common key versus the number of keys in the key chain of cluster nodes.

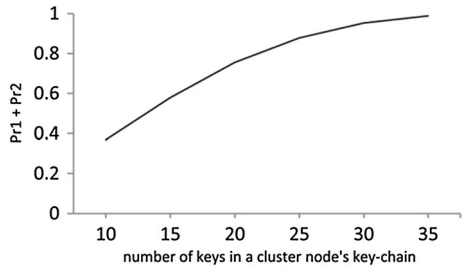

Figure 3. Probability of existence a path of length one or two between two cluster nodes versus the number of keys in the key chain of cluster nodes.

node is fixed to be 10. $TD(n, 37)$ is employed as a KPS in predeployment phase.

Figure 1 shows the probability of key sharing as the key chain size of CN_i^j (the size of block A_i in $TD(n, p)$) is varied. We observe that $Pr1$ increases linearly with the increase of the size of $K_i^{'}$. It means the more keys preloaded on CNs the better connectivity, so that, maximum happens when p keys are stored on each CN. For instance, from Figure 1, we see that for networks with nearly 1370 nodes, storing 37 keys on each CN provides connectivity nearly 97%.

Although the probability $Pr1$ is very approximate to one when p keys are preloaded on each node, but it does not take the value of one. It means that even with preloading of the maximum keys number, which a TD scheme allows, there are some nodes that are not able to communicate via a one-hop path. So it is essential that the secure connections within two-hops to be established.

In Figure 2, we study two hop path existence probability, and Figure 3 indicates the probability that two nearby nodes are connected via one or two-hop path versus the size of $K_i^{'}$. The results in Figure 3 demonstrates that the key chain length can be decreased, provided that two hop links additional one hop links be involved in providing the network connectivity. In other words, In other words, if both one and two hop paths are involved in communication between nodes, being network connected can be provided with less key storage in comparison it would be provided only through the expansion of one hop paths. For instance,

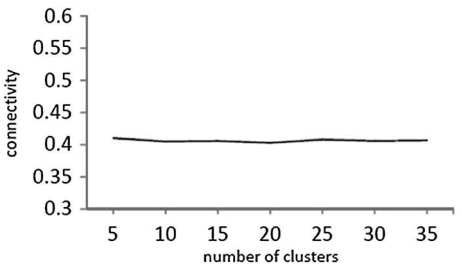

Figure 4. Probability of local connectivity versus the number of clusters (when the length of cluster node's key chain is fixed to 15).

with storing only 25 keys on each cluster node, $Pr1 + Pr2$ is estimated as 0.88, while to obtain such value for $Pr1$, we need to store almost 35 keys on each node. Hence, we reduce the key storage with contributing two hop paths.

It should be considered that although full connectivity is a nice feature for KPSs, it seems unnecessary in a hierarchical WSN. Because sending and receiving information in such networks is mainly responsible for CHs, overall direct communication among all of CNs is unnecessary. Hence, imposing high storage space to establish direct communication between each two CNs is not reasonable in such WSNs.

Acquired values in Figure 4 illustrate the effect of increasing the number of clusters versus the network connectivity. Obtained values indicate that the increase or decrease in the number of clusters has no significant effect in reducing or increasing the probability of key sharing. However, in the next section, we will show an increase in the number of clusters, which can significantly improved the network resiliency.

4.2. Resiliency

Our ideal condition is one in which the node compromise few effects on network communications. In this proposal, $fail(s)$ is given by (5) for each cluster, then resiliency of entire network is computed by taking the weighted average on all clusters,

$$\text{fail(s)} = \frac{\text{number of broken links}}{\text{total number of links between uncompromised nodes}} \quad (5)$$

Experimental values of $fail(s)$ for 20 runs are shown in Table IV. We have considered a network with $N = 1369$ nodes and $c = 10$ clusters such that nodes are distributed randomly in a square area of length 100. Let sensor node SN_i is located at (x[i],y[i]), where $0 \leq x[i], y[i] \leq 100$ and the radio radius of each node is fixed to be 10.

