We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



24

Group Key Managements in Wireless Sensor Networks

Ju-Hyung Son* and Seung-Woo Seo
LG Electronics, Seoul National University*
Korea

1. Abstract

A sensor network operating in open environments requires a network-wide group key for confidentiality of exchanged messages between sensor nodes. When a node behaves abnormally due to its malfunction or a compromise attack by adversaries, the central sink node should update the group key of other nodes. The major concern of this group key update procedure will be the multi-hop communication overheads of the rekeying messages due to the energy constraints of sensor nodes. Many researchers have tried to reduce the number of rekeying messages by using the logical key tree. In this chapter, we propose an energy-efficient group key management scheme called Topological Key Hierarchy (TKH). TKH generates a key tree by using the underlying sensor network topology with consideration of subtree-based key tree separation and wireless multicast advantage. Based on our detailed analysis and simulation study, we compare the total rekeying costs of our scheme with the previous logical key tree schemes and demonstrate its energy efficiency.

2. Introduction

Advances in wireless networking, embedded microprocessors, and micro-fabrication technology have enabled a new generation of large-scale sensor networks which target a range of commercial and military applications (Zhao & Guibas, 2004). A typical wireless sensor network is composed of a large number of sensor nodes and one or several sinks collecting data from them (Akyildiz et al., 2002). The sink broadcasts control messages to engage sensor nodes into specific tasks. In response to the control messages, sensor nodes deliver the requested data back to the sink collectively. Usually, a sink has lap-top class computing power while sensor nodes have very limited computing resources. Even though it can vary by specific application scenarios of a sensor network, most sensor nodes are powered by limited batteries. Therefore, energy efficiency should be considered as an important parameter during the design of services or protocols for a sensor network. This also applies to the design of security services. If we want to apply a security service to a sensor network, we should consider the computation & communication efficiencies of the service in addition to its security performances.

From the nature of multi-hop wireless communications, an adversary can easily eavesdrop exchanged messages to gather secret information from a sensor network. Furthermore, it can

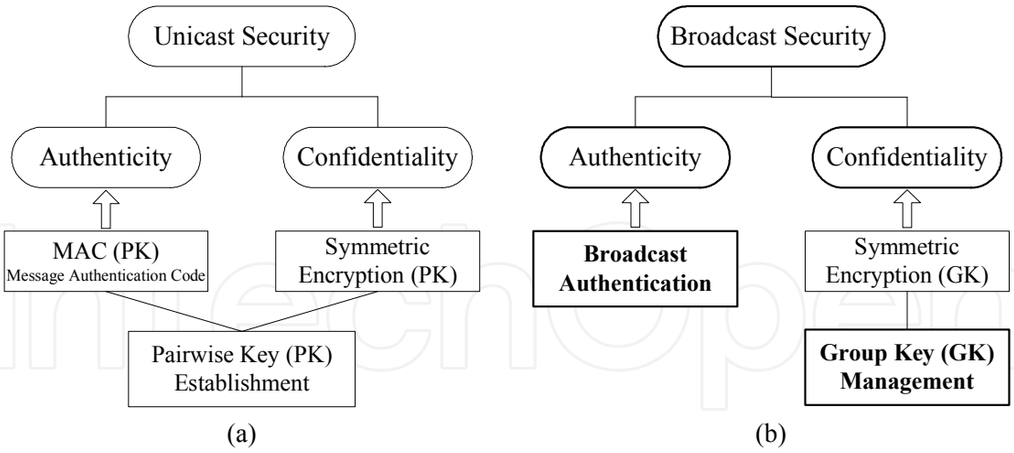


Fig. 1. Classification of security services according to communication patterns.

also inject malicious control messages to disrupt normal operations of sensor nodes. Therefore, in order to protect a sensor network from these attacks, we need *authenticity* and *confidentiality* of exchanged messages.

In general, the authenticity and confidentiality of messages are provided by different security primitives according to their underlying delivery mechanisms (*unicast* and *broadcast*) as depicted in Fig.1. The unicast security provides authentication/encryption of unicast messages between a pair of nodes. For the unicast security, we use MAC (Message Authentication Code) and symmetric encryption for the authenticity and confidentiality respectively while the both schemes use the same pairwise key (PK) established between a pair of nodes. Since there are many standard algorithms for the MAC and encryption such as HMAC (Bellare et al., 1997) and AES (U.S. DoC NIST, 2001) which can be efficiently run even on a sensor node, the problem of the unicast security converted to the pairwise key establishment problem.

The broadcast security provides authentication/encryption of broadcast messages between multiple nodes in a network. For the broadcast authenticity, we need new security primitives called *broadcast authentication*. We can provide the broadcast confidentiality by using the symmetric encryption with the network-wide group key (GK). Unlike the pairwise key, it is hard to secure the group key since it is known to every node in a network. Therefore, we need a new security primitive called *group key management* which updates a group key when a node joins/leaves a network.

While majority of researches on sensor network security have focused on pairwise key management schemes to provide the unicast security, there were not much research efforts to provide the broadcast security. However, the group security is more crucial security service compared to the unicast security for a sensor network from the specific characteristics of a sensor network:

- *Broadcast communications are commonly used in sensor networks*: Unlike the Internet environment where the server-client-based unicast communication type is prevalent, the broadcast communications are commonly used in a sensor network since all sensor nodes are usually controlled by a single administrative domain. A sink should continuously control sensor nodes by using the broadcast control messages, while there are not many scenarios where a pair of sensor nodes communicate each other.

- *Effects of the security breaches of the broadcast communications are much more devastating in sensor networks*: Without proper authentication mechanism, all sensors will react to the false control messages from adversaries which can exhaust energies of all sensor nodes. Also if a network-wide group key is compromised, all communications within a network can be eavesdropped by an adversary. However, revealed pairwise keys from a compromised node will not affect security of other links in a network. Therefore, while the effects of security breaches of the pairwise connections are negligible to other nodes, the compromise of the broadcast security affects the whole network.
- *Existing security primitives for the group security are not applicable to sensor networks*: The existing solutions for the group security such as digital signature (Rivest et al., 1978) and logical key hierarchy (LKH) (Wong et al., 1998) are inadequate to be applied to sensor network environments due to their computation and communication overheads. While the public key cryptography (PKC)-based pairwise key establishment scheme is also computationally heavy, it only occurs once when a pair of sensor nodes first contact. Thus sensors can adopt the PKC-based key establishment without much concerns in energy. However, communication & computation overhead of the broadcast security occurs frequently along with the broadcast messages which can exhaust energies of sensor nodes.

Based on the above motivations for the development of the group security mechanisms for wireless sensor networks, we investigate efficient & secure group key management schemes in this chapter.

2.1 Group Key Managements in Wireless Sensor Networks

In a wireless sensor network (WSN), many sensor nodes collect data from their surroundings, and report them to the central sink node (Akyildiz et al., 2002). The sink broadcasts control messages to sensor nodes to regulate their sensing/reporting operations. From these *many-to-1* and *1-to-many* communication characteristics, typical WSNs utilize a multicast tree topology rooted from the sink. For the design of communication protocol on this topology, the energy efficiency is the most important design principle due to sensor node's energy constraints. This also applies to the design of security services for WSNs. In addition to its security performances, a security service should take into account the energy efficiency of its protocol.

The message *confidentiality* is the imperative security primitive for various security services in a sensor network. Generally, a network-wide group key (GK) is used for message en/decryption for the message confidentiality. The sink should occasionally update GK to prevent a compromised node from decrypting messages. The simplest solution is to separately distribute a new GK to each node after encrypting it by each node's individual key (IK) that is only shared between each node and the sink. However, this will generate $O(N)$ rekeying messages with the network size N.

The Logical Key Hierarchy (LKH) scheme (Wallner et al., 1997) (Wong et al., 1998) reduces the number of rekeying messages to $O(\log N)$ by building a tree of key encryption keys (KEKs). Based on LKH, many researchers tried to further reduce the number of rekeying messages in trade-off of local key computations (Canetti et al., 1999) (Sherman & McGrew, 2003) (Lin et al., 2005). In these schemes, each node requires only several rekeying messages among the total rekeying messages according to its logical position in a key tree. However, in a multi-hop WSN where each node routes messages of other nodes, rekeying messages generated from the logical key tree should be forwarded to many irrelevant nodes before reaching their destinations. In other words, these logical key tree-based schemes can incur heavy communication

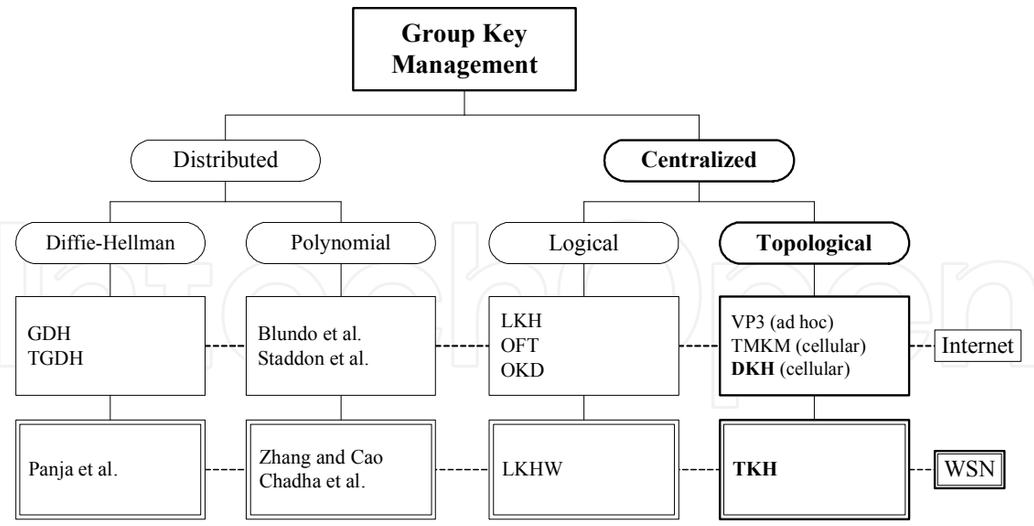


Fig. 2. Taxonomy of group key management schemes.

overheads in multi-hop WSN environments since the key tree structure does not reflect the underlying network topology.

In this chapter, we propse the Topological Key Hierarchy (TKH) scheme which generates a key tree from the sensor network's topology information. The basic principle is to enable topologically adjacent nodes in a network to share the same KEKs so that they can receive the same rekeying messages. Then each rekeying message can be delivered to its designated recipients while minimizing communication costs. While the previous group key management schemes only tried to minimize the number of rekeying messages, our TKH minimizes *the total rekeying cost* which reflects both *the number of rekeying message* and *the communication costs of rekeying messages*. We demonstrate the energy saving of TKH compared to the previous logical key tree-based schemes by using our detailed analysis and simulation study.

2.2 Related Works

A primary method to limit access to information within a group is the message encryption. Along with the message to be encrypted, we need a cryptographic key only shared within a group. Only those who knows the group key are able to decrypt the encrypted message. The most challenging problem of this scenario is to update the group key according to membership changes. We can divide the research literatures for this group key management problem according to their group key establishment style.

In the *Distributed* approaches, members generate a group key in contributory manner by combining their own secret information. Most of group key management schemes for sensor networks mainly focus on the distributed group key management schemes (Zhang & Cao, 2005) (Chadha et al., 2006) (Panja et al., 2006). In these schemes, sensor nodes collaboratively generate and update a group key without the help from a central sink node. However, establishing the group key in a large-scale network by using the distributed manner incurs much overheads. First, these schemes incur computational overheads since they use complex algorithms such as Polynomial (Blundo et al., 1992) (Staddon et al., 2002) or group Diffie-Hellman (Diffie & Hellman, 1976) (Steiner et al., 1996) (Kim et al., 2000) methods between sensor nodes. Also after the collaborative local group key generation procedure between neighbors, each lo-

cal group key should be merged with other local group keys to generate a single network-wide group key which requires multiple rounds of communications.
On the contrary, in the *Centralized* group key management scheme, a central key distribution center (KDC) randomly generates a new group key and produces related rekeying messages, which eliminates computation overheads of end nodes. Also, the communication costs are the one-time delivery costs of rekeying messages from the KDC to all nodes. Therefore, we think that the centralized group key management scheme is more preferable to a sensor network in terms of rekeying procedure's computation and communication overheads.
After introduction of the logical key hierarchy scheme independently by Wong et al. (Wong et al., 1998) and Wallner et al. (Wallner et al., 1997) in Internet environment, many researchers have tried to further reduce the number of rekeying messages by using the tradeoff between central rekeying and local computation (Sherman & McGrew, 2003) (Lin et al., 2005). In (Pietro et al., 2003), authors combined the directed diffusion data dissemination protocol (Intanagonwiwat et al., 2000) with LKH, and proposed LKHW (LKH for WSN) . LKHW is only compatible with the directed diffusion routing protocol. Our TKH can be applied to any tree-based routing algorithm .
Previously, Sun et al.(Sun et al., 2004) used topological information of a cellular network for efficient group key management. While the wired network part from KDC to each BS (base station) has abundant bandwidth which can easily carry $O(\log N)$ rekeying messages, the wireless network part from BS to each MN (mobile node) suffers from scarce bandwidth. The topological information on the latter part constantly changes due to the mobility of MNs. Therefore, the scheme in (Sun et al., 2004) is superior to the LKH when MN's mobility is not very high. On the contrary, a typical sensor network is a multi-hop wireless network which severely suffers from the limited bandwidth, and each sensor node does not have mobility in most scenarios. Therefore, our TKH can outperform the LKH in most conditions. Also, the hierarchical cellular network topology is quite different from the multi-hop wireless sensor network. Recently, Salido et al. (Salido et al., 2008) proposed VP3 (Vertex-Path, Power-Proximity) scheme for topology-based key management in multi-hop wireless ad hoc network environments. However, they assume dynamic power control capability of each node which is not the case for a sensor network. Also, they assume a subset of nodes belong to a certain group according to application scenarios where all nodes form a group in a sensor network.