The values in Table IV show that the proposed scheme provides very high resiliency. For instance, $fail(648) \simeq 0.837$, so any given link remains secure with a probability of about 0.17 when 50% of sensor nodes are compromised. This resiliency improvement is due to concatenation $challenge\,\alpha_j$. As explained in Section 3.2, we

Table IV. Value of fail(s) when s nodes are compromised.

s	Fraction of compromised nodes (%)	Fail(s)
14	1	0.02
69	5	0.1709
137	10	0.3159
273	20	0.5195
648	50	0.8370

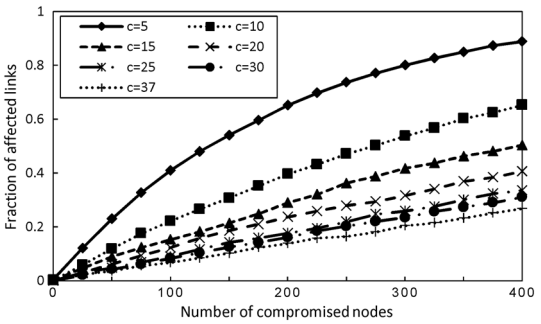

Figure 5. Fraction of affected links when the number of clusters increases versus the number of compromised nodes.

added the *challenge* α_j to each cluster node keys in the key setup phase. It makes the key chains of CNs in one cluster not share any common key with key chains in other clusters. So, captured nodes in one cluster do not affect links in other clusters. Hence, although an adversary in a cluster compromises all cluster nodes, only the links in that cluster are affected and those of other clusters remain completely secure.

Furthermore, we show how the increase in the number of clusters affects the network resiliency. The curves in Figure 5 show the values for *fail*(*s*) when the number of clusters increases from $c = 5$ to $c = 37$.

Obtained values demonstrate that the network resiliency can be substantially improved when the number of clusters increases. For example, the compromise 20% of sensors in the network with five clusters affect approximately 75% of links, whereas capturing the same number of nodes in the network with 20 clusters affects just 29% of links. Although the trend of illustrated curves decreases with enhancement of the number of clusters and Figure 5 follows on linear trend, we cannot increase clusters number arbitrarily. As we explained before in Section 2, the maximum number of clusters can be equal to the number of generated groups in the used transversal design $TD(n, p)$ in predeployment phase. Furthermore, performed simulations verify that increasing the number of clusters more than accredited threshold ($c > n$ for $TD(n, p)$) does not have phenomenal effect on resiliency improvement. We can observe that the resiliency changes slightly if the network has more than n clusters. So, network resiliency will have maximum own value if the number of clusters is equal to the number of generated groups in the TD used as a KPS.

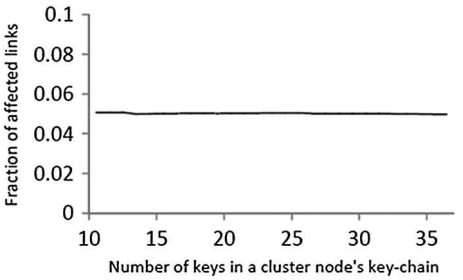

Figure 6. Fraction of affected links when 20 nodes are compromised versus the size of cluster node's key chain.

As mentioned earlier in Section 4.1, increasing the key chain size increases the network connectivity, while the obtained simulations (Figure 6) illustrate the increasing key chain size does not affect the network resiliency.

These observations verify that we can increase network connectivity by increasing the size of key chain without impressing the resiliency; however, this enhancement should be performed regarding to nodes storage limitations and used TD in predeployment phase (for more details see Section 2 and Theorem 2.1). Also, we can improve network resiliency by increasing the number of clusters without any misused effect on network connectivity. So, in our proposed scheme, we could improve two important metrics: network connectivity and resiliency.

5. COMPARISON

As discussed earlier, our scheme proposes to employ a combinatorial design named TD to establish cryptographic keys in a hierarchical WSN in a predistribution approach. Predistribution in a hierarchical WSN has potential advantages in comparison with the public key protocols for such networks, some of which have been mentioned in [3,4]. In order to better illustrate these advantages, in the first part of this section, we compare our proposed scheme with [13] and [20], two key generation and distribution schemes based on public key cryptography (particularly elliptic curve cryptography [ECC]), in terms of computational and communicational overhead, and memory storage. And the second part demonstrates how resiliency can be improved when a combinatorial design is used as a KPS for hierarchical networks comparing when it is used as a KPS for homogenous networks. We compare our scheme with represented schemes in [14] and [17] as well.

5.1. Comparison with represented schemes in [13] and [20]

In [13], authors employ public key cryptography to establish secure communications in clustered hierarchical WSNs, and in [20], authors present a unified security framework embodying three key management schemes: SACK, SACK-P, and SACK-H. Among these three

schemes, only SACK-P uses the public key cryptography; so, to investigate positive aspects of our proposed scheme into public key protocols, only the aforementioned scheme is involved in our comparisons.

5.1.1. Memory analysis.

Assume that the number of CHs and CNs in a hierarchical WSN is c and N, respectively, and according to the mapping from Section 2.3, a $TD(n, p)$ is employed to assign required keys in such a network, where $N = p^2$ and $c \leq n$. In a KPS based on a $TD(n, p)$, each CH is preloaded with a key chain of length p for communicating with its CNs, plus a key chain from BIBD design to establish secure communications with other CHs that approximately equals to $\sqrt{c}$. Consequently, the memory storage requirement for each CH is obtained as $(\sqrt{N} + \sqrt{c}) \times B^k$, where B^k is the size of key in symmetric cryptography. On the other hand, each CN is preloaded with a key chain of length n, so the memory storage requirement for it is $n \times B^k$. Now, if m denotes the number of CNs in each cluster and B^u the key size in public key cryptography, the memory storage requirement in SACK-P and in [13] can be computed as given in Table V. Notice that d_m is the maximum neighborhood degree in [13], which equals to 7.

Now let us use an example to compare the storage requirement of our scheme, SACK-P and proposed scheme in [13]. Suppose that there are $N = 1369 \approx 1370$ CNs and $c = 10$ clusters in a hierarchical WSN. We use a $TD(25, 37)$ to provide key chains for CHs and CNs in such a network. And also assume that ECC (163-bit) is used for asymmetric cryptography and the RC5 (80 bit) is used in symmetric cryptography. Hence, from Table VI, memory requirement for each CH and CN can be estimated as acquired values in Table VI. Acquired values confirm that SACK-P needs memory storage 7 and 11 times more than our scheme for each CH and CN, respectively, and memory storage requirement for each CH in [13] is about 69 times of our scheme. So, we can easily claim that our proposed scheme requires much less total storage space than the two other schemes.

5.1.2. Communication and computation overhead.

Analyzing the efficiency of key management schemes in a WSN, we are interested in the communication overhead, the amount of transmitted messages between nodes to establish pairwise keys in key setup phase, and the computational overhead which describes the amount of computations (encryption and decryption) that must be performed by network nodes during the pairwise key establishment.

The number of exchanged messages to establish pairwise keys for our scheme, SACK-P and [13] is reported in Table VII. In the proposed scheme, at the predeployment stage the keys are stored on sensor nodes. In the key setup phase, the reconstruction of new keys merely requires the concatenation with a *challenge* α_j which is broadcasted by each CH in its cluster. So, the overall exchanged message to establish pairwise keys in key setup phase in our scheme equals to c, where c is the number of clusters. Now let us take N as the number of sensor nodes in a WSN. In SACK-P, each SN sends a message to its pertinent CH to join it, the result of which would be a total N unicast message. Following the reception of each message of the relevant CN by the CH, The CH forwards it to the

Table V. Comparison of three schemes for memory storage requirements.

Scheme	Memory requirement for each cluster head	Memory requirement for each cluster node
Proposed scheme	$(\sqrt{N} + \sqrt{c}) \times B^k$	$n \times B^k$
SACK-P	$(m + c + 1) \times B^u$	$(m + 2) \times B^u$
[13]	$(2 + N) \times B^u$	$(c + 2) \times B^u + (d_m \times B^k)$

Table VI. Aquired values for memory storage requirements for our scheme, SACK-P and [13].

Scheme	Number of stored bits on each cluster head	Number of stored bits on each cluster node
Proposed scheme	$(37 + 3) \times 80 = 3200$	$25 \times 80 = 2000$
SACK-P	$(137 + 10) \times 163 = 23961$	$(137 + 2) \times 163 = 22657$
[13]	$(1369 + 2) \times 163 = 223743$	$(12 \times 163) + (7 \times 80) = 2516$

Table VII. Comparison of three schemes for communication overhead.

Scheme	Number of exchanged messages in key setup		
Proposed scheme	c broadcast		
SACK-P	$c + 1$ broadcast	+	$3N$ unicast
[13]			$\frac{cmd_m}{2}$ unicast

BS so that an N number of unicast messages would be obtained. BS in response to this message unicasts an OK or a REVOKE message to the CH, so we have N unicast message in this step too. Broadcasting the BS public key to all the CHs and the CHs public key to its CNs necessitates $c + 1$ broadcast message. So, the total exchanged message in SACK-P is $3N$ unicast and $c + 1$ broadcast message. In represented scheme in [13], when secure clustering was carried out and routing algorithm was determined, because each CH is aware of the one-hop neighbors of its CNs, it unicasts the symmetric key of each CN and its one-hop neighbors to its CNs. So, if d_m shows the maximum neighborhood degree and m shows the number of nodes in each cluster, then the number of exchanged messages to establish secure pairwise keys will be equal to $cmd_m/2$. On the basis of these results, we can say that our scheme imposes less communication overhead to establish secure pairwise keys between nodes than two other schemes.