3. Logical Key Tree-based Group Key Management

3.1 Logical Key Hierarchy & Related Schemes

The Logical Key Hierarchy (LKH) (Wallner et al., 1997) (Wong et al., 1998) is a centralized group key management scheme which utilizes the logical key tree. A key tree is maintained at the central KDC (Key Distribution Center) and the corresponding rekeying messages are delivered to all nodes when a node joins or leaves a group. A GK (Group Key) which is the root of a key tree is used to encrypt all data traffic within a group. KEKs (Key Encryption Keys) which reside in intermediate edges of a key tree are used to update the root GK and other KEKs. The leaves of a key tree are IKs (Individual Keys) which are individually shared by each node and the KDC. As a result, each node in a group possesses three kinds of keys: its own IK, KEKs (on the path to the root), and a root GK. Figure 3 denotes an example of the logical key tree. By using this example, let us examine the key tree update procedures of both '*Node Join*' and '*Node Leave*' events.

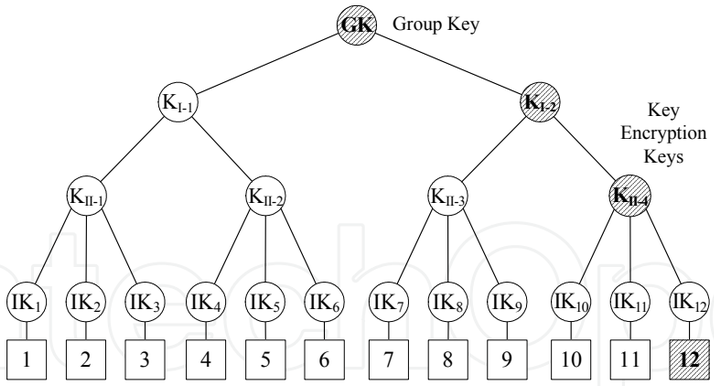


Fig. 3. A logical key tree example consisted of 12 nodes.

3.1.1 Node Join

First, let us assume that there were only eleven nodes initially in Figure 3, then the node 12 newly joins the group. Let $\{K_A\}_{K_B}$ denote key K_A encrypted by key K_B, and K' denote the updated version of key K. The keys that will be possessed by the joining node (GK, $K_{I–2}$, $K_{II–4}$) should be updated to prevent the node from decrypting the previously exchanged messages within the group (*Backward Secrecy*) (Kim et al., 2000). After rekeying messages $\{GK'\}_{GK}$, $\{K'_{I–2}\}_{K_{I–2}}$, $\{K'_{II–4}\}_{K_{II–4}}$ are sent to the existing members, the node 12 receives $\{GK'$, $K'_{I–2}$, $K'_{II–4}\}_{IK_{12}}$. However, the rekeying messages for the existing members can be safely replaced by local key computations (Waldvogel et al., 1999). Each subset of nodes can locally compute keys as $\{1 \sim 11\} : GK' = f(GK)$, $\{7 \sim 11\} : K'_{I–2} = f(K_{I–2})$, $\{10 \sim 11\} : K'_{II–4} = f(K_{II–4})$ with a common one-way function f. It means that the group key update for a node join event only incurs a rekeying message unicast to the joining node.

3.1.2 Node Leave

Second, let us assume that there were initially twelve nodes and the node 12 leaves the group. Then the possessed keys of the leaving node also should be updated to prevent the leaving node from decrypting the future messages (*Forward Secrecy*) (Kim et al., 2000). In this case, however, several current keys cannot be used in the rekeying procedure since the leaving node also knows them. Therefore, more complicated rekeying messages are generated and delivered to the remaining nodes. During the generation of the rekeying messages at KDC, there are two different rekeying strategies in LKH: *group-oriented rekeying* (LKH(g)) and *user-oriented rekeying* (LKH(u)) according to the underlying rekeying message delivery mechanisms (Wong et al., 1998)[1]:

$$\mathrm{LKH}(g) \begin{cases} m_{\mathrm{KDC}\rightarrow all} : \{GK'\}_{K'_{I–1}} || \{GK'\}_{K'_{I–2}} || \{K'_{I–2}\}_{K_{II–3}} \\ \qquad\qquad || \{K'_{I–2}\}_{K'_{II–4}} || \{K'_{II–4}\}_{IK_{10}} || \{K'_{II–4}\}_{IK_{11}} \end{cases} \tag{1}$$

$$\mathrm{LKH}(u) \begin{cases} m_{\mathrm{KDC}\rightarrow\{1\sim6\}} : \{GK'\}_{K_{I–1}} \\ m_{\mathrm{KDC}\rightarrow\{7\sim9\}} : \{GK', K'_{I–2}\}_{K_{II–3}} \\ m_{\mathrm{KDC}\rightarrow\{10\}} : \{GK', K'_{I–2}, K'_{II–4}\}_{IK_{10}} \\ m_{\mathrm{KDC}\rightarrow\{11\}} : \{GK', K'_{I–2}, K'_{II–4}\}_{IK_{11}}. \end{cases} \tag{2}$$

[1] The *key-oriented rekeying* defined in (Wong et al., 1998) is not considered in this chapter since it equals to the *user-oriented rekeying* in terms of the number of rekeying messages and their delivery mechanism.

In *the group-oriented rekeying*, KDC combines all rekeying messages and broadcasts the whole messages to all nodes. Upon receiving the whole messages, each node selects its messages and decrypts the necessary keys. In *the user-oriented rekeying*, KDC generates rekeying messages for each subset of nodes and multicasts (or unicasts) each rekeying message only to the corresponding subset of nodes. While the group-oriented rekeying generates the smaller number of rekeying messages in total, it incurs more communication overheads in multi-hop WSN since all sensors should receive and forward the whole messages. Even the user-oriented rekeying is more energy-efficient, it requires multicast routing protocol to deliver messages. Without the multicast support in WSNs, rekeying messages for a subset of nodes will be separately delivered to them by unicast.

McGrew and Sherman proposed an improvement over LKH called One-way Function Tree (OFT) (Sherman & McGrew, 2003). OFT reduces the number of rekeying messages from $(2\log_2 N)$ to $(\log_2 N)$ in the binary key tree by using the local key computations (Waldvogel et al., 1999) similar to the node join operation. However, OFT is susceptible to node collusion attacks (Horng, 2002) (Ku & Chen, 2003). There are similar approaches that achieve the same communication overhead as OFT without node collusion vulnerabilities: One-way Function Chain (OFC) (Canetti et al., 1999), and One-way Key Derivation (OKD) (Lin et al., 2005).

In the One-way Key Derivation, KDC reduces the number of rekeying messages by not sending the rekeying messages to nodes that can derive the keys by themselves. Therefore, when node 12 is revoked in Figure 3, the keys can be locally derived in each subset of nodes: $\{1\sim6\} : \mathrm{GK}' = f(\mathrm{K}_{\mathrm{I}\text{–}1} \oplus \mathrm{GK})$, $\{7\sim9\} : \mathrm{K}'_{\mathrm{I}\text{–}2} = f(\mathrm{K}'_{\mathrm{II}\text{–}3} \oplus \mathrm{K}_{\mathrm{I}\text{–}2})$, $\{10\} : \mathrm{K}'_{\mathrm{II}\text{–}4} = f(\mathrm{IK}_{10} \oplus \mathrm{K}_{\mathrm{II}\text{–}4})$. Here, f denotes a one-way function and $\oplus$ denotes an exclusive-or computation. After the local key computations, KDC transmits the corresponding rekeying messages to the remaining subset of nodes either by using *group-oriented rekeying* (OKD(g)) or *user-oriented rekeying* (OKD(u)) methods:

$$\mathrm{OKD}(g) \left\{ \; m_{\mathrm{KDC}\rightarrow all} : \{\mathrm{GK}'\}_{\mathrm{K}'_{\mathrm{I}\text{–}2}} || \{\mathrm{K}'_{\mathrm{I}\text{–}2}\}_{\mathrm{K}'_{\mathrm{II}\text{–}4}} || \{\mathrm{K}'_{\mathrm{II}\text{–}4}\}_{\mathrm{IK}_{11}} \right. \tag{3}$$

$$\mathrm{OKD}(u) \left\{ \begin{array}{l} m_{\mathrm{KDC}\rightarrow\{7\sim9\}} : \{\mathrm{GK}'\}_{\mathrm{K}'_{\mathrm{I}\text{–}2}} \\ m_{\mathrm{KDC}\rightarrow\{10\}} : \{\mathrm{GK}', \mathrm{K}'_{\mathrm{I}\text{–}2}\}_{\mathrm{K}'_{\mathrm{II}\text{–}4}} \\ m_{\mathrm{KDC}\rightarrow\{11\}} : \{\mathrm{GK}', \mathrm{K}'_{\mathrm{I}\text{–}2}, \mathrm{K}'_{\mathrm{II}\text{–}4}\}_{\mathrm{IK}_{11}}. \end{array} \right. \tag{4}$$

Comparing (1)(2) with (3)(4), it is evident that OKD reduces the number of rekeying messages in trade-off of the local key computations.

3.1.3 Total Rekeying Costs

When a group key management scheme properly updates a group key when a node joins or leaves the group as described above, the Backward Secrecy and Forward Secrecy properties are preserved (Kim et al., 2000). Since LKH, OKD, and our TKH are designed to preserve both properties, we argue that they are equal in terms of the security level. However, our TKH achieves the same security level with smaller amount of rekeying cost compared to the logical key tree based schemes including LKH and OKD.

To quantitatively compare the rekeying costs, we define the *Total Rekeying Cost (TRC)* of a group key management scheme as the product of *the number of rekeying messages* and *the communication costs of the rekeying messages*. Previously, most group key management schemes tried to reduce the number of rekeying messages (Rafaeli & Hutchison, 2003). However, it is also important to deliver rekeying messages efficiently to its designated recipients in multi-hop WSN environments. Generally, 1) OKD incurs smaller TRC compared to LKH due to the

reduced number of rekeying messages, and 2) the user-oriented rekeying incurs smaller TRC compared to the group-oriented rekeying since each node receives/forwards the smaller number of messages. However, *OKD's user-oriented rekeying*, currently the most communication-efficient logical key tree-based scheme, is not optimal in multi-hop WSN environments from the following reasons.

First, the multicast routing incurs heavy storage and communication overheads in WSN. Unlike the Internet environment where routers and end-hosts are separated in functionality, each sensor should act as both a router and an end-host in WSNs. Therefore, every sensor should maintain routes to all sensors to support multicast routing. This is infeasible for the resource constrained sensor nodes specifically in large scale networks. Second, even if the multicast routing is supported, it is hard to expect multicast advantage (minimally using the network resources before reaching multiple destinations) with the logical key tree-based schemes. For example, if nodes $\{7, 8, 9\}$ receiving $\{\mathrm{GK}'\}_{\mathrm{K}'_{\mathrm{I}-2}}$ in equation (4) are distinctly located in a network, this one multicast session will incur the similar multi-hop communication overheads as three unicast sessions to each of them. To overcome these constraints, we propose Topological Key Hierarchy that does not require multicast routing protocol and utilize multicast advantage by mapping the topological neighbors to the key tree neighbors.

4. Topological Key Hierarchy

In this section, we provide design principles, key tree generation, and key tree update procedures of Topological Key Hierarchy. TKH operates without the multicast routing and minimizes the network usages by using the topology-mapped key tree structure.

4.1 Design Principles

In the key tree-based schemes, the nodes sharing the same KEK mostly receive the same rekeying messages. In order to assign a KEK for a group of topologically adjacent nodes, we use two kinds of tree topology information: *Subtree* and *Sibling* information.

4.1.1 Subtree-based Key Tree Separation (Tree Key)

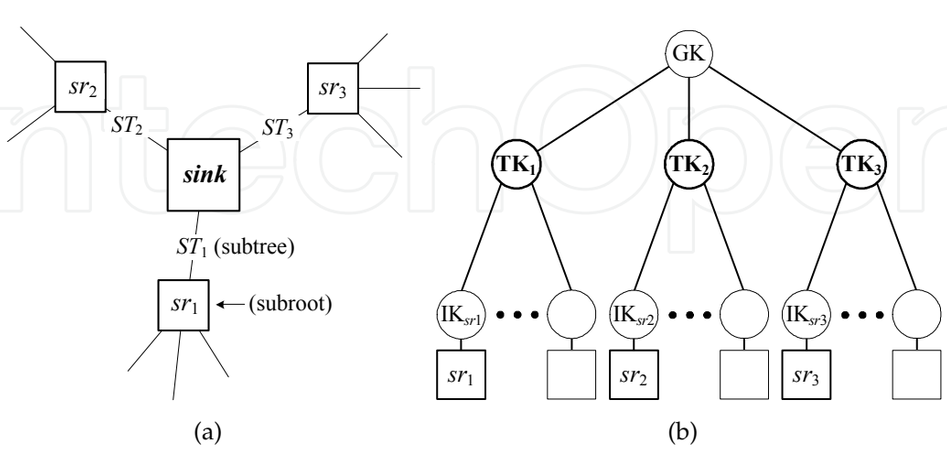


Fig. 4. (a) A sensor network topology and (b) the corresponding TK assignment.

First, we make the nodes in the same subtree share the same KEK called *Tree Key* (TK). The subtree is a tree with nodes below each subroot node, where subroot nodes are direct neighbors of a sink. The sample sensor network topology and its tree key assignment is depicted in Figure 4. From the three subtree branches, three tree keys (TK_1, TK_2, TK_3) are mapped to nodes in each subtree. From this key tree separation, rekeying messages for each subtree will be different from those of other subtrees. It means that TKH separates rekeying messages and delivers each subset only to the corresponding subtree. Nodes in each subtree are required to receive and forward rekeying messages only destined to nodes in their subtree.

4.1.2 Wireless Multicast Advantage Utilization (Sibling Key)

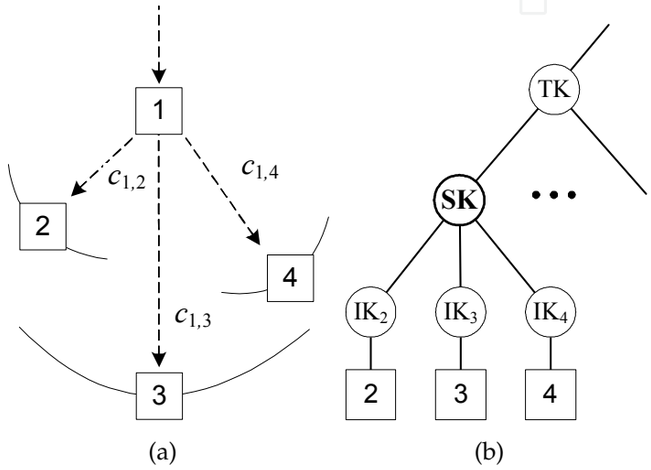


Fig. 5. a) A sensor network topology and (b) the corresponding SK assignment.

Second, we make the nodes sharing the same parent node in a tree topology (*sibling nodes*) to share the same KEK called *Sibling Key* (SK). For a node in a tree, a parent node is a neighbor node that delivers messages from the root sink node. In a wireless medium, since a message transmission can be heard by multiple neighbors, sibling nodes can efficiently receive a message by a single transmission from their parent. For example in Figure 5.(a) where node 1 has three one-hop neighbors $\{2, 3, 4\}$ in a wireless network, the costs of multicasting a single message to them is $C_{\text{multicast}} = \max(c_{1,2}, c_{1,3}, c_{1,4})$ where $c_{i,j}$ is a unicast cost from node i to j. Therefore, the one-hop multicast in a wireless medium can save energy from the broadcast nature of a wireless medium.

However, the important necessary condition for this wireless multicast advantage is that the message destined to neighbors should be the same. In other words, even if we have n one-hop neighbors which can be heard simultaneously, if the messages destined to them are different from each other, we have no choice but to unicast the messages one-by-one to each recipient. For rekeying messages generated from a key tree, we can make the same message to be destined to specific nodes by locating them under the same KEK. Therefore, we make children nodes of a parent node to share a SK to utilize the wireless multicast advantage.

4.2 Key Tree Generation

Based on the previous design principles, constructing a TKH key tree is composed of three steps: 1) *Routing Tree Construction*, 2) *Routing Tree Learning*, and 3) *Key Tree Generation*. How-

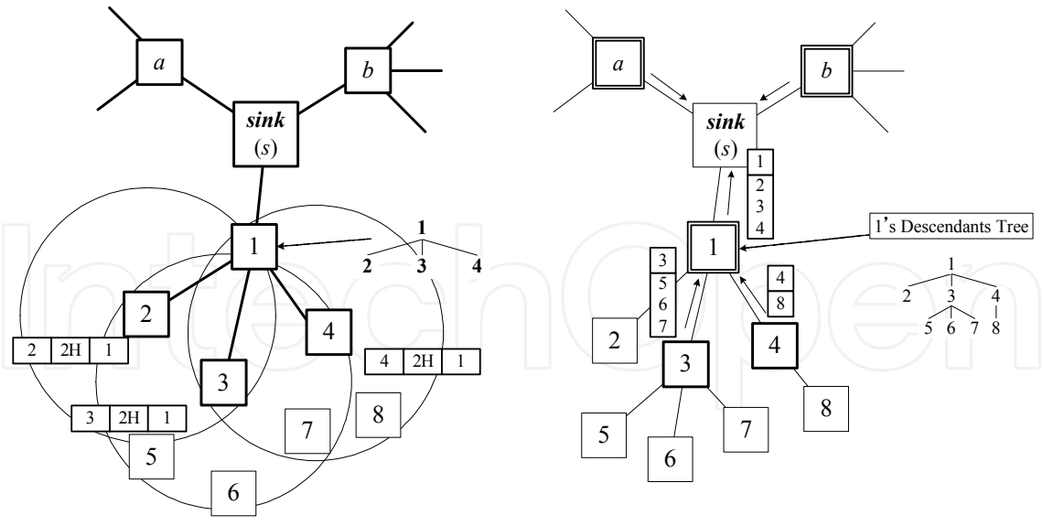


Fig. 6. (a) Routing Tree Construction and (b) Routing Tree Learning procedures.

ever, if a sensor network is already employing the tree-based routing and a central sink knows the topology information, TKH does not require the first two steps. For example, in a ZigBee-based WSN utilizing the tree-based hierarchical routing (ZigBee Alliance, 2006), the central sink can immediately generate the topology-based key tree by using the current topology information. If a WSN does not operate a tree-based routing, TKH needs to setup a sink-based routing tree to generate a topology-mapped key tree. Also the constructed routing tree will be used to deliver rekeying messages afterwards.

4.2.1 Routing Tree Construction

Constructing an efficient multicast source tree has been an active research area both in wired (Diot et al., 1997) and wireless (Wieselthier et al., 2002) networks. Here we introduce a simple routing tree construction method while TKH can generate a key tree from any routing tree construction method. After sensor node deployment, a sink broadcasts *Cost Advertisement (CA)* message to make sensor nodes to setup paths to the sink node. Each CA message contains three information: 1) node ID, 2) hop count to the sink, and 3) parent node ID. For example in Figure 6.(a), the node 3's CA message is '$[3|2H|1]$' since node '3' is '2 Hops' away from the sink through the parent node '1'. After hearing CA messages, a node chooses its parent node which has the minimum hop count to the sink (if multiple CA messages have the same hop count value, a node can choose the CA message received with the highest SNR). After selecting a parent node, each node also broadcasts its own CA message to neighbors. By overhearing CA messages, a parent node can learn the association of its children nodes with itself. In Figure 6.(a), by overhearing CA messages of nodes $\{2,3,4\}$, node 1 learns that it is associated with three children nodes. This routing tree construction procedure continues until it reaches all nodes.

4.2.2 Routing Tree Learning

After construction of a tree topology, every parent node reports *Parent-Child Relationship* (PCR) message to the sink. Each PCR message contains two information: 1) parent node ID and 2)

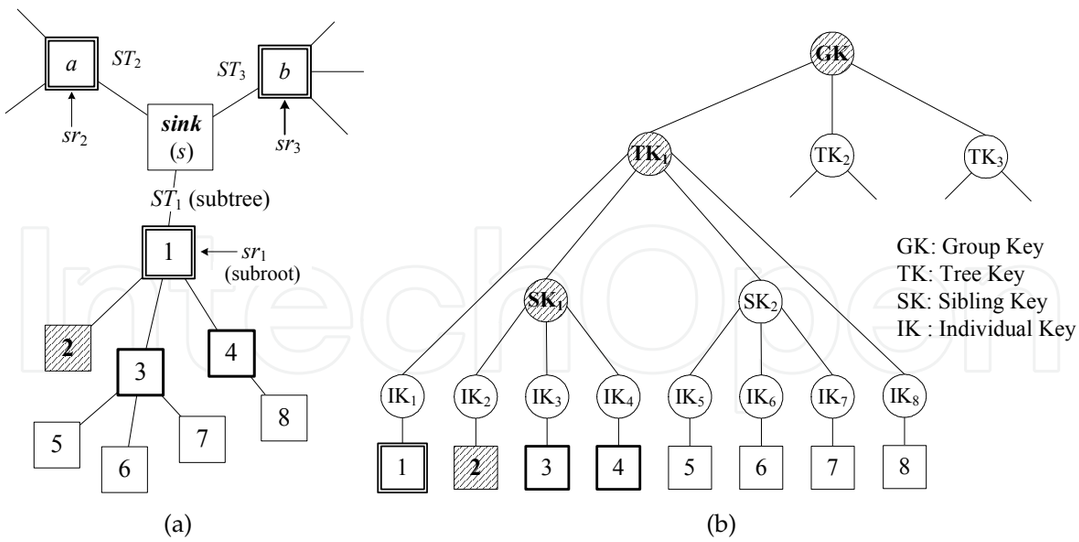


Fig. 7. (a) A sensor network tree topology example and (b) the corresponding TKH key tree structure. We depict the keys that need to be updated as shaded circles when node 2 is revoked.

children node IDs. For example in Figure 6.(b), node 1's PCR message is $[1|2,3,4]$ since it has three children nodes. After collecting all PCR messages, the sink can learn the whole network topology like Figure 6.(b). Also, during the PCR message forwardings, each parent node can learn and save its descendant node IDs in *Descendants Tree*. For example, by overhearing PCR messages from node 3 and 4, node 1 can build its *Descendants Tree* like in Figure 6.(b). By maintaining this tree, each parent can only forward messages destined to its descendants which prevents redundant message forwarding. Therefore, the routing overhead of TKH is only to maintain *Descendants Tree* in each parent node.

4.2.3 Key Tree Generation

Based on the topology information obtained from the previous tree learning procedure, now the sink can build a topology-based key tree. Before describing the key tree generation procedure, we first define several parameters (we show an example of each parameter by using the sample topology of Figure 7.(a)): We describe the key tree generation algorithm of TKH in Figure 8. As an example, Figure 7.(b) depicts the corresponding key tree structure generated from the topology of Figure 7.(a). In addition to GK and IK, Tree Key (TK) is shared by nodes in the same subtree (ST) and Sibling Key (SK) is shared by nodes in the same sibling set (ss). TKH has an advantage that the depth of the key tree is bounded to '4' independent of the network size. Therefore, each sensor is only required to save maximum four keys which are beneficial for storage-limited sensor nodes. In contrast, the logical key tree-based schemes should increase the depth of the key tree according to the network size in order to maintain the optimal tree degree (LKH and OKD achieve the best performance with the tree degree of 4 and 2 respectively (Li et al., 2001) (Lin et al., 2005)). Therefore, they should increase the number of keys in each sensor node as network grows.

parameter	definition
$\mathbb{T}$	a tree topology with a sink at its root and sensors at vertices
N	the total number of sensor nodes in $\mathbb{T}$
l	a number of revoked sensor nodes during a rekeying interval
sr_i	i-th subroot node (e.g. $sr_1=1, sr_2=a, sr_3=b$ in Figure 7.(a))
ST_i	i-th subtree with sr_i as the subroot
N_i	a set of all nodes in ST_i (e.g. $N_1=\{1,2,3,4,5,6,7,8\}$)
$ss_{i,j}$	j-th sibling set in ST_i (nodes connected to the same parent) a single child consists a *single-node sibling set* without SK assignment; (e.g. $ss_{1,1}=\{1\}, ss_{1,2}=\{2,3,4\}, ss_{1,3}=\{5,6,7\}, ss_{1,4}=\{8\}$)
rn_i	a set of revoked nodes in ST_i (e.g. $rn_1=\{2\}$)
rns_i	a set of revoked node's sibling nodes in ST_i (e.g. $rns_1=\{3,4\}$)
RST	a set of subtrees which have revoked nodes in its vertices (e.g. $RST=\{ST_1\}$)
e_{tx}	energy dissipated during 1-bit transmission by a sensor node
e_{rx}	energy dissipated during 1-bit reception by a sensor node
$cu_{i,j}$	wireless unicast cost delivering 1-bit from node i to j ($cu_{i,j}=e_{tx}+e_{rx}$)
$cm_{i,\{1,\cdots,n\}}$	wireless multicast cost delivering 1-bit from node i to its n neighbors, ($cm_{i,\{1,\cdots,n\}}=e_{tx}+n\cdot e_{rx}$)

Table 1. Parameters for TKH algorithm explanation.