Aside from less communication overhead, to establish pairwise keys, our scheme needs less computations. In other words, in SACK-P and [13], establishing pairwise keys imposes expensive computational overhead [3], whereas storing keys on nodes predeployment would be unneeded to design such calculations. For example in [13], communication between each CN and its neighbors is subject to unicast of required common keys by each relevant CH, which necessitates d_m encryption operations for each CN. Whereas in our scheme, two neighboring CNs are able to discover their common keys only with exchange of their identifiers in O(1), with no need of cryptographic operations. Additionally, encryption and decryption operations in SACK-P and [13] are performed on the basis of ECC which needs much computational energy, while symmetric key cryptography algorithm in proposed scheme consumes much less computational energy [3].

5.2. Comparison with represented schemes in [14] and [17]

In this part, the resiliency of our scheme has been compared with two presented schemes in [14,17], which are referred to as BIBD and TD scheme, respectively, for networks of comparable sizes.

In [14], Camtepe and Yener calculate the probability of compromising a secure link under the BIBD scheme as

$$fail(s) = 1 - \frac{\binom{q^2}{s}}{\binom{q^2+q+1}{s}}$$

where $q+1$ is the number of preloaded keys in each sensor and s is the number of captured nodes.

In [17], a $TD(n,p)$ is used as a KPS, where n is the number of preloaded keys in each sensor and n^2 is the total number of nodes in the network. The probability of compromising a secure link under a TD scheme was computed as:

$$fail(s) = 1 - \left(1 - \frac{p-2}{p^2-2}\right)^s$$

In order to compare the performance of three mentioned schemes in a similar setting, we normalize by fixing the number of keys per node, k, and the local connectivity parameter $Pr1$ of each scheme. In order to obtain the same values of k and $Pr1$ for the different schemes, we set parameters, which are summarized in Table VIII. For each scheme considered, we also list the maximum network size denoted by N.

In Figure 7, we plot the probability that an adversary can decrypt the communications between two sensor nodes when s sensor nodes are compromised. Figure 7 shows that the same keys preloaded in a sensor node under the BIBD and TD schemes, the larger the compromising probability, that is, less resilient to node compromise attack. For example, while capture of 100 nodes treats more than 90% of links in BIBD and TD shemes, only 22% of links are affected in our proposal, which is 1/4 times of that in BIBD and TD. It can be even much better with increment of the clusters number as explained a bit earlier in Section4.2. Thus, our proposed key management scheme is very resilient against node compromise attack.

Table VIII. Parameter sets for comparison of resiliency.

	Parameters	k	N	$Pr1$
our scheme	$n = 37$	37	$1369CN + 10CH =$	0.97
	$p = 37$		1379	
BIBD	$q = 37$	38	1407	1
TD	$n = 37$	37	1369	0.97
	$p = 37$			

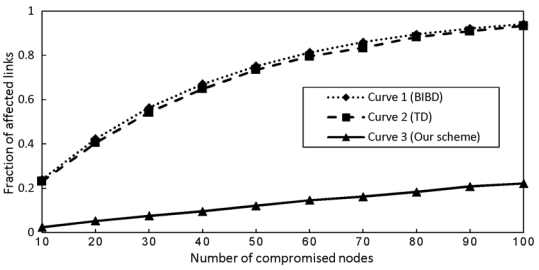

Figure 7. Fraction of compromised links between uncompromised nodes versus number of compromised nodes.

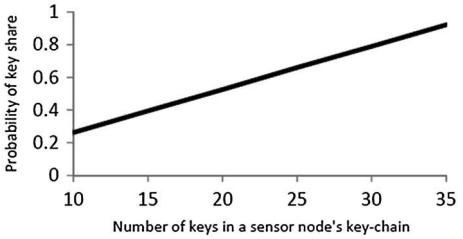

Figure 8. Local connectivity comparison of proposed scheme with transversal design scheme.

As the Figure 8, and the achieved values for $Pr1$ in Table VIII confirm, the local connectivity in our scheme remains the same as well in TD scheme, which means our proposed scheme greatly improves the resiliency against node capture without sacrificing the local connectivity (note that Figure 8 is composed of two curves coincided).

6. CONCLUSION

This study proposes a key distribution scheme that is suitable for clustered hierarchical WSNs. We take advantage of combinatorial designs to associate key chains to sensor nodes before deployment and reducing complexity of shared key discovery algorithm after deployment.