4.3 Key Tree Update

When a sensor node is newly deployed or revoked, a routing tree and the corresponding key tree should also be updated. One may think that the sink does not need to update the group key when a sensor node dies due to energy exhaustion. However, it is secure to update the group key also in this scenario since it is hard to verify by the remote sink whether the non-responding sensor node is pretending to be energy-less due to compromise attack. Therefore, we assume that the revocation of a sensor node take places when it is compromised or it runs out of energy.

Key tree update is composed of three steps: 1) *Routing Tree Repair*, 2) *Routing Tree Re-learning*, and 3) *Key Tree Update*. However, if a sensor network is already employing a tree-based routing or if node join or revocation events do not affect the topology of the remaining nodes, TKH does not require the first two steps.

4.3.1 Routing Tree Repair

When a node joins or leaves a network, a routing tree of the remaining node can be modified according to the node's topological position.

– *Node Join:* A newly deployed sensor node firstly broadcasts *join request* to neighbors. Then each neighbor reply CA messages containing its hop count to the sink. After selecting the parent node, the new node sends its CA message containing the parent ID. Then the selected parent reports a new PCR message to the sink which then locates the new node to the key tree according to its topological position. A joining node can either 1) *create a new single-node sibling set* or 2) *join the existing sibling set*. In both cases, the existing nodes can change the corresponding GK, TK, and SK by using the pre-shared one-way function same as the node

Input: a tree topology $\mathbb{T}$, all nodes' individual keys (IKs)
Output: a key tree

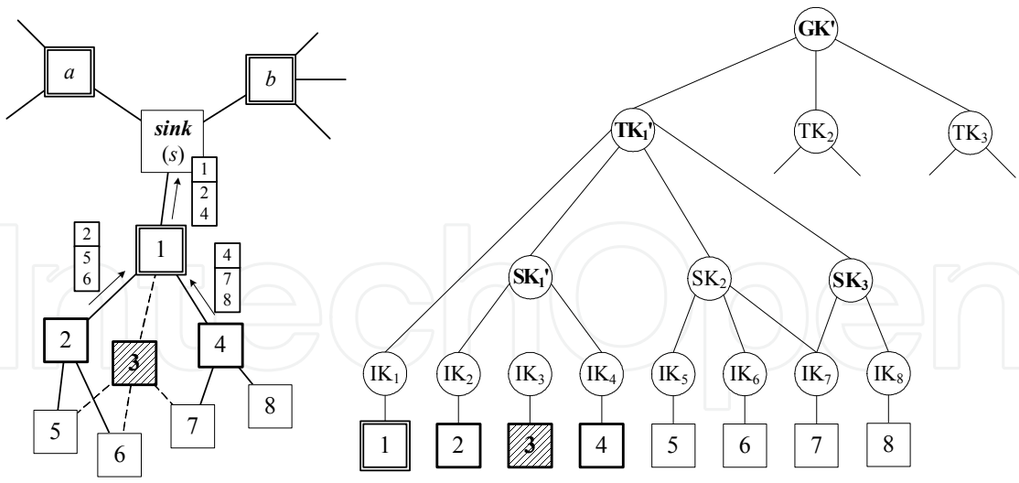
1) generate a group key (GK)
2) **for** (each ST_i) **do**
  if $|N_i| = 1$ **then**
    attach sr_i's IK to GK
  else ($|N_i| \geq 2$)
    generate a new tree key TK_i and attach it to GK
    for each $ss_{i,j}$ in ST_i **do**
      if $\left|ss_{i,j}\right| = 1$ **then**
        attach IK of the node in $ss_{i,j}$ to TK_i
      else $\left(\left|ss_{i,j}\right| \geq 2\right)$
        generate a new sibling key (SK) and attach it to TK_i
        attach all IKs of nodes in $ss_{i,j}$ to SK
      end if
    end for
  end if
end for
3) return a key tree

Fig. 8. Key Tree Generation Algorithm of TKH.

join procedure in Section 2.1.1. The new node receives the corresponding keys from the sink afterwards. Therefore, we do not consider the node join event since the topology change and the corresponding rekeying cost is negligible.
– *Node Revocation:* We further classify the node revocation event into 1) *leaf node revocation* and 2) *non-leaf node revocation*. The leaf node revocation does not affect the topology of the remaining nodes and the sink can send the rekeying messages based on the current key tree. For example in Figure 7.(a), revocation of the leaf node '2' does not affect the network topology, and rekeying messages can be generated from the current key tree of Figure 7.(b). However, the non-leaf node revocation can disconnect the network topology, and the sink should wait until the orphaned nodes of the revoked parent find new parent nodes. For the routing tree repair, each orphaned node performs the same procedure as the node join case.

4.3.2 Routing Tree Re-learning

If the sink revokes a non-leaf parent node, it waits until it receives new PCR messages from new parents of the orphaned nodes. After receiving PCR messages, the sink modifies the current key hierarchy based on the modified network topology. For example in Figure 9.(a), after revocation of node 3, the sink waits until it receives new PCR messages containing the orphaned nodes $\{5, 6, 7\}$. Then node 2 and 3, new parents of $\{5, 6, 7\}$ report their new PCR messages to the sink. Also by overhearing these new PCR messages, other nodes along the path to the sink modifies their *Descendants Tree*. Finally, the sink can send the rekeying messages based on the modified key tree structure.



(a) (b)

Fig. 9. After non-leaf node 3 in Figure 7 is revoked, a) the repaired routing tree with the re-learning procedure and b) the modified key tree structure.

4.3.3 Key Tree Update

Based on the modified key tree structure, the sink send the corresponding rekeying messages to each subset of nodes. By using the example of Figure 9, we examine the rekeying message delivery procedures in detail. When the non-leaf node 3 in ST_1 is revoked, rekeying messages (m) and the corresponding communication cost (C) to deliver m from the sink (s) to its recipients are

$$\begin{cases} m_{s\to\{1\}} : \{\mathrm{GK}', \mathrm{TK}'_1\}_{\mathrm{IK}_1} \\ m_{s\to\{2,4\}} : \{\mathrm{GK}', \mathrm{TK}'_1\}_{\mathrm{SK}'_1} \\ m_{s\to\{5,6\}} : \{\mathrm{GK}', \mathrm{TK}'_1\}_{\mathrm{SK}_2} \\ m_{s\to\{7,8\}} : \{\mathrm{GK}', \mathrm{TK}'_1\}_{\mathrm{SK}_3} \\ m_{s\to 2} : \{\mathrm{SK}'_1\}_{\mathrm{IK}_2} \\ m_{s\to 4} : \{\mathrm{SK}'_1\}_{\mathrm{IK}_4} \\ m_{s\to 7} : \{\mathrm{SK}_3\}_{\mathrm{IK}_7} \end{cases} \quad \begin{cases} C_{s\to\{1\}} : e_{tx}+e_{rx} \\ C_{s\to\{2,4\}} : 2e_{tx}+3e_{rx} \\ C_{s\to\{5,6\}} : 3e_{tx}+4e_{rx} \\ C_{s\to\{7,8\}} : 3e_{tx}+4e_{rx} \\ C_{s\to 2} : 2e_{tx}+2e_{rx} \\ C_{s\to 4} : 2e_{tx}+2e_{rx} \\ C_{s\to 7} : 3e_{tx}+3e_{rx}. \end{cases}$$

Rekeying messages for ST_2 and ST_3 are $\{\mathrm{GK}'\}_{\mathrm{TK}_2}$ and $\{\mathrm{GK}'\}_{\mathrm{TK}_3}$ respectively. Upon receiving each rekeying message, a node can route it to one of its children nodes based on its *Descendants Tree*. Nodes in the same sibling set $(\{2,4\},\{5,6\},\{7,8\})$ will receive the same rekeying messages by using the wireless multicast advantage from their parents.

Comparing Figure 7.(b) and Figure 9.(b), we observe that the sibling sets sharing SK_2 and SK_3 are slightly changed. However, TKH does not update SK_2 and SK_3 since none of the sensors sharing them are revoked. By maintaining the link from node 7 to SK_2 in the key tree, the sink can update both SK_2 and SK_3 later when node 7 is revoked. Finally, the total rekeying cost (TRC) of ST_1 is calculated as

$$\begin{aligned} \mathrm{TRC}_{ST_1} &= 2|m|\times\left(C_{s\to\{1\}}+C_{s\to\{2,4\}}+C_{s\to\{5,6\}}+C_{s\to\{7,8\}}\right) \\ &+ |m|\times(C_{s\to 2}+C_{s\to 4}+C_{s\to 7}+C_{s\to 8}) = |m|\,(25e_{tx}+31e_{rx})\,. \end{aligned}$$

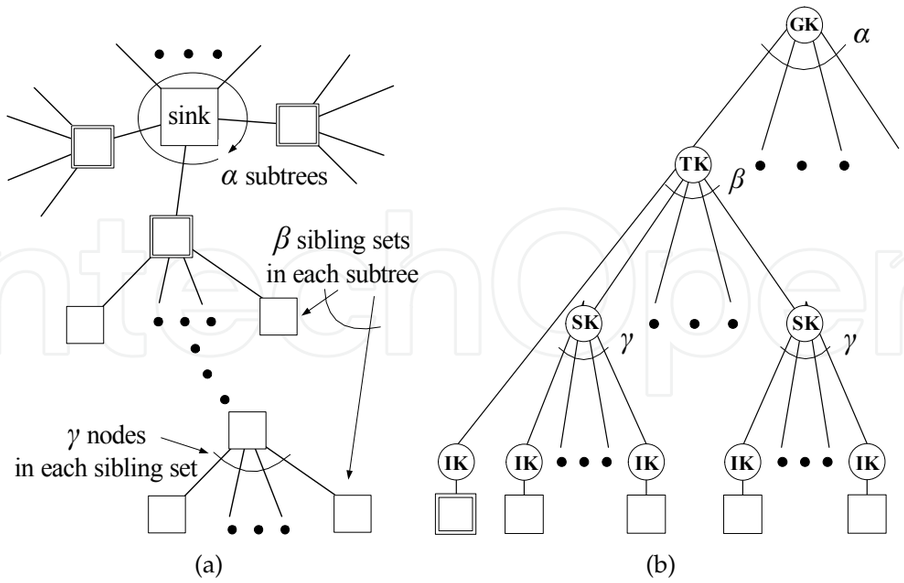


Fig. 10. (a) '$\alpha\beta\gamma$-tree' and (b) the corresponding TKH key tree structure.

where $|m|$ is the size of a unit rekeying message $\{K_A\}_{K_B}$ ($2|m|$ for $\{K_A, K_B\}_{K_C}$). It means that we need 25 transmissions and 31 receptions of a unit rekeying messages to update ST_1 when node 3 is revoked.

5. Analysis of the Total Rekeying Cost

In this section, we analyze and compare the total rekeying costs of LKH, OKD, and TKH in multi-hop WSN environments. For the analysis, we need to derive the average number of rekeying messages and the communication costs. The former is derived in Section 4.3 by employing *the bins-and-balls problem*. To calculate the latter, we model a typical WSN topology as '$\alpha\beta\gamma$-tree' in Section 4.1. Both results are used to derive the total rekeying costs in Section 4.4 while the communication costs of the routing tree maintenance are calculated in Section 4.2.

5.1 '$\alpha\beta\gamma$-tree' Topology Model

For the analysis of the communication cost, we model a sensor network topology by using '$\alpha\beta\gamma$-tree' model. In the $\alpha\beta\gamma$-tree, there are 'α' subtree branches from the sink, and each subtree has 'β' sibling sets, and each sibling set has 'γ' sibling nodes. The resulting topology and the corresponding TKH key tree structure is depicted in Figure 10.(a) and (b) respectively. The total number of sensor nodes excluding a sink is $N = \alpha(\beta\gamma+1)$ and each subtree has $(\beta\gamma+1)$ nodes. Among N sensor nodes, $(\alpha\beta)$ nodes are non-leaf parents and the rest $(\alpha(\beta\gamma+1)-\alpha\beta)$ nodes are leaf children nodes. During the routing tree repair in $\alpha\beta\gamma$-tree, we assume that a revoked non-leaf parent node is replaced by one of its siblings, and a revoked subroot node is replaced by one of its children.

5.2 Cost of Routing Tree Maintenance

When there are N nodes in a network, each node can be identified by using $\lceil \log_2 N \rceil$ bits, and the hop count value ranging from 0 to β can be identified by $\lceil \log_2 \beta \rceil$ bits. Then the size of the CA message ($|m_{\mathrm{CA}}|$) and the PCR message ($|m_{\mathrm{PCR}}|$) are respectively

$$|m_{\mathrm{CA}}| = 2\lceil \log_2 N \rceil + \lceil \log_2 \beta \rceil, \quad |m_{\mathrm{PCR}}| = (\gamma+1)\lceil \log_2 N \rceil$$

where $\lceil x \rceil$ denotes the smallest integer equal or greater than x ($\lfloor x \rfloor$ denotes the largest integer equal or smaller than x).

5.2.1 Routing Tree Construction & Learning

The communication cost of the 'Routing Tree Construction & Learning' (C_{CL}) defined in Section 3.2.1 and 3.2.2 is derived as

$$C_{\mathrm{CL}} = |m_{\mathrm{CA}}| \left\{ (N+1) \cdot e_{tx} + N\gamma \cdot e_{rx} \right\} + |m_{\mathrm{PCR}}| \left\{ \alpha\beta \cdot \mathrm{avg}(1,\beta)(e_{tx}+e_{rx}) \right\} \tag{5}$$

where $\mathrm{avg}(1,n) = \frac{1+2+\cdots+n}{n} = \frac{(n+1)}{2}$ $\left(\mathrm{sum}(1,n) = 1+2+\cdots+n = \frac{n(n+1)}{2} \right)$. We assume that every sensor plus the sink broadcast one CA message and each sensor receives γ CA messages on average. PCR messages are generated by all $\alpha\beta$ parent nodes and they require $\mathrm{avg}(1,\beta)$ hops to reach the sink.

5.2.2 Routing Tree Repair & Re-Learning

Also the communication cost of the 'Routing Tree Repair & Re-learning' (C_{RR}) defined in Section 3.3.1 and 3.3.2 is derived as follows

$$C_{\mathrm{RR}} = |m_{\mathrm{PCR}}| \sum_{i=0}^{\min(l,\alpha\beta)} \frac{C_i^{\alpha\beta} C_{l-i}^{\alpha(\beta\gamma+1)-\alpha\beta}}{C_l^{\alpha(\beta\gamma+1)}} \times i \times \mathrm{avg}(1,\beta)(e_{tx}+e_{rx}) \tag{6}$$

where C_b^a is the binomial coefficient. Among the total $\alpha(\beta\gamma+1)$ nodes, only revocations of $\alpha\beta$ parent nodes incur new PCR message reports. The corresponding m_{PCR} should be delivered to the sink along $\mathrm{avg}(1,\beta)$ hops.

5.3 Average Number of Rekeying Messages

5.3.1 Basic Functions

When l nodes are revoked, $\overline{\mathbb{B}}(l,v,w)$ calculates *the average number of intermediate KEKs that need to be updated*. v is the total number of intermediate KEKs at a certain key tree level, where each KEK on that level is shared by w nodes. By analogy, $\overline{\mathbb{B}}(l,v,w)$ is equivalent to the average number of *non-full* bins when l balls are randomly picked out from v identical bins each filled with w balls. The picked-out balls represent revoked nodes and the non-full bins represent KEKs need to be updated. The number of non-full bin ($\overline{n}(l,v,w)$) is in the range of $\lceil l/w \rceil \le \overline{n}(l,v,w) \le \min(l,v)$. Then, $\overline{\mathbb{B}}(l,v,w)$ is represented as

$$\overline{\mathbb{B}}(l,v,w) \triangleq E[\overline{n}(l,v,w)] = \sum_{i=\lceil l/w \rceil}^{\min(l,v)} \Pr\{\overline{n}(l,v,w) = i\} \times i.$$

In the above equation,

$$\Pr\{\overline{n}(l,v,w) = i\} = C_i^v \cdot N(l,i,w) / C_l^{vw}$$

where $N(l,i,w)$ is the number of ways that there is no full bins when l balls are picked out from i bins containing w balls each. $N(l,i,w)$ is calculated by using the inclusion-exclusion principle (Tucker, 1995, Ch. 3) which results

$$\overline{\mathbb{B}}(l,v,w) = \sum_{i=\lceil l/w \rceil}^{\min(l,v)} \frac{C_i^v \cdot \left(\sum_{j=0}^{i-\lceil l/w \rceil} (-1)^j C_j^i C_l^{w(i-j)} \right)}{C_l^{vw}} \times i. \tag{7}$$

Another function $\underline{\mathbb{B}}(l,v,w)$ calculates *the average number of intermediate KEKs that do not need to be updated* since all the nodes shared the same KEK are revoked. $\underline{\mathbb{B}}(l,v,w)$ is equivalent to the average number of *empty* bins when l balls are randomly picked out from v identical bins each filled with w balls, and calculated as

$$\underline{\mathbb{B}}(l,v,w) = \sum_{i=\max(l-vw+v,0)}^{\lfloor l/w \rfloor} \frac{C_i^v \cdot \left(\sum_{j=0}^{\lfloor l/w \rfloor - i} (-1)^j C_j^{v-i} C_{l-w(i+j)}^{w(v-i-j)} \right)}{C_l^{vw}} \times i. \tag{8}$$

Finally, $\mathbb{B}(l,v,w)$ defined as the difference between (7) and (8) is the actual average number of intermediate KEKs that need to be updated on a certain key tree level when l nodes are revoked

$$\mathbb{B}(l,v,w) = \overline{\mathbb{B}}(l,v,w) - \underline{\mathbb{B}}(l,v,w). \tag{9}$$

While the analysis in this subsection is motivated by the previous results (Sun et al., 2004, Appendix A), we improve them in that 1) we provide non-recursive, closed-form solutions for the bins-and-balls problem and 2) we also analyze the number of KEKs that do not need to be updated by introducing $\underline{\mathbb{B}}(l,v,w)$.

5.3.2 Average Number of Rekeying Messages

We denote a key tree of N nodes as $T(d_1,\cdots,d_h)$ where d_i is the degree of a vertex at the i-th level from the top and h is the height of the tree $(\because d_1 \times \cdots \times d_h = N)$. For example, the key tree in Figure 3 is denoted as $T(2,2,3)$. For the simplicity in equations, we assume $d_0 = d_{h+1} = 1$. When l nodes are revoked, the average number of total rekeying messages of LKH and OKD generated by group-oriented rekeying are respectively

$$|M_{\mathrm{LKH}(g)}| = \left\{ d_1 + \sum_{i=1}^{h-1} \overline{\mathbb{B}}\left(l, \prod_{j=1}^{i} d_j, \prod_{k=i+1}^{h} d_k \right) \cdot d_{i+1} - l \right\} - \left\{ \sum_{i=1}^{h-1} \underline{\mathbb{B}}\left(l, \prod_{j=1}^{i} d_j, \prod_{k=i+1}^{h} d_k \right) \right\} \tag{10}$$

$$|M_{\mathrm{OKD}(g)}| = \left\{ (d_1 - 1) + \sum_{i=1}^{h-1} \overline{\mathbb{B}}\left(l, \prod_{j=1}^{i} d_j, \prod_{k=i+1}^{h} d_k \right) \cdot (d_i - 1) - l \right\}. \tag{11}$$

With parameters $N = 12, d_1 = 2, d_2 = 2, d_3 = 3, h = 3, l = 1$ of Figure 3, the number of rekeying messages are calculated as $|M_{\mathrm{LKH}(g)}| = 6$ and $|M_{\mathrm{LKH}(g)}| = 3$ by using the above (10) and (11), and they are consistent with (1) and (3) respectively.

For the user-oriented rekeying, we assume that each rekeying message is delivered to its recipient by unicast without multicast routing support in WSNs. For example in (13), $m_{\mathrm{KDC}\to\{7\sim9\}} : \{\mathrm{GK}'\}_{\mathrm{K}'_{1-2}}$ is calculated as 3 rekeying messages unicast to $(7,8,9)$ independently. When l nodes are revoked, the average number of total rekeying messages of LKH and OKD generated by user-oriented rekeying are respectively

$$|M_{\mathrm{LKH}(u)}| = \sum_{i=1}^{h} \left\{ i \times \left(\overline{\mathbb{B}}\left(l, \prod_{j=0}^{i-1} d_j, \prod_{k=i}^{h} d_k \right) \cdot d_i - \overline{\mathbb{B}}\left(l, \prod_{j=1}^{i} d_j, \prod_{k=i+1}^{h+1} d_k \right) \right) \cdot \left(\prod_{k=i+1}^{h+1} d_k \right) \right\} \tag{12}$$

i	m_i	$\lvert m_i\rvert$	tx_i	rx_i (dest+relay)
1	$\{\mathrm{GK}\}_{\mathrm{TK}}$	$\alpha - \underline{\mathbb{B}}(l,\alpha,\beta\gamma+1)$	$\beta+1$	$(N-l)+0$
2	$\{\mathrm{TK}\}_{\mathrm{SK}}$	$\mathbb{B}(l,\alpha,\beta\gamma+1)(\beta+1)$	$\mathrm{avg}(1,\beta+1)$	$N_R(l)+\mathbb{B}(l,\alpha,\beta\gamma+1)\cdot\mathrm{sum}(1,\beta)$
3	$\{\mathrm{SK}\}_{\mathrm{IK}}$	$N_r(l)$	$\mathrm{avg}(2,\beta+1)$	$N_r(l)+N_r(l)\cdot\mathrm{avg}(1,\beta)$

Table 2. For each rekeying message (m_i) in TKH, the number of rekeying messages $(|m_i|)$, the number of transmissions per message (tx_i), and the total number of receptions at destinations and relay nodes (rx_i) are derived.

$$|M_{\mathrm{OKD}(u)}| = \sum_{i=0}^{h-1}\Bigg\{ i\times\left(\overline{\mathbb{B}}\left(l,\prod_{j=0}^{i}d_j,\prod_{k=i+1}^{h}d_k\right)\cdot d_{i+1} - \overline{\mathbb{B}}\left(l,\prod_{j=1}^{i+1}d_j,\prod_{k=i+2}^{h+1}d_k\right)\right) + \tag{13}$$

$$\left(\overline{\mathbb{B}}\left(l,\prod_{j=0}^{i}d_j,\prod_{k=i+1}^{h}d_k\right)\cdot(d_{i+1}-1) - \overline{\mathbb{B}}\left(l,\prod_{j=1}^{i+1}d_j,\prod_{k=i+2}^{h+1}d_k\right) + \underline{\mathbb{B}}\left(\left\langle l/\prod_{j=i+2}^{h+1}d_j\right\rangle,\prod_{j=0}^{i}d_j,d_{i+1}\right)\right)^{+}\Bigg\}\left(\prod_{j=i+2}^{h+1}d_j\right)$$

In (13), $(x)^+$ is defined as $\{x \text{ if } x \geq 0, 0 \text{ if } x < 0\}$ and $<x>=\lfloor x+0.5\rfloor$. With the parameters of Figure 3, the number of rekeying messages are calculated as $|M_{\mathrm{LKH}(u)}|=18$ and $|M_{\mathrm{OKD}(u)}|=8$ by using the above (12) and (13), and they are consistent with (2) and (4) respectively.
TKH has three kinds of rekeying messages (m_i): $\{\mathrm{GK}\}_{\mathrm{TK}}$, $\{\mathrm{TK}\}_{\mathrm{SK}}$, and $\{\mathrm{SK}\}_{\mathrm{IK}}$. From the key tree structure of Figure 10.(b) generated from the $\alpha\beta\gamma$-tree topology, the average number of rekeying messages are calculated in $|m_i|$ column of Table 2.