In our scheme, the key establishment is completed after deployment by incorporating a pseudo-random generated number by each CH and using a hash function. Concatenating challenge α_j causes sensor nodes in one cluster not to have any common key with clusters nodes in other clusters that reduce the effect of capture node. The hash function makes it possible to compress data into a fixed length and avoid data collision. This function also makes it impossible for the adversary to easily detect the pseudo random number in node compromise attack.

We show by simulations, in our scheme, that the network connectivity can be varied if the key chain size is changed. Because the size of key chain can be varied, it can be chosen such that, furthermore, the network connectivity remains in an acceptable level, there is no need to store a large number of keys on each node, and in this way, the nodes storage constraints are considered. Also, it has been demonstrated that the resiliency extremely improves when the number of clusters increases, although this improvement does not have any negative effect on the network connectivity. Consequently, with reasonable choice of key chain size and number of clusters, WSN can achieve higher resiliency and system performance.

REFERENCES

1. Kavitha T, Sridharan D. Hybrid design of scalable key distribution for wireless sensor networks. *International Journal of Engineering and Technology* 2010; **2** (2): 136–141.
2. Chong C, Kumar S. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE* 2003; **91**(8): 1247–1256.
3. Wang Y, Attebury G, Ramamurthy B. A survey of security issues in wireless sensor networks. *IEEE Communications Surveys & Tutorials* 2006; **8**: 2–23.
4. Eschenauer L, Gligor V. A key-management scheme for distributed sensor networks, *Proceedings of the 9th ACM conference on Computer and communications security*, New York, 2002; 41–47.
5. Camtepe S, Yener B. Combinatorial design of key distribution mechanisms for wireless sensor networks. *European Symposium on Research Computer Security* 2004; **3193**: 293–308.
6. Lee J, Stinson D. A combinatorial approach to key predistribution for distributed sensor networks. *Wireless Communications and Networking Conference* 2005; **2**: 1200–1205.
7. Ruj S, Roy B. *Key predistribution using partially balanced designs in wireless sensor networks*, Vol. 4742. 5th International Symposium (ISPA), 2007, 431–445.
8. Dong JW, Pei DY, Wang XL. A Class of Key Predistribution Schemes Based On Orthogonal Arrays. *Journal of Computer Science and Technology* 2008; **23** (5): 825–831.
9. Gupta P, Kumar P. The capacity of wireless networks. *IEEE Transactions on Information Theory* 2000; **46** (2): 388–404.
10. Samundiswary P, Priyadarshini P, Dananjayan P. Performance evaluation of heterogeneous sensor networks. *International Conference on Future Computer and Communication* 2009; **1**: 264–267.
11. Huang JY, Liao IE, Tang HW. A forward authentication key management scheme for heterogeneous sensor networks. *Journal on Wireless Communications and Networking* 2011; Article ID 296704: 10 pages, DOI:10.1155/2011/296704.
12. Lu K, Qian Y, Guizani M, Chen H. A framework for a distributed key management scheme in heterogeneous wireless sensor networks. *IEEE Transactions on Wireless Communications* 2008; **7**(2): 639–647.
13. Azarderakhsh R, Reyhani-Masoleh A. Secure clustering and symmetric key establishment in heterogeneous wireless sensor networks. *Journal on Wireless Communications and Networking* 2011; Article ID 296704: 10 pages, DOI:10.1155/2011/893592.
14. Camtepe S, Yener B. Combinatorial design of key distribution mechanisms for wireless sensor networks. *IEEE/ACM Transactions on Networking* 2007; **15**(2): 346–358.
15. Mittal N, Belagodu T. On maximum key pool size for a key pre-distribution scheme in wireless sensor networks. *International Journal of Computers and Applications* 2009; **31**(1): 30–35.
16. Anderson I. *Combinatorial designs: construction methods*. Ellis Horwood: Chicester, U.K., 1990.
17. Lee J, Stinson D. On the construction of practical key predistribution schemes for distributed sensor networks using combinatorial designs. *The ACM Transactions on Information and System Security* 2008; **11**: 1–35.
18. Younis O, Fahmy S. HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Transactions on Mobile Computing* 2004; **3**(4): 366–379.

19. Martin K, Paterson M, Stinson D. Key predistribution for homogeneous wireless sensor networks with group deployment of nodes. *ACM Transactions on Sensor Networks* 2010; **7**(2): 27–46.

20. Riaz R, Naureen A, Akram A, Akbar A, Kim K H. A unified security framework with three key management schemes for wireless sensor networks. *Computer Communications* 2008; **31**(18): 4269–4280.

DOI: 10.1002/sec