5.4 Total Rekeying Costs

By using both the previous results on the average number of rekeying messages and the $\alpha\beta\gamma$-tree model for calculation of the communication costs, we derive the total rekeying costs of LKH, OKD, and TKH as follows

$$\mathrm{TRC}_{\mathrm{LKH}(g)} = |M_{\mathrm{LKH}(g)}|\times\{\alpha(\beta+1)\cdot e_{tx}+(N-l)\cdot e_{rx}\}|m| \tag{14}$$

$$\mathrm{TRC}_{\mathrm{OKD}(g)} = |M_{\mathrm{OKD}(g)}|\times\{\alpha(\beta+1)\cdot e_{tx}+(N-l)\cdot e_{rx}\}|m| \tag{15}$$

$$\mathrm{TRC}_{\mathrm{LKH}(u)} = |M_{\mathrm{LKH}(u)}|\times\Big\{\mathrm{avg}(1,\beta)\cdot(e_{tx}+e_{rx})\Big\}|m|+C_{\mathrm{CL}}+C_{\mathrm{RR}} \tag{16}$$

$$\mathrm{TRC}_{\mathrm{OKD}(u)} = |M_{\mathrm{OKD}(u)}|\times\Big\{\mathrm{avg}(1,\beta)\cdot(e_{tx}+e_{rx})\Big\}|m|+C_{\mathrm{CL}}+C_{\mathrm{RR}} \tag{17}$$

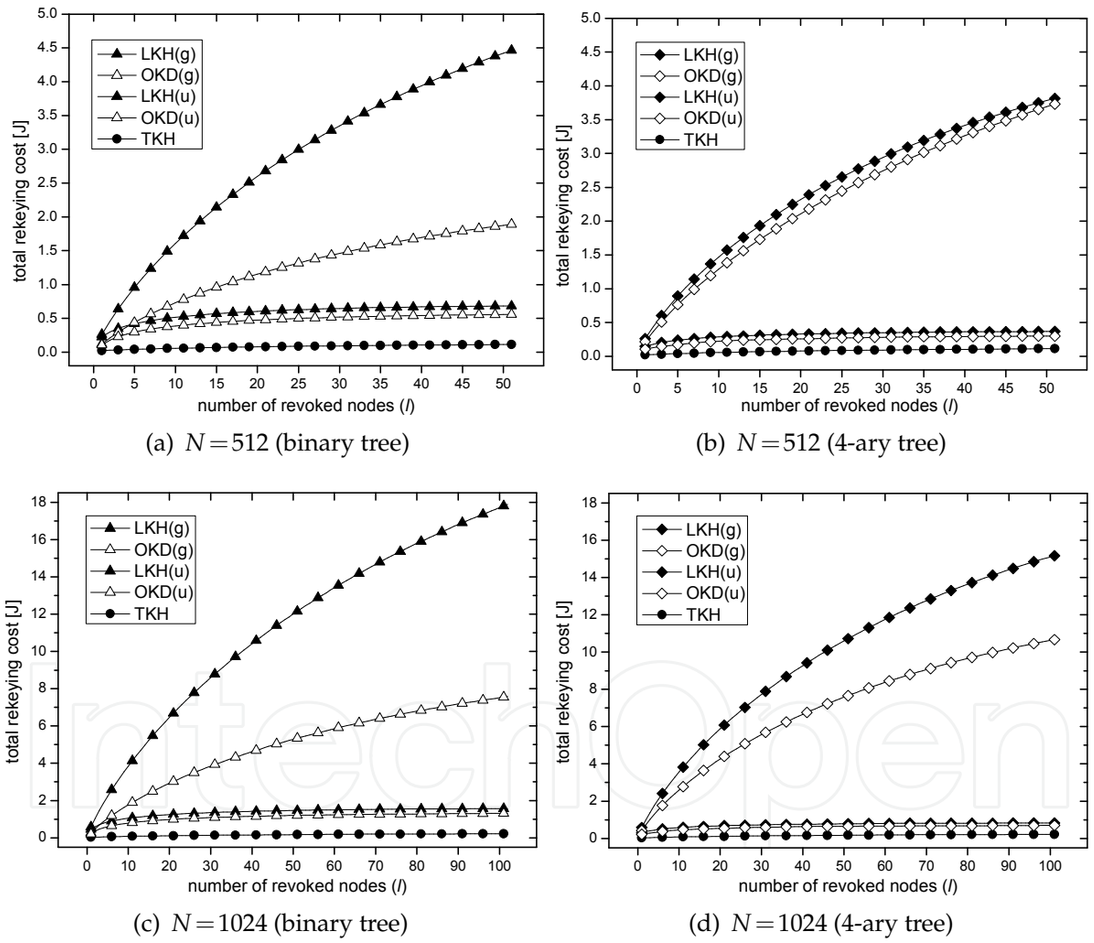
$$\mathrm{TRC}_{\mathrm{TKH}} = \sum_{\forall m_i}\Big\{(|m_i|\times tx_i)\cdot e_{tx}+(rx_i)\cdot e_{rx}\Big\}|m|+C_{\mathrm{CL}}+C_{\mathrm{RR}}. \tag{18}$$

In group-oriented rekeying ((14) and (15)), all rekeying messages are broadcast to all nodes requiring $\alpha(\beta+1)$ transmissions and $(N-l)$ receptions within a network. In user-oriented rekeying ((16) and (17)), each rekeying message is independently unicast to each node requiring $\mathrm{avg}(1,\beta)$ transmissions and receptions on average. While the group-oriented rekeying is independent of the network topology, the user-oriented rekeying and TKH requires topology information to deliver rekeying messages (reflected by $C_{CL}+C_{RR}$ in (16), (17), and (18)). Therefore, LKH(u), OKD(u), and TKH requires additional C_{CL} and C_{RR} costs in the total rekeying cost. In TKH, for each rekeying message (m_i), we calculate the average number of rekeying messages $(|m_i|)$, the number of transmissions per message (tx_i), and the total number of receptions at destinations and relay nodes (rx_i) in Table 2. Here $N_R(l)$ and $N_r(l)$ are defined

and derived as follows

$$N_R(l) = \{\text{avg. \# of nodes in revoked subtrees}\} \tag{19}$$
$$= \sum_{\forall ST_i \in RST} |N_i| = \mathbb{B}(l,\alpha,\beta\gamma{+}1)(\beta\gamma{+}1) - \Big(l - \underline{\mathbb{B}}(l,\alpha,\beta\gamma{+}1)(\beta\gamma{+}1)\Big)$$
$$N_r(l) = \{\text{avg. \# of revoked nodes' sibling nodes}\} \tag{20}$$
$$= \sum_{\forall ST_i \in RST} |rns_i| = \sum_{k=\max(0,l-\alpha\beta\gamma)}^{\min(l,\alpha)} \frac{C_k^{\alpha} C_{l-k}^{\alpha\beta\gamma}}{C_l^{\alpha(\beta\gamma+1)}} \Big(\overline{\mathbb{B}}(l-k,\alpha\beta,\gamma)\gamma - (l-k)\Big).$$

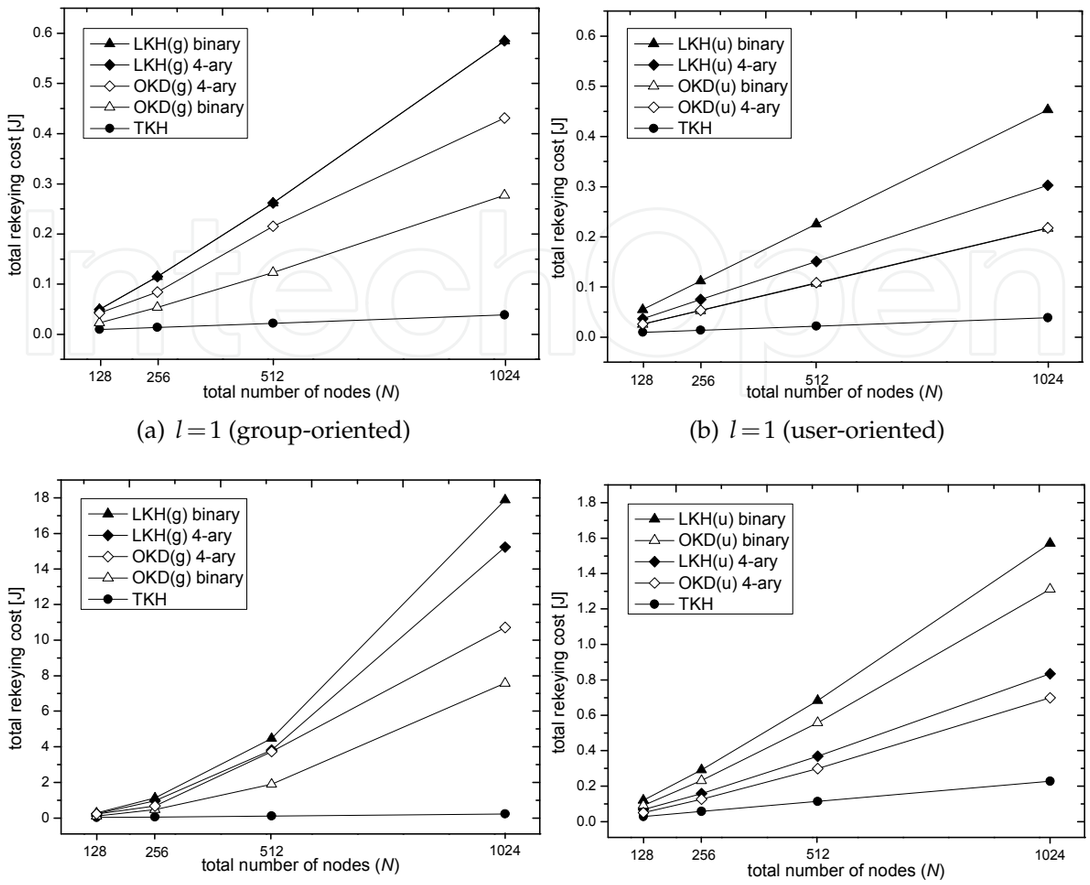
5.5 Analysis Results



(a) $N = 512$ (binary tree) (b) $N = 512$ (4-ary tree)

(c) $N = 1024$ (binary tree) (d) $N = 1024$ (4-ary tree)

Fig. 11. Total rekeying costs of LKH, OKD, and TKH.



(a) $l = 1$ (group-oriented) (b) $l = 1$ (user-oriented)

(c) $l = 0.1N$ (group-oriented) (d) $l = 0.1N$ (user-oriented)

Fig. 12. Total rekeying costs of LKH, OKD, and TKH.

We plot the total rekeying costs (TRC) of LKH, OKD, and TKH in Figure 11 and 12. We vary the total number of nodes (N) as 128, 256, 512, and 1024 by varying (α, β, γ) tuples as (2,7,9), (4,7,9), (8,7,9), and (16,7,9). We consider two logical key trees (binary and 4-ary) for LKH and OKD, while key trees of TKH are directly determined by (α, β, γ) values of each N. The unit rekeying message size is set to $|m| = 128$ bits. The unit communication costs are set to $e_{tx} = 0.209[\mu\text{J}]$ and $e_{rx} = 0.226[\mu\text{J}]$ from the characteristics of the CC2420 transceiver used in the Xbow's MICA-Z and Telos B sensor nodes.

Figure 11 depicts the increasing TRC values according to the increasing number of revoked nodes (l) when $N = 512, 1024$. For various number of the total nodes (N), Figure 12.(a) and (b) depict TRC when one node is revoked $(l = 1)$, and Figure 12.(c) and (d) depict TRC when 10% of nodes are revoked $(l = 0.1N)$. From combinations of three key tree schemes (*LKH, OKD, and TKH*), two rekeying strategies (*User-oriented and Group-oriented*), and two key tree structures (*binary and 4-ary*), we observe the following principles between them in terms of the total rekeying costs.

- TKH is superior to OKD and LKH in all cases.
- OKD is superior to LKH, given the same rekeying strategy and key tree structure.

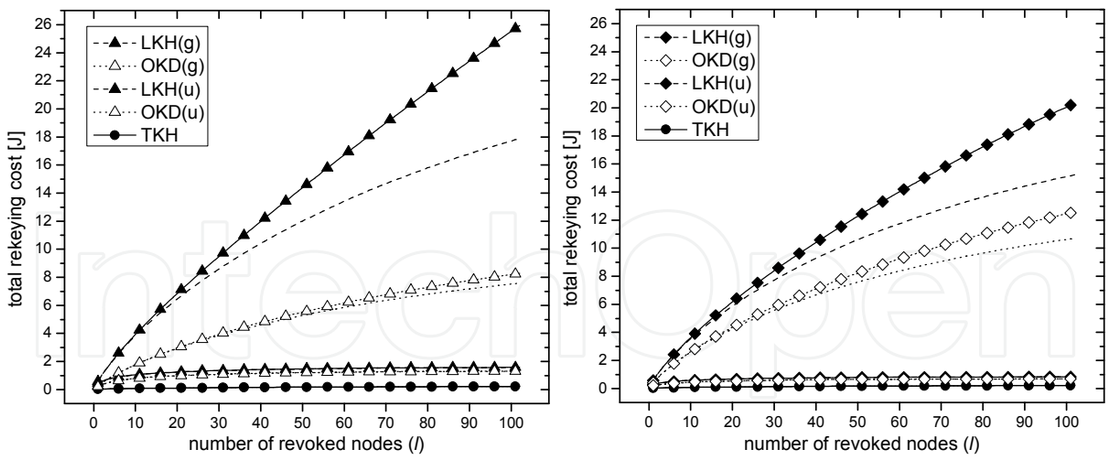
- User-oriented rekeying is superior to group-oriented rekeying, given the same logical key tree scheme, rekeying strategy, and key tree structure.
- For LKH, 4-ary key tree is superior to binary key tree independent of the rekeying strategies.
- For OKD(g), binary key tree is superior to 4-ary key tree. For OKD(u), 4-ary key tree is superior to binary key tree.

By considering the topological information during the key tree construction, TKH always incurs the lowest rekeying cost compared to the previous logical key tree schemes. Between the logical schemes, OKD is superior to LKH by reducing rekeying messages due to its local key computations. Since rekeying messages are individually delivered to each node in user-oriented rekeying, it is more energy-efficient than group-oriented rekeying which combines-and-broadcasts all rekeying messages. Given the same number of the total nodes, nodes in a 4-ary key tree only stores the half number of keys compared to those in a binary key tree. The reduced number of keys for each node translates into the reduced number of rekeying messages for each node in user-oriented rekeying. Therefore, we observe that LKH(u) and OKD(u) achieve lower rekeying costs when they utilize 4-ary key tree. However, while the 4-ary key tree is also optimal in LKH(g), it is inferior to binary key tree in OKD(g). This is due to the fact that binary key tree is optimal for OKD's local key computations in terms of the number of the total rekeying messages. Our results are consistent with the results of (Li et al., 2001) (Lin et al., 2005) that tried to find the optimal key tree structure for LKH and OKD in terms of the total number of rekeying messages.
In Figure 11.(a),(b),(c),(d) respectively, TKH only requires 17.7%, 32.1%, 14.5%, 26.6% of TRC compared to *OKD(u) with 4-ary* on average, while 13.9%, 25.4%, 11.8%, 21.8% of TRC compared to *LKH(u) with 4-ary* on average. Compared to the best logical key tree scheme: *OKD(u) with 4-ary*, TKH only requires 37.2%, 25.9%, 20.5%, 17.8% of TRC in Figure 12.(b) and 57.2%, 45.5%, 38.1%, 32.6% of TRC in Figure 12.(d) when $N = 128, 256, 512, 1024$ respectively.

5.6 Effects of Wireless Channel Errors

During message delivery between nodes in wireless sensor networks, it is probable that a transmitted message is corrupted due to wireless channel errors. Then the sender should retransmit the failed message and the receiver should retry to receive it which will consume additional communication costs at both sides. In LKH and OKD, group-oriented rekeying strategy uses multicast communications while user-oriented rekeying uses unicast communications to deliver rekeying messages. Our TKH utilizes the both communication methods according to rekeying message types: $\{\mathrm{GK}\}_{\mathrm{TK}}$ is delivered by multicast communications, while $\{\mathrm{TK}\}_{\mathrm{SK}}$ and $\{\mathrm{SK}\}_{\mathrm{IK}}$ are delivered by unicast communications. Therefore, message retransmissions incurred by wireless channel errors will have different effects on the total rekeying costs of the three schemes.
In unicast communications between a pair of wireless nodes, let p be the probability that a message is not received correctly at a receiver side (correctly received with $1-p$). If we assume the message length is L bits and bit error probability is p_b, p would be $p = 1-(1-p_b)^L$. Then the expected number of transmission attempts required to successfully deliver a message in wireless unicast ($\mathbb{E}(N_{\mathrm{U}})$) is

$$\mathbb{E}(N_{\mathrm{U}}) = 1\times(1-p)+2\times p(1-p)+3\times p^2(1-p)+\cdots = \frac{1}{1-p} = \frac{1}{(1-p_b)^L}$$



(a) $N = 1024$ (binary tree) (b) $N = 1024$ (4-ary tree)

Fig. 13. Effects of wireless channel error probability ($p_b = 10^{-5}$) on TRC according to the increasing number of revoked nodes (l) when $N = 1024$.

However, in multicast communications between a group of wireless nodes, the probability of a successful message reception would increase since a receiver can overhear multiple copies of a message not only from a sender but also from its neighbors. Let us assume that each node in multicast communications receives n copies of a message on average. Then the probability that a multicast message is not received correctly at a receiver side is p^n. Similar to (21), the expected number of transmission attempts required to successfully deliver a message in wireless multicast ($\mathbb{E}(N_{\mathrm{M}})$) is

$$\mathbb{E}(N_{\mathrm{M}}) = \frac{1}{1-p^n} = \frac{1}{1-(1-(1-p_b)^L)^n} \tag{21}$$

If we consider increased communication costs due to wireless channel errors, the total rekeying costs of group-oriented and user-oriented rekeying will be increased by the rates of $\mathbb{E}(N_{\mathrm{M}})$ and $\mathbb{E}(N_{\mathrm{U}})$ respectively, while that of TKH is affected by both. By applying $\mathbb{E}(N_{\mathrm{U}})$ and $\mathbb{E}(N_{\mathrm{M}})$ into the previous total rekeying costs in Section 4.4, we obtain

$$\mathrm{TRC}'_{\mathrm{LKH}(g)} = \mathrm{TRC}_{\mathrm{LKH}(g)} \times \mathbb{E}(N_{\mathrm{M}})_{\mathrm{LKH}(g)} \tag{22}$$

$$\mathrm{TRC}'_{\mathrm{OKD}(g)} = \mathrm{TRC}_{\mathrm{OKD}(g)} \times \mathbb{E}(N_{\mathrm{M}})_{\mathrm{OKD}(g)} \tag{23}$$

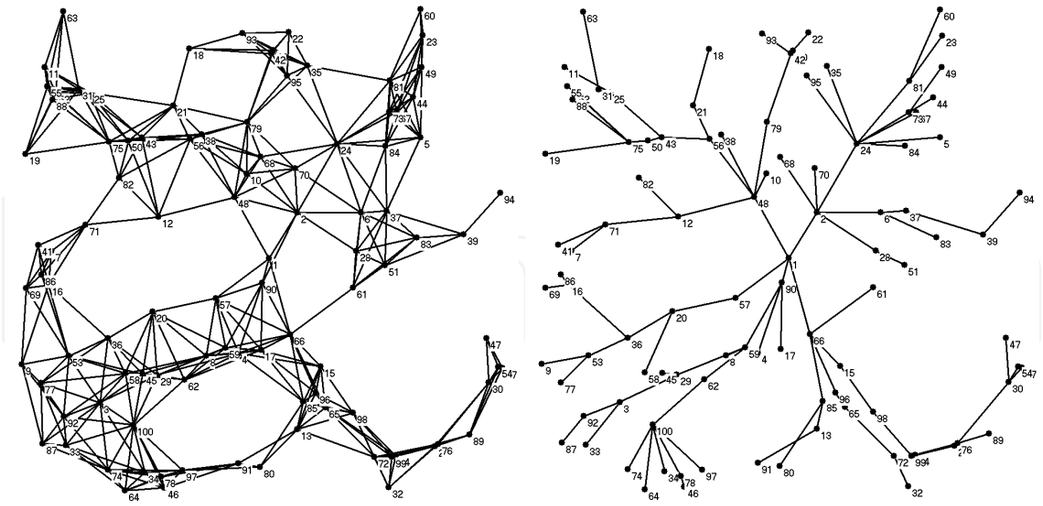
$$\mathrm{TRC}'_{\mathrm{LKH}(u)} = \mathrm{TRC}_{\mathrm{LKH}(u)} \times \mathbb{E}(N_{\mathrm{U}})_{\mathrm{LKH}(u)} \tag{24}$$

$$\mathrm{TRC}'_{\mathrm{OKD}(u)} = \mathrm{TRC}_{\mathrm{OKD}(u)} \times \mathbb{E}(N_{\mathrm{U}})_{\mathrm{OKD}(u)} \tag{25}$$

$$\mathrm{TRC}'_{\mathrm{TKH}} = \{(|m_1| \times tx_1) \cdot e_{tx} + (rx_1) \cdot e_{rx}\} \times \mathbb{E}(N_{\mathrm{M}})_{\mathrm{TKH}} \tag{26}$$

$$+ \left\{ \sum_{i=2}^{3} \{(|m_i| \times tx_i) \cdot e_{tx} + (rx_i) \cdot e_{rx}\} + C_{\mathrm{CL}} + C_{\mathrm{RR}} \right\} \times \mathbb{E}(N_{\mathrm{U}})_{\mathrm{TKH}}.$$

To calculate $\mathbb{E}(N_{\mathrm{M}})$ and $\mathbb{E}(N_{\mathrm{U}})$ in the above equations, we input message lengths (L) from (10)$\sim$(13) and $|m_i|$ equations in Table 2.



(a) (b)

Fig. 14. (a) A sample sensor network connectivity graph of 100 nodes in an 1×1 unit square area with r=0.171 (P_c=0.99). Sink node numbered as 1 is set to reside at the center of the area. (b) A multicast source tree generated from the topology of Figure 14.(a) by using the DSA heuristic.

Figure 13 depicts the increased TRC values due to the wireless channel error $(p_b = 10^{-5})$ according to the increasing number of revoked nodes when $N = 1024$. We assume that each node in multicast communication can hear two copies of a message on average $(n = 2)$. For comparison purpose, we also plot the original TRC values of LKH, OKD, and TKH as dash, dot, and solid lines respectively. Due to wireless channel errors, LKH(g) and OKD(g) obtain about 20% and 10% increases in their total rekeying costs respectively, while LKH(u) and OKD(u) only obtain about 1% additional rekeying costs. Since group-oriented rekeying combines-and-multicasts all rekeying messages simultaneously, it has a large message size. Therefore, it suffers more from wireless channel errors than user-oriented rekeying which delivers individual small rekeying messages to each node. By combining multicast and unicast communications and exploiting topological information, TKH is resistant to wireless channel errors (only 0.048% TRC increase in Figure 13). TKH's multicast delivery of $\{\mathrm{GK}\}_{\mathrm{TK}}$ is more error-tolerant than unicast since the message length is always '1' while it can have multicast advantage. Other two unicast rekeying message types ($\{\mathrm{TK}\}_{\mathrm{SK}}$, $\{\mathrm{SK}\}_{\mathrm{IK}}$) are also error-tolerant since they also have very small message sizes.

6. Simulation Results

In the previous section, we provided the analysis of the total rekeying costs based on the homogeneous '$\alpha\beta\gamma$-tree' topology model. In this section, we further investigate the rekeying costs of TKH and other schemes in more general and heterogeneous sensor network topology model.

Generating a typical sensor network multicast topology is consisted of two phases: *connectivity graph generation* and *multicast source tree generation*. First, we generate a wireless sensor network connectivity graph by using the *Random Geometric Graph* model (Penrose, 2003). Let

us assume that N sensor nodes are randomly deployed in an 1×1 unit square area. Each node has a common communication range of r, and a pair of nodes are connected if they reside within r to each other. The resulting network topology will be a graph (G) consisted of vertices (V) of sensors and edges (E) of wireless connectivity.

Under the given deployment area of a sensor network, increasing the number of nodes (N) or the communication range (r) will respectively increase the number of connections in the network. To obtain the appropriate value of r which connects N sensor nodes with the desired level of connectivity, we utilize the results from (Penrose, 1997). For N points placed uniformly at random on the unit square in the 2-dimensional space, Penrose (Penrose, 1997) found an asymptotic bound on the length of the longest edge (M_n) of MST (Minimum Spanning Tree) as follows

$$\lim_{N\to\infty} \text{Prob}\left[N\pi(M_N)^2 - \log N \leq c\right] = \exp(-e^{-c}) \tag{27}$$

with constant c. If we choose the communication range r the same as M_n, we can assure that the graph is almost surely connected with probability of $\exp(-e^{-c})$ because all the nodes have the communication range same as the longest edge of their MST. That is, given the value of N, if we set r as $N\pi r^2 - \log N = c$, "the probability that a given graph is connected" is $\exp(-e^{-c})$. This probability is a "connectivity" of a graph which is denoted as P_c. By setting c according to the desired level of connectivity, we can derive the communication range r. Figure 14.(a) depicts a sample sensor network connectivity graph of 100 nodes in a unit square area with r=0.171 (P_c=0.99).

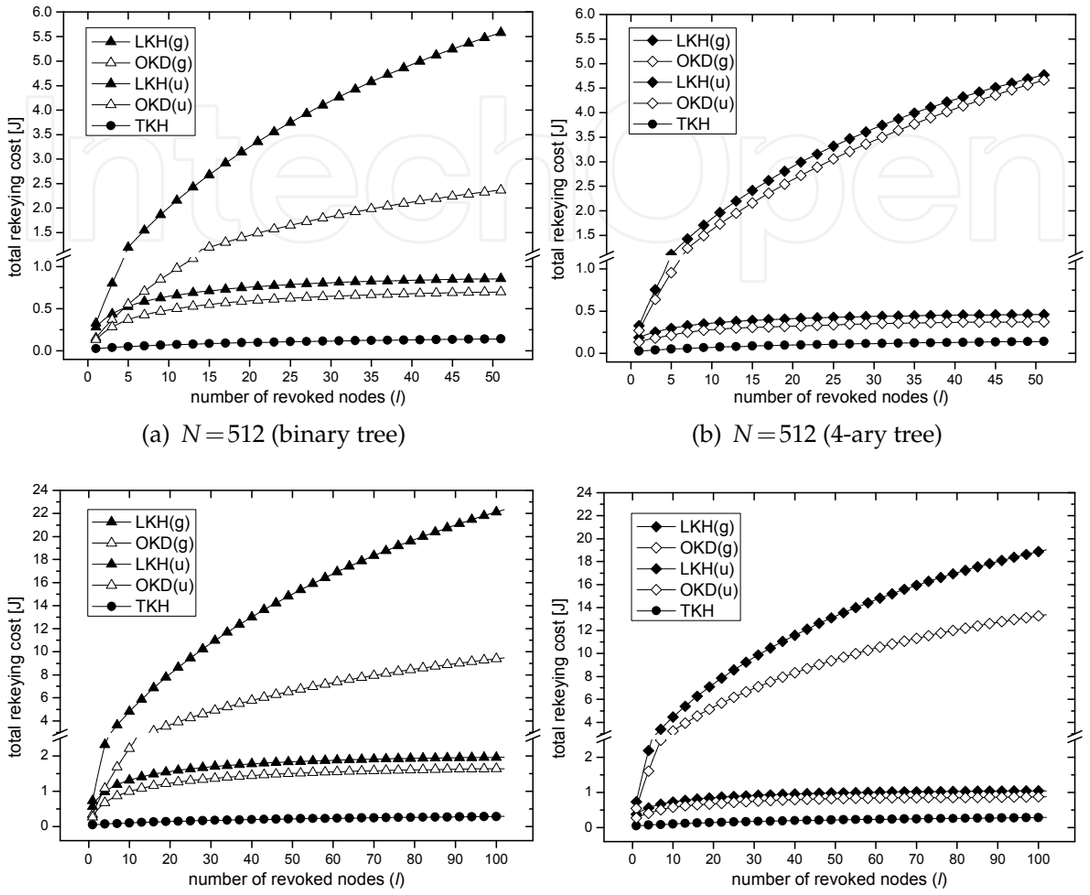
Second, from the network graph generated by using the previous method, we now transform it into a sink-based multicast source tree which actually delivers the central sink node's multicast messages on it. Among the many source tree generation algorithms (Diot et al., 1997), we use the simple and well-known algorithm: DSA (Dijkstra's Shortest path Algorithm) heuristic. If we overlap all the shortest paths from a source (s) to every nodes obtained from DSA (Cormen et al., 2001), we can build a multicast source tree starting from the central sink. However, our TKH can apply to any multicast source tree structures. We depict the multicast source tree in Figure 14.(b) which is generated from the Figure 14.(a) by using the DSA heuristic.

6.1 Simulation Results

In our simulations, we assume the network area of 1000×1000 size. For $N = 512, 1024$, we randomly placed sensor nodes with the communication range $(r = 82.1, 59.9)$ obtained by setting the connectivity (P_c) as 0.99. We set the unit communication costs and the unit rekeying message size same as the analysis settings. For LKH and OKD, binary and 4-ary key trees are generated where each sensor node is randomly assigned in the key trees. TKH's key trees are automatically generated from the generated sensor network multicast topology. After revoking randomly chosen node from a network, we calculated total rekeying costs of the three schemes which occurred during the update of the group key of the remaining nodes. We obtain the total rekeying costs by averaging 1000 independent simulation results for each number of N.

Figure 15 depicts simulation results of TRC according to the increased number of revoked nodes (l). We plot the graphs until 10% of nodes are revoked from $N = 512, 1024$. By comparing Figure 15 with 11, simulation results also possesses similar trends with the analysis results. We also verify that the previous principles in terms of the total rekeying costs obtained in analysis results are still hold in Figure 15. This confirms that our TKH always incurs lower rekeying costs compared to the logical key tree schemes. On average, TKH only

requires 18.6%, 33.7%, 15.2%, 27.9% of TRC compared to the most efficient logical key tree scheme (OKD(u) with 4-ary) in Figure 15.(a),(b),(c),(d) respectively.



(a) $N = 512$ (binary tree) (b) $N = 512$ (4-ary tree)

(c) $N = 1024$ (binary tree) (d) $N = 1024$ (4-ary tree)

Fig. 15. Simulation results of totla rekeying costs according to increasing number of revoked nodes (l) when $N = 512, 1024$.

Many researchers have proposed methods to construct an efficient multicast tree topology for a multi-hop wireless network (Park & Sahni, 2005) (Wieselthier et al., 2002). These schemes explicitly considers the wireless multicast advantage during multicast tree generation. For example, by applying *the sweep operation* (Wieselthier et al., 2002) after the DSA heuristic will modifies the multicast tree to adopt more sibling nodes in each sibling set. This kind of wireless-optimized topology will further reduce the total rekeying cost of TKH.

7. Conclusions

In this chapter, we proposed energy efficient group key management scheme for a wireless sensor network. By explicitly considering the topological information during a key tree generation, we showed that the Topological Key Hierarchy could greatly reduce the total rekeying costs compared to the previous logical key tree-based schemes. After description of our key

tree design principles, we proved performance improvements based on our detailed analysis results. We further compared rekeying costs in realistic simulation environments. TKH only requires about 10 to 30 percentages of rekeying costs compared to the best logical key tree scheme (OKD(u) with 4-ary) in the network of 1024 sensors. We conclude that our TKH can scale to large-scale sensor networks providing small rekeying cost for group key management.

8. References

Akyildiz, I. F., Weilian Su, Y. S. & Cayirci, E. (2002). A survey on sensor networks, *IEEE Communications Magazine* **40**(8): 102–114.

Bellare, M., Canetti, R. & Krawczyk, H. (1997). HMAC: Keyed-hashing for message authentication. IETF RFC 2104.

Blundo, C., Santis, A. D., Herzberg, A., Kutten, S., Vaccaro, U. & Yung, M. (1992). Perfectly-secure key distribution for dynamic conferences, *Advances in Cryptology—CRYPTO '92*, pp. 471–486.

Canetti, R., Garay, J., Itkis, G., Micciancio, D., Naor, M. & Pinkas, B. (1999). Multicast security: A taxonomy and some efficient constructions, *In Proceedings of the 18th IEEE INFOCOM*.

Chadha, A., Liu, Y. & Das, S. K. (2006). Group key distribution via local collaboration in wireless sensor networks, *IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON)*.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2001). *Introduction to Algorithms*, The MIT Press.

Diffie, W. & Hellman, M. E. (1976). New directions in cryptography, *IEEE Transactions on Information Theory* **22**(5): 644–654.

Diot, C., Dabbous, W. & Crowcroft, J. (1997). Multipoint communication: A survey of protocols, functions, and mechanisms, *IEEE Journal on Selected Areas in Communications* **15**(3): 277–290.

Horng, G. (2002). Cryptanalysis of a key management scheme for secure multicast communications, *IEICE: IEICE Transactions on Communications/Electronics/Information and Systems* **E85-B**(5): 1050–1051.

Intanagonwiwat, C., Govindan, R. & Estrin, D. (2000). Directed diffusion: a scalable and robust communication paradigm for sensor networks, *6th Annual ACM Internation Conference on Mobile Computing and Networking (MobiCom '00)*, pp. 56–67.

Kim, Y., Perrig, A. & Tsudik, G. (2000). Simple and fault-tolerant key agreement for dynamic collaborative groups, *7th ACM Conference on Computer and Communications Security (CCS)*.

Ku, W.-C. & Chen, S.-M. (2003). An improved key management scheme for large dynamic groups using one-way function trees, *International Conference on Parallel Processing Workshops*.

Li, X. S., Yang, Y. R., Gouda, M. G. & Lam, S. S. (2001). Batch rekeying for secure group communications, *WWW10*.

Lin, J.-C., Lai, F. & Lee, H.-C. (2005). Efficient group key management protocol with one-way key derivation, *IEEE Conference on Local Computer Networks (LCN)*, pp. 336–343.

Panja, B., Madria, S. K. & K.Bhargava, B. (2006). Energy and communication efficient group key management protocol for hierarchical sensor networks, *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*.

Park, J. & Sahni, S. (2005). Maximum lifetime broadcasting in wireless networks, *IEEE Tran. on Computers* **54**: 1081–1090.

Penrose, M. (2003). *Random Geometric Graphs*, Oxford Studies in Probability, Oxford University Press.

Penrose, M. D. (1997). The longest edge of the random minimal spanning tree, *The Annals of Applied Probability* **7**(2): 340–361.

Pietro, R. D., Mancini, L. V., Law, Y. W., Etalle, S. & Havinga, P. (2003). Lkhw: A directed diffusion-based secure multicast scheme for wireless sensor networks, *Proceedings of the 2003 International Conference on Parallel Processing Workshops*.

Rafaeli, S. & Hutchison, D. (2003). A survey of key management for secure group communication, *ACM Computing Surveys* **35**(3): 303–329.

Rivest, R. L., Shamir, A. & Adleman, L. A. (1978). A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM* **21**(2): 120–126.

Salido, J., Lazos, L. & Poovendran, R. (2008). Energy and bandwidth-efficient key distribution in wireless ad-hoc networks: A cross-layer approach, *IEEE/ACM Transactions on Networking (to appear)* **15**(6): 1527–1540.

Sherman, A. T. & McGrew, D. A. (2003). Key establishment in large dynamic groups using one-way function trees, *IEEE Transactions on Software Engineering* **29**(5): 444–458.

Staddon, J., Miner, S., Franklin, M., Balfanz, D., Malkin, M. & Dean, D. (2002). Self-healing key distribution with revocation.

Steiner, M., Tsudik, G. & Waidner, M. (1996). Diffie-hellman key distribution extended to groups, *Third ACM Conference on Computer and Communications Security (CCS)*, pp. 31–37.

Sun, Y., Trappe, W. & Liu, K. J. R. (2004). A scalable multicast key management scheme for heterogeneous wireless networks, *IEEE/ACM Transactions on Networking* **12**(4): 653–666.

U.S. DoC NIST (2001). Advanced encryption standard. FIPS Publication 197.

ZigBee Alliance (2006). Zigbee specifications (version 1.0, r13).

Tucker, A. (1995). *Applied Combinatorics, Ch. 3*, John Wiley & Sons.

Waldvogel, M., Caronni, G., Sun, D., Weiler, N. & Plattner, B. (1999). The VersaKey framework: Versatile group key management, *IEEE Journal on Selected Areas in Communications* **17**(9): 1614–1631.

Wallner, D. M., Harder, E. J., & Agee, R. C. (1997). Key management for multicast: issues and architectures. IETF RFC 2627.

Wieselthier, J. E., Nguyen, G. D. & Ephremides, A. (2002). Energy-efficient broadcast and multicast trees in wireless networks, *Mobile Networks and Applications* **7**(6): 481–492.

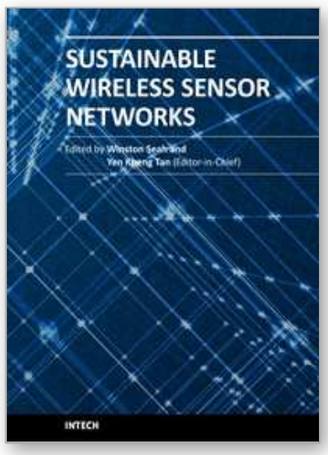
Wong, C. K., Gouda, M. G. & Lam, S. S. (1998). Secure group communications using key graphs, *ACM SIGCOMM*.

Zhang, W. & Cao, G. (2005). Group rekeying for filtering false data in sensor networks: a predistribution and local collaboration-based approach, *IEEE INFOCOM*, pp. 503–514.

Zhao, F. & Guibas, L. J. (2004). *Wireless Sensor Networks: An Information Processing Approach*, Elsevier.

IntechOpen

IntechOpen



Sustainable Wireless Sensor Networks
Edited by Yen Kheng Tan

ISBN 978-953-307-297-5
Hard cover, 574 pages
Publisher InTech
Published online 14, December, 2010
Published in print edition December, 2010

Wireless Sensor Networks came into prominence around the start of this millennium motivated by the omnipresent scenario of small-sized sensors with limited power deployed in large numbers over an area to monitor different phenomenon. The sole motivation of a large portion of research efforts has been to maximize the lifetime of the network, where network lifetime is typically measured from the instant of deployment to the point when one of the nodes has expended its limited power source and becomes in-operational â€“ commonly referred as first node failure. Over the years, research has increasingly adopted ideas from wireless communications as well as embedded systems development in order to move this technology closer to realistic deployment scenarios. In such a rich research area as wireless sensor networks, it is difficult if not impossible to provide a comprehensive coverage of all relevant aspects. In this book, we hope to give the reader with a snapshot of some aspects of wireless sensor networks research that provides both a high level overview as well as detailed discussion on specific areas.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Ju-hyung Son, Seung-woo Seo and Seung-woo Seo (2010). Group Key Managements in Wireless Sensor Networks, Sustainable Wireless Sensor Networks, Yen Kheng Tan (Ed.), ISBN: 978-953-307-297-5, InTech, Available from: http://www.intechopen.com/books/sustainable-wireless-sensor-networks/group-key-managements-in-wireless-sensor-networks

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2010 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen