Available online at www.sciencedirect.com

ScienceDirect

Ad Hoc Networks 5 (2007) 14–23

Ad Hoc Networks

www.elsevier.com/locate/adhoc

Self-healing group key distribution with time-limited node revocation for wireless sensor networks

Yixin Jiang [a,*], Chuang Lin [a], Minghui Shi [b], Xuemin (Sherman) Shen [b]

[a] *Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China*
[b] *Department of Electrical and Computer Engineering, University of Waterloo, Canada*

Available online 23 June 2006

Abstract

A novel key distribution scheme with time-limited node revocation is proposed for secure group communications in wireless sensor networks. The proposed scheme offers two important security properties: the seal-healing re-keying message distribution which features periodic one-way re-keying with implicitly authentication, efficient tolerance for the lost re-keying messages, and seamless Traffic Encryption Key (*TEK*) switch without disrupting ongoing data transmissions; and the time-limited dynamic node attachment and detachment, so that both forward and backward secrecy is assured by dual directional hash chains. It is shown that the communication and computation overhead of the proposed protocol is light, and the protocol is robust under poor communication channel quality and frequent group node topology change.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Self-healing; Node revocation; Key distribution; Group communication

1. Introduction

Applications of wireless sensor networks (WSNs) have attracted great attention from both academia and industry recently. Secure group communication is increasingly used as an efficient communication method for group-oriented applications in WSNs. Communications within each sensor node group should not be eavesdropped by other groups or malicious nodes. Given the open nature of broadcast channel, the combination of group communication and WSNs is more susceptible to unauthorized access. Thus, it is required to provide the confidentiality in group communications so that non-legitimate nodes are prevented from having access to the secret content, whereas only legitimate nodes can decrypt the multicast data even if the data is broadcast to the entire network. Traffic Encryption Key (*TEK*), a symmetric key, is used to encrypt data by the source and decrypt them by the destination [1]. Moreover, considering the dynamic node topology due to nodes' attachment and detachment, it is necessary to refresh the *TEK* to prevent the detached node from accessing future communications and the newly attached node from accessing previous communications. Group Key Manager

* Corresponding author.
E-mail addresses: yxjiang@csnet1.cs.tsinghua.edu.cn (Y. Jiang), clin@csnet1.cs.tsinghua.edu.cn (C. Lin), mshi@bbcr.uwaterloo.ca (M. Shi), xshen@bbcr.uwaterloo.ca (X. (S.) Shen).

1570-8705/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2006.05.007

(GKM), which is located in sensor network controller, is responsible for distributing re-keying messages to the nodes in the group securely by encrypting them using the Key Encrypting Key (*KEK*) [2].

The *TEK* is updated by trigging a re-keying process after a node attaches to or detaches from an active group session. The process ensures that a new node cannot decrypt previous multicast data, and prevents a detached node from eavesdropping future multicast data. Since each node topology change triggers a new re-keying process, *TEK* renew messages may impact the performance and scalability when node topology frequently changes.

A number of approaches have been proposed to efficiently tackle the scalability problem of key distribution with high dynamic node topology. Some surveys are available in [3,4]. Considering the interdependency of re-keying messages for revocation, group key distribution scheme with revocation can be classified into two distinct classes: *stateless* or *stateful* scheme. In stateful scheme [5,6], a legal node's state in the current re-keying will affect its ability to decrypt future group keys. However, the group re-keying in a stateless scheme [6–9], relies only on current re-keying message and the node's initial configuration. A non-revoked node can decrypt the new *TEK* independently from the previous re-keying messages without contacting the GKM, even if the node is off-line for a while. The property makes stateless scheme more useful in scenarios where some nodes are not constantly on-line or suffer from burst packet losses.

In addition to re-keying and the node revocation, some recent works address the self-healing issue that a group node can recover the missed session keys from the latest re-keying message on its own. Based on two-dimension t-degree polynomials, a self-healing group key distribution scheme is first presented in [10] and improved in [11,12].

Typically, group key distribution protocols for WSNs should take both security and service quality into consideration. The basic security requirements include [3,4] (1) Group confidentiality: nodes that are not the part of the group should not have access to any key that can decrypt any data broadcast to the group. (2) Forward secrecy: a node that detaches from the group should not have access to any future keys so that it cannot decrypt future data. (3) Backward secrecy: a new node that attaches to the session should not have access to any old key so that it cannot decrypt previous data.

The impaired channel usually results in scheme failure if nodes cannot communicate with GKM. The dynamics of the node topology also increase service disruption probability, since some nodes may lose connections temporarily. Hence, it is required to offer a reliable re-keying process with sufficient small number and size of re-keying messages. In addition, the re-keying scheme must not require either a large number of storage keys or high computation overhead at GKM or the nodes in the group.

In this paper, we propose an efficient self-healing group key scheme with time-limited node revocation based on dual directional hash chains (DDHC), which assures forward and backward secrecy, in high packet loss environment. Comparing with existing literature, the scheme offers a practical seal-healing method and an implicit node revocation algorithm with lightweight computation and communication overhead, favouring the group application scenarios in WSNs which employ dynamic node topology and broadcast channel with high packet loss or error rate. The *TEK* is re-keyed periodically instead of on every node topology change. Periodic or batch re-keying can remarkably reduce both the computation and communication overhead at the GKM and the nodes, and thus improve the scalability and performance of key distribution protocols. The proposed scheme also offers seamless *TEK* refreshment without disrupting the ongoing data transmission. The performance of the proposed scheme under poor broadcast channel condition is evaluated by both analysis and numerical results. It is shown that the proposed scheme can tolerate high channel loss rate, and hence make a good balance between performance and security, which is suitable for WSN applications.

The rest of the paper is organized as follows. In Section 2, the basic principle of time-limited group node revocation and self-healing method are introduced. In Section 3, the proposed self-healing group key distribution protocol for wireless sensor networks is described in detail. In Section 4, the security and the performance analysis are presented, respectively, followed by the conclusion in Section 5.

2. Self-healing and time-limited node revocation

2.1. One-way hash chain

We first introduce the concept of one-way hash function, which is the foundation of dual directional hash chain (DDHC). A hash function takes a binary string of arbitrary length as input, and outputs a binary string of fixed length. A one-way function H satisfies the following two properties: (1) given x, it is easy to compute y such that$y = H(x)$; (2) given y, it is computationally infeasible to compute$x$such that$y = H(x)$.

A one-way hash chain is a sequence of hash values $\{x_n, \ldots, x_j, \ldots, x_0\}$ such that $\{x_j | \forall j$: $0 < j \leqslant n$, $x_{j-1} = H(x_j)\}$. The random x_n is a secret seed of the one-way hash chain. Thus, there exists the following relation: $x_1 = H(x_2) = \cdots = H^{n-2}(x_{n-1}) = H^{n-1}(x_n)$.

Due to the one-way property of hash function H, given x_i, it is computationally infeasible to calculate x_j $(j < i)$, but is easy to compute any x_j $(j > i)$ with $x_j = H^{j-i}(x_i)$.

2.2. Dual directional hash chains

A dual directional hash chain (DDHC) is composed of two one-way hash chains with equal length, a forward hash chain K^{F} and a backward hash chain K^{B}. It can be derived as follows: (1) generating two random key seed values, K_0^{F} and K_0^{B}, for forward and backward hash chains with the size $z + 1$, respectively; (2) repeatedly applying the same one-way function on each seed to produce two hash chains of equal length $z + 1$. The dual hash sequences are generated as $\{K_0^{\mathrm{F}}, H(K_0^{\mathrm{F}}), \ldots, H^i(K_0^{\mathrm{F}}), \ldots, H^z(K_0^{\mathrm{F}})\}$ and $\{K_0^{\mathrm{B}}, H(K_0^{\mathrm{B}}), \ldots, H^i(K_0^{\mathrm{B}}), \ldots, H^z(K_0^{\mathrm{B}})\}$.

2.3. Time-limited node revocation scheme

The concept of node revocation can be described as follows. Let N be the set of all possible group nodes, and R the set of revoked nodes, where $R \subseteq N$. The group node revocation is required to offer a secure way for GKM to transmit re-keying messages over a broadcast channel shared by all nodes so that any node $U_i \in \{N \backslash R\}$ can decrypt the re-keying message, whereas the nodes in R cannot decrypt re-keying message.

The proposed time-limited group node revocation scheme is implemented by DDHC. Let z denote the lifecycle of the group communication system. Then the maximum number of time periods is $z + 1$. Without loss of generality, assume that z is an integer, i.e., the system starts at time 0 and ends at time z. We argue that this constrain on the maximum number of time periods should not be considered as a limitation in a group communication.

During the system lifecycle, when a node attaches to an active group, the GKM assigns the pair $\left(H^{t_1}(K_0^{\mathrm{F}}), H^{z-t_2}(K_0^{\mathrm{B}})\right)$ to the new node according to its prearranged lifecycle (t_1, t_2). Due to the DDHC, once a node's lifecycle is expired, it is forced to detach from the group session without requiring the direct intervention of the GKM.

The application of the DDHC in time-limited node revocation mechanism is shown in Fig. 1. The range of hash keys hidden from the group node with the lifecycle (t_1, t_2) are presented with grey background. The forward hash chain guarantees the backward secrecy. A new node that participates in the group communication at time t_1 cannot calculate the previous hash keys $\{K_0^{\mathrm{F}}, H(K_0^{\mathrm{F}}), \ldots, H^{t_1-1}(K_0^{\mathrm{F}})\}$ before t_1 because of the property of one-way hash function. Similarly, the backward hash chain guarantees the forward secrecy. Once a node detaches from the group session at time t_2, it cannot compute the subsequent hash keys $\{H^{z-t_2-1}(K_0^{\mathrm{B}}), H^{z-t_2-2}(K_0^{\mathrm{B}}), \ldots, K_0^{\mathrm{B}}\}$ after t_2.

For a group system with lifecycle $(0, z)$, the traffic encryption key (TEK) at time x is defined as a function of x, $H^x(K_0^{\mathrm{F}})$, $H^{z-x}(K_0^{\mathrm{B}})$, and RK_i, respectively,

$$TEK_i = f\left(H^x(K_0^{\mathrm{F}}), H^{z-x}(K_0^{\mathrm{B}}), RK_i\right), \tag{1}$$

where $0 \leqslant x \leqslant z$, and RK_i denotes the ith re-keying message from GKM, which will be described later.

Due to the one-way property of the DDHC, a group node with $\left(H^{t_1}(K_0^{\mathrm{F}}), H^{z-t_2}(K_0^{\mathrm{B}})\right)$ is restricted

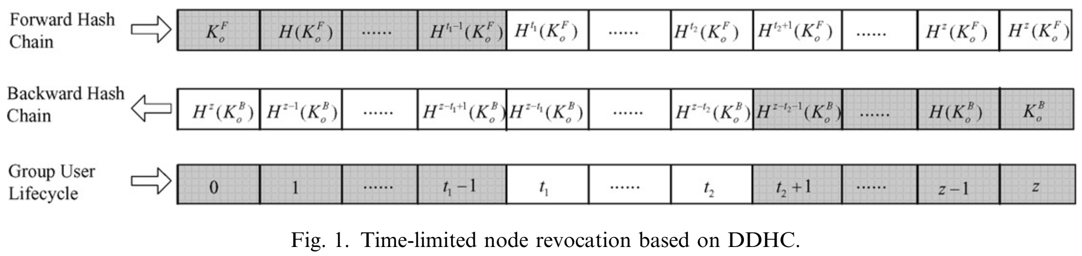

Fig. 1. Time-limited node revocation based on DDHC.

in accessing the *TEK*s in the time-range of (t_1, t_2), since the group node can only use the pre-assigned seeds $\left(H^{t_1}(K_0^{\mathrm{F}}), H^{z-t_2}(K_0^{\mathrm{B}})\right)$ to compute $H^x(K_0^{\mathrm{F}})$ and $H^{z-x}(K_0^{\mathrm{B}})$ as $H^x(K_0^{\mathrm{F}}) = H^{x-t_1}\left(H^{t_1}(K_0^{\mathrm{F}})\right)$ and $H^{z-x}(K_0^{\mathrm{B}}) = H^{t_2-x}\left(H^{z-t_2}(K_0^{\mathrm{B}})\right)$, respectively. However, out of the time-range (t_1, t_2), it cannot compute $H^x(K_0^{\mathrm{F}})$ and $H^{z-x}(K_0^{\mathrm{B}})$ for $x \leqslant t_1$ or $x \geqslant t_2$. Furthermore, it cannot calculate the *TEK* at time x either. Thus, an implicit time-limited node revocation is implemented. Each group node can only be given access to a pre-defined contiguous range of the *TEK* according to its lifecycle (t_1, t_2).

2.4. Self-healing re-keying mechanism

Self-healing re-keying mechanism offers equivalent reliable *RK* transmission over impaired broadcast channel. As described above, each legal node can derive the *TEK* at time $x (0 \leqslant x \leqslant z)$ by $TEK_i = f\left(H^x(K_0^{\mathrm{F}}), H^{z-x}(K_0^{\mathrm{B}}), RK_i\right)$, where $H^x(K_0^{\mathrm{F}})$ and $H^{z-x}(K_0^{\mathrm{B}})$ are not required to be transmitted at each re-keying. Each node can individually compute them according to the pre-assigned seeds $\left(H^{t_1}(K_0^{\mathrm{F}}), H^{z-t_2}(K_0^{\mathrm{B}})\right)$ and current time x. RK_i is encapsulated in the re-keying message, which is periodically sent by the GKM to all nodes.

At the initial phase, the GKM first selects a secret seed RK_n, and then uses the seed to pre-compute a one-way hash chains $\{RK_i | i = 1, 2, \ldots, n\}$, where n is reasonably large. Specifically, the GKM chooses RK_n as the last key in the hash chain and repeatedly performs the hash function H to compute all the rest of keys as $RK_i = H(RK_{i+1})$, $0 \leqslant i \leqslant n-1$. All *RK*s satisfies $RK_0 = H(RK_1) = \cdots = H^{n-2}(RK_{n-1}) = H^{n-1}(RK_n)$.

In the subsequent re-keying phases, all RK_i, $i = 1, 2, \ldots, n$, will be released to all nodes by the GKM in reverse order, i.e., RK_0 will be released for session 0, RK_1 for session 1,..., and RK_n for session n, and so on. Given current RK_j in the hash chain, nodes can only use one-way function H to compute the previous keys $\{RK_i | 0 \leqslant i \leqslant j\}$ recursively, however they cannot compute other keys $\{RK_i | j+1 \leqslant i \leqslant n\}$.

Consider that each re-keying message contains only one *RK* in current session. Then, though the re-keying messages may be lost during the transmission, self-healing can be achieved since the lost *RK*s in previous re-keying messages can be recovered by using the one-way hash function and the last received *RK*. Furthermore, the *TEK* will be successfully derived by each node. Thus, the proposed self-healing scheme can efficiently tolerate high packet loss or error rate.

3. Proposed group key distribution scheme

We propose a lightweight and robust group key distribution scheme with time-limited node revocation to protect the group communications in WSNs. As shown in Fig. 2, the scheme consists of the two components: self-healing and *TEK* switching.

The self-healing mechanism provides a robust way for tolerating the packet loss in impaired broadcast channel. On receiving a *RefreshKey* message, each node implicitly verifies the authenticity of the received *RK* by using pre-stored *RK*s. If necessary, the node recovers the lost *RK*s using the new *RK* without requesting GKM to re-transmit them. The proposed self-healing mechanism relies on the one-way property of a hash function. Similar mechanism is also used in LiSP [13] and μTESLA [14,15]. The proposed mechanism improves (1) efficiency in that each node only buffers the constant number of keys, whereas TESLA is required to buffer all the received messages until the node receives an authentic message; and (2) implicit time-limited node revocation while it is not considered in LiSP.

The *TEK* Switching module seamlessly switches the *TEK* without disrupting ongoing data transmissions. Two key-slots are set up for each node, which can be operated concurrently. When the *RK* in one key-slot is used for data encryption or decryption, the received new *RK* will be written into the other key-slot. At the middle point of the key update interval, the node switches the active key-slot to the one with the new *RK*.

There are three types of signal messages used in the proposed key distribution scheme: *InitGroupKey*, *RequestKey*, and *RefreshKey*, respectively. *InitGroupKey* and *RequestKey* are unicasted between GKM and a group node, while *RefreshKey* is broadcasted to all group nodes. The *InitGroupKey* message is sent to nodes to initiate re-keying parameters at the initial phase. The *RefreshKey*

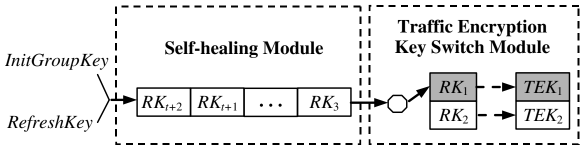

Fig. 2. Self-healing group key distribution with time-limited node revocation.

message periodically broadcasts the next *RK* in the key sequence to nodes. The *RequestKey* message is used by each node to explicitly request the current *RK* in the key sequence. The *RequestKey* message will be generated by a node when it fails to receive *RK* over *t* renewal intervals.

3.1. Initial configuration setup

Fig. 3 illustrates how to initialize and refresh the *TEK*. At the initial phase, the GKM pre-computes a one-way *RK* sequence $\{RK_i | i = 1, 2, \ldots, n\}$ such that $RK_i = H(RK_{i+1})$, $0 \leqslant i \leqslant n-1$. Each node U_i keeps a main Key Encryption Key, KEK_i, for *InitGroupKey* message encryption and its authentication. This secret is shared only between GKM and U_i via entity authentication. At time t_{init}, the GKM sends the following message to U_i.

$$\text{GKM} \rightarrow U_i : \{E_{KEK_i}(t \| RK_{t+2} \| T_{\text{refresh}} \| H^{t_1}(K_0^{\text{F}}) \times \| H^{t_2}(K_0^{\text{B}})) \| \text{MAC}(t \| RK_{t+2} \| T_{\text{refresh}} \| H^{t_1}(K_0^{\text{F}}) \| H^{t_2}(K_0^{\text{B}}))\},$$

where t is the length of *RK* buffer; T_{refresh} is the re-keying period; $MAC(\cdot)$ generates a message digest code using a hash function, e.g., MD5; $H^{t_1}(K_0^{\text{F}})$ and $H^{z-t_2}(K_0^{\text{B}})$ are the corresponding seeds in the DDHC for node U_i with lifecycle (t_1, t_2).

Algorithm 1. Initial group communication system

```
01: Function Init_TEK (){
02:   if (Receiving InitGroupKey message){
03:     Decrypt InitGroupKey to get
          {$RK_{t+2}$, $t$, $T_{\text{refresh}}$};
04:     Allocate a key buffer with length $t$
          ($kb[1], \ldots, kb[t]$)
          and two key-slots ($ks[1], ks[2]$);
05:     for ($i = 1$; $i <= t - 1$; $i$++) do
          $kb[i] = H^{t-i}(RK_{t+s})$;
06:     $ks[2] = H^{t}(RK_{t+s})$; $ks[1] = H^{t+1}(RK_{t+s})$;
07:     $RK_w = H^{t+2}(RK_{t+s})$;
08:     Set key $ks[1]$ as the data encryption key;
09:     Set RefreshKeyTimer to $T_{\text{refresh}}$ /2;}}
```

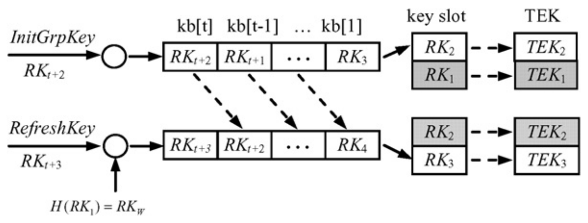

Fig. 3. Initial setup and *TEK* refreshment.

Algorithm 2. Refresh key timer

```
01: Function Refresh_Key_Timer(){
02:   if (RefreshKeyTimer triggered){
03:     Right-shift the key buffer and key-slot;
04:     $e$++; $RK_w$ = {the inactive key in the
              key slots};
05:   }
06:   Set active RKs in key slots;
07:   Set RefreshKeyTimer to $T_{\text{refresh}}$;
08:   if ($e == t$) send RequestKey message to
        GKM;}
```

When node U_i receives the *InitGroupKey* message, it processes this message according to Algorithm 1. Fig. 3 also illustrates how the node copies *RK* sequence into its key buffer and key-slots, computes *TEK*, and seamlessly switches the active *TEK* after obtaining RK_{t+2}.

If the timer is expired, each node will trigger the execution of Algorithm 2, which performs the right-shift operations so that *RK* can be automatically renewed at each interval T_{refresh}. Here, each node maintains two variables e and RK_w. SentryRK_w tracks the most recent futile *RK*, and e tracks the number of *RK* that the node fails to receive.

3.2. TEK refreshment: re-keying with implicit authentication

After the initial phase, the GKM periodically discloses the next *RK* in the pre-computed *RK* sequence to all nodes. Assume that the initial phase ends at time t_{init}. For the ith re-keying, the GKM broadcasts *RefreshKey* message with RK_{i+t+2}, $i = 0, \ldots, n-t-2$ to all nodes at time $t_{\text{init}} + i \cdot T_{\text{refresh}}$: $\text{GKM} \rightarrow U_j\ (j = 1, \ldots, m) : \{E_{TEK_{i+1}}(RK_{i+t+2}, RK_{i+1})\}$, where TEK_{i+1} is the active *TEK* at the time when *RefreshKey* message is broadcasted.

Algorithm 3. TEK refreshment and RK recovery

```
01: Function TEK_Refresh_Recover (){
02:   while (RefreshKey message received){
03:     Decrypt it with $TEK_{i+1}$ to get
          {$RK_{i+t+2}$, $RK_{i+1}$};
04:     if ($H(RK_{i+1}) \neq RK_w$){
05:       Discard the rekey message
            {$E_{TEK_{i+1}}(RK_{i+t+2}, RK_{i+1})$};
06:       continue;
07:     }
08:     $RK_w$ = the inactive key in key slots};
09:     Right-shift $kb[1] = RK_{i+3}$ to the inactive
        key slot;
10:     $TEK_{i+3} = f(H^t(K_0^F), H^{z-t}(K_0^B)RK_{i+3})$;
11:     for ($i = 1$; $i <= t - 1$; $i$++ ) do
        $kb[i] \rightarrow kb[i-1]$;
12:     if ($e \neq 0$){/* there are lost RKs */
13:       for($j = 0$; $j <= e$; $j$++ ) do /* recover
          lost RKs */
14:         $H^j(RK_{i+t+2}) \rightarrow kb[t-j]$;
15:       $e = 0$;/* reset value $e$ */}}}
```

On receiving the *RefreshKey* message, each node processes the message following Algorithm 3. The *TEK* can be synchronously renewed with the new re-keying message (line 10 in Algorithm 3). Due to the one-way property of *RK* sequence, the *RefreshKey* message does not need message authentication code, since the receiver can verify if the received *RK* belongs to the same key sequences by checking $H(RK_{i+1}) \neq RK_w$. Such implicit authentication notably decreases the message size.

The computation overhead in re-keying is not heavy, since it only needs to handle low-cost hash operations. The communication overhead is also lightweight, since the proposed key distribution scheme provides an implicit authentication for re-keying messages without message retransmissions.

3.3. Recovery of lost TEK

The *RefreshKey* message provides a self-healing mechanism to recover the lost *RK*s. Suppose that there are $r(\leqslant t)$ *RK*s reserved in the key buffer due to the previous lost messages, so there are $e = t - r$ empty slots in the key buffer. Let $\{RK'_r, \ldots, RK'_1\}$ denote these r *RK*s in the key buffer $\{kb[r], \ldots, kb[1]\}$, respectively. They also belong to the same *RK* sequence, and satisfy $H(RK'_r) = RK'_{r-1}, \ldots, H(RK'_2) = RK'_1$.

Algorithm 3 also shows the details of the self-healing mechanism (line 12–15). Upon getting a *RefreshKey* message, each node checks if $H(RK_{i+1}) = RK_w$ and uses RK_{i+t+2} to recover the lost *RK*s in the same key sequence if it holds. Fig. 4 illustrates lost *RK*s recovery. Assume that a node receives a *RefreshKey* with RK_{t+2}. Due to $H(RK_{t+2}) = RK_{t+1}$ and $e = 0$, there are no message loss. If the re-keying messages are lost in the next two intervals, $e = 2$ and there are $t - 2$ *RK*s in the key-buffer. Afterwards, the node receives an authentic *RefreshKey* message with RK_{t+5}. Since $H(RK_3^*) = RK_w$, it can recover previous two lost *RK*s as $RK_{t+3} = H^2(RK_{t+5})$ and $RK_{t+2} = H^3(RK_{t+5})$.

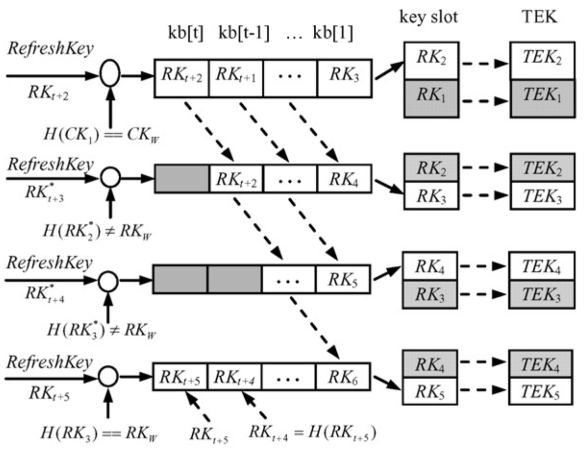

Fig. 4. Self-healing scheme: recovering the lost *TEK*.

3.4. Dynamic participation mechanism

The dynamic participation mechanism, as a basic security requirement, allows that any node can attach to or detach from the active group while assuring the freshness of the *TEK*.

Node attachment: When a node U_{Join} attaches to an active group, the corresponding actions (steps) are described as follows:

1. *Step 1:* Node U_{Join} first obtains the permission to attach to the group communication from the GKM. If it is successful, U_{Join} can establish a common secret KEK_i shared with the GKM. The GKM then sends the current system configuration to U_{Join} via an *InitGroupKey* message.

$$\text{GKM} \rightarrow U_{\text{Join}} : \{E_{KEK_{\text{Join}}}(t\|RK_k\|T_{\text{refresh}}\|H^{t_1}(K_0^F) \times\|H^{z-t_2}(K_0^B))\|\text{MAC}(t\|RK_k\|T_{\text{refresh}}\|H^{t_1}(K_0^F) \times\|H^{z-t_2}(K_0^B))\},$$

   where $H^{t_1}(K_0^F)$ and $H^{z-t_2}(K_0^B)$ are the corresponding seeds in the DDHC for U_{Join} with lifecycle (t_1, t_2).
2. *Step 2:* On receiving *InitGroupKey* message, U_{Join} generates the message following Algorithm 1, and then attaches to the active group communication. It can receive the subsequent re-keying messages and renew the *TEK* synchronously, as shown in Algorithm 3.

Node revocation: Assume that a node T_{Quit} with lifecycle (t_1, t_2) detaches from the session at time t_2. It cannot derive the *TEK* by $TEK_i = f\big(H^x(K_0^{\mathrm{F}}),$ $\mathrm{x}H^{z-x}(K_0^{\mathrm{B}}), RK_i\big)$ for $x \leqslant t_1$ or $x \geqslant t_2$. Thus, a time-limited node revocation is achieved implicitly without the need of intervention from the GKM so that the communication and the computation overhead on the GKM and group nodes are remarkably reduced.

3.5. Re-initialization mechanism

The GKM re-initializes the group communication system, if (1) all *nRK*s in the *RK* sequence have been used up; and (2) a node has explicitly requested *RK*, since more than *t RK*s are lost. In the former scenario, all nodes are forced to be re-initialized. GKM re-computes a new *RK* sequence $\{RK'_i | i = 1, 2, \ldots, n\}$, and then broadcasts a new *InitGroupKey* message with RK'_{t+s} to all the nodes. In the latter scenario, the GKM only sends the requesting node the *InitGroupKey* message with current configuration. The node can then periodically renew the *TEK* by the received *RefreshKey* message.

4. Performance analysis

We analyze the proposed scheme to verify that it satisfies the security and performance requirements for secure group communication described in Section 1.

4.1. Security analysis

The proposed scheme meets the security requirement for forward and backward secrecy, since it can assure the refreshment of the *TEK* by periodic or dynamic re-keying mechanism, when a node attaches to or detaches from an active group session, and when the two cases discussed in 3.5 occur.

The time-limited node revocation algorithm offers an efficient way to assure forward and backward secrecy. Assume that a node U_i with lifecycle (t_1, t_2) attaches to the session at time t_1 and detaches from the session at time t_2. During the lifecycle (t_1, t_2), U_i can use its pre-assigned seeds $\big(H^{t_1}(K_0^{\mathrm{F}}), H^{z-t_2}(K_0^{\mathrm{B}})\big)$ to compute the respective hash values $H^x(K_0^{\mathrm{F}})$ and $H^{z-x}(K_0^{\mathrm{B}})$ and derive the *TEK* at time x as

$$TEK_j = f\big(H^x(K_0^{\mathrm{F}}), H^{z-x}(K_0^{\mathrm{B}}), RK_j\big).$$

However, when $x \leqslant t_1$ or $x \geqslant t_2$, U_i cannot derive the corresponding *TEK*, since it can neither calculate $H^x(K_0^{\mathrm{F}})$ before it joins the group session $(x < t_1)$, nor $H^{z-x}(K_0^{\mathrm{B}})$ after it leaves the group session $(x > t_2)$ due to the one-way property of DDHC. Therefore, it is only restricted in access the *TEK*s in the time-range of (t_1, t_2). The forward and backward secrecy is assured with the time-limited node revocation.

4.2. Steady Markov state distribution

To evaluate the performance of *RK* renewal, we quantify the cost of the communication and computation overheads when nodes renew *TEK*.

As shown in Fig. 5, we first apply the Markov chain to derive the steady-state distributions of key-buffer states for a node. Then we investigate the communication and computation overhead. We assume each occurrence of *RK* loss is random and mutually independent. Let state S_i denote that there are i lost *RK* packets, and thus there are i empty slots in the key-buffer. The state transition is triggered by two events: packet lost, or receiving a re-keying *RK* packet successfully. Let $p_s = Pr\{RK$ message is received$\}$. Without loss of generality, we also assume that each transmitted message via the channel has the same loss probability or error rate, i.e., $p_l = Pr\{RK$ Message is lost$\}$. The assumption is reasonable since the channel does not distinguish the different packets.

Let $P(k)$ denote the steady-state probability of state S_i that there are exactly k empty slots. Since $\sum_{k=0}^{t} P(k) = 1$, according to the global balance equations, we have

$$\begin{cases} P(i) \cdot (p_s + p_l) = P(i-1) \cdot p_l, & i = 1, \ldots, t \\ P(0) \cdot p_l = P(t) + \sum_{i=1}^{t-1} P(i) \cdot p_s \end{cases}. \tag{2}$$

Hence, the steady-state distributions $P(k)$ are given as

$$\begin{cases} P(0) = (1 - p_l)/\big(1 - p_l^{t+1}\big) \\ P(i) = P(0) \cdot p_l^k, & k = 1, 2, \ldots, t \end{cases}. \tag{3}$$

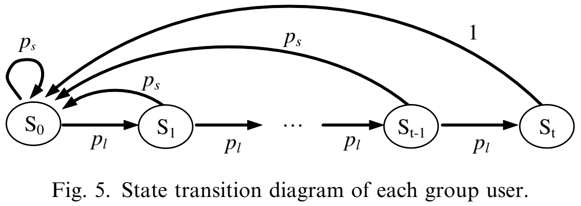

Fig. 5. State transition diagram of each group user.

4.3. Communication overhead

To evaluate the communication overhead between the GKM and nodes, we normalize the expected communication overhead C_{comm} by the cost of sending the *RefreshKey* messages. Let C_{init} and C_{refresh} denote the communication cost for transmitting *InitGroupKey* and *RefreshKey* message, respectively. Let $\alpha = C_{\text{init}}/C_{\text{refresh}}$ be the ratio of communication cost of *InitGroupKey* to that of *RefreshKey*. $\alpha > 1$ since the *InitGroupKey* message needs more bandwidth or resources than *RefreshKey* message. As discussed before, the GKM needs to transmit the *InitGroupKey* message when a node attaches to an active group session, or when all *nRK*shave been used, or when a node has explicitly requested *RK*. Note that in case 2, all nodes are required to be reinitialized, while in cases 1 and 3, only the specific node needs to be reinitialized by being sent *InitGroupKey* message. Besides these cases, GKM periodically broadcasts *RefreshKey* messages. Therefore the expected communication cost of a node is $E[C_{\text{comm}}] = C_{\text{init}} \cdot [1/n + P(t) + p_j] + C_{\text{refresh}} \cdot \sum_{k=0}^{t-1} P(k)$, where p_j denotes the probability when a node attaches to a group session. To analyze the dynamic of *RK* update, we assume that the frequency of attachment is low. According to (3), the communication cost C_{comm} can be normalized with C_{refresh} as:

$$C_{\text{comm}} = \alpha \cdot [1/n + p_l^t \cdot P(0)] + \sum_{k=0}^{t-1} p_l^k \cdot P(0). \tag{4}$$

If the value of C_{comm} is close to 1, it indicates that most *RefreshKey* messages should work well. By contrast, if C_{comm} is close to α, the protocol works less efficiently.

Fig. 6 shows the relationship between C_{comm} and the key buffer length t, where $n = 500$, $p_l = 0.1$–0.5, and $\alpha = 10$. The choice of α implies that the cost of transmitting and processing the *InitGroupKey* message is higher than that of transmitting the *RefreshKey* message. It can be seen that the key-buffer length t in each node determines the communication cost. A larger t can significantly reduce the communication overhead.

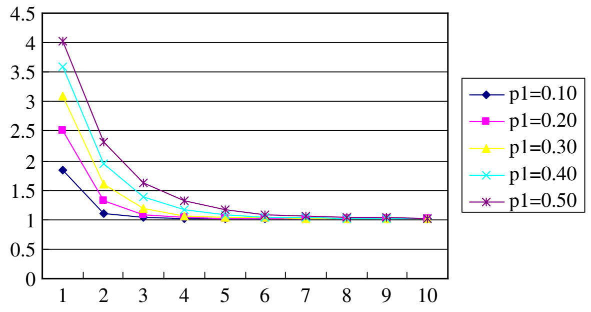

Fig. 6. Normalized communication costs C_{comm} vs. the key buffer length t ($n = 500$, $\alpha = 10$).

4.4. Computation overhead

Since the GKM is usually a high-performance server which is capable for heavy computation, we only focus on the computation complexity required in the nodes. The main computation overhead over nodes is the hash computations per *RefreshKey* message. Let N_h denote the number of hash computations per *RefreshKey* message. If there are k empty slots, the corresponding conditional expected value of N_h, $E[N_h|k \text{ slots}]$, can be derived as

$$E[N_h|k \text{ slots}] = \begin{cases} 0 \cdot p_l + (k+1) \cdot (1-p_l), & (k < t) \\ (t+1) \cdot p_l + (t+1) \cdot p_s, & (k = t) \end{cases} = \begin{cases} (k+1) \cdot (1-p_l), & (k < t), \\ t+1, & (k = t). \end{cases} \tag{5}$$

Then the expected value of N_h is

$$E[N_h] = \sum_{k=0}^{t} E[N_h|k \text{ slots}] \cdot P(k) = (t+1) \cdot P(0) \cdot p_l^t + \sum_{k=0}^{t-1} (k+1) \cdot (1-p_l) \cdot P(0) \cdot p_l^k. \tag{6}$$

Fig. 7 depicts the computation cost $C_{\text{comp}} = E[N_h]$ as a function of the key buffer length t, where p_l vary from 0.1 to 0.5 under the assumption of $n = 500$. Fig. 7 also indicates that the computation cost of each node is low, since each node only computes less than two hash functions per RK refreshment even in the worst case, e.g., $p_l = 0.5$.

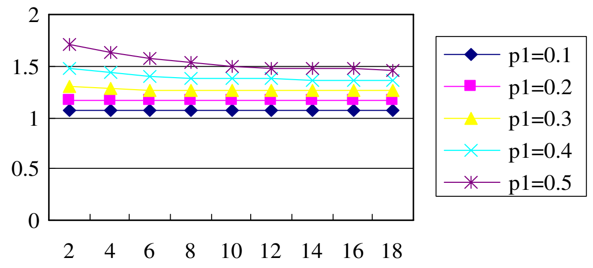

Fig. 7. Computation costs of node vs. key buffer length t ($n = 500$).

From Figs. 6 and 7, the desirable number of key buffer should satisfy $t \geqslant 10$ so that the normalized communication or computation cost is lower and in the range of 1–1.5, which indicates that the proposed scheme is efficient in terms of communication and computation overhead, even in high packet loss or error rate environment.

5. Conclusion

In this paper, we have developed a novel key distribution scheme for secure group communications in WSNs. The scheme can offer two important security properties: self-healing group key distribution, which features periodic re-keying with implicit authentication, efficient tolerance for the lost re-keying messages, and seamless *TEK* switching without disrupting ongoing data transmissions; and time-limited group node revocation, so that the forward and backward secrecy can be assured. The performance analysis indicates that proposed key distribution scheme is suitable for WSNs group compunctions applications.

Acknowledgements

This research has been supported in part by the NSFC under contracts No. 60573144, 60218003, 60429202, and 90412012 and Intel IXA University Research Plan.

References

[1] H. Harney, C. Muckenhirn, Group Key Management Protocol (GKMP) Specification, RFC 2093, 1997.
[2] H. Harney, C. Muckenhirn, Group Key Management Protocol (GKMP) Architecture, RFC 2094, 1997.
[3] S. Rafaeli, D. Hutchison, A survey of key management for secure group communication, ACM Computing Surveys 35 (3) (2003) 309–329.
[4] Y. Challal, H. Seba, Group key management protocols: a novel taxonomy, International Journal of Information Technology 2 (1) (2005) 105–119.
[5] D.M. Wallner, E.J. Harder, R.C. Agee, Key Management for Multicast: Issues and Architectures, RFC 2627, June 1999.
[6] C.K. Wong, M.G. Gouda, S.S. Lam, Secure group communications using key graphs, IEEE/ACM Transaction on Networking 8 (1) (2000) 16–30.
[7] D. Naor, M. Naor, et al., Revocation and tracing schemes for stateless receivers, Proc. of Advances in Cryptology (CRYPTO 2001), Springer-Verlag, LNCS 2139, 2001, pp. 41–62.
[8] A. Fiat, M. Naor, Broadcast encryption, in: Proceedings of Advances in Cryptology–CRYPTO'93, LNCS, vol. 773, 1994, pp. 480–491.
[9] D. Halevy, A. Shamir, The LSD broadcast encryption scheme, in: Proceedings of Advances in Cryptology–CRYPTO'02, LNCS, vol. 2442, 2002, pp. 47–60.
[10] J. Staddon, S. Miner, M. Franklin, Self-healing key distribution with revocation in: Proceedings of IEEE Symposium on Security and Privacy, 2002, pp. 241–257.
[11] D. Liu, P. Ning, K. Sun, Efficient self-healing group key distribution with revocation capability, in: Proceedings of the 10th ACM CCS, 2003, pp. 231–240.
[12] A. Blundo, P. D'Arco, A. De Santis, M. Listo, Design of self-healing key distribution schemes, Design, Codes, and Cryptography 32 (1–3) (2004) 15–44.
[13] T. Park, K.G. Shin, LiSP: a lightweight security protocol for wireless sensor networks, ACM Transactions on Embedded Computing Systems 3 (3) (2004) 634–660.
[14] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, SPINS: security protocols for sensor netowrks, in: Proceedings of IEEE/ACM MobiCom'01, 2001, pp. 189–199.
[15] D. Liu, P. Ning, Multilevel μTESLA: broadcast authentication for distributed sensor networks, ACM Transactions on Embedded Computing Systems 3 (4) (2004) 800–836.

Yixin Jiang is a Ph.D. candidate of the Department of Computer Science and Technology, Tsinghua University, China. He received the M.E degree in Computer Science from Huazhong University of Science and Technology in 2002. In 2005, he was a Visiting Scholar with the Department of Computer Sciences, Hong Kong Baptist University. His current research interests include security and performance evaluation in wireless communication and mobile computing. He has published more than 20 papers in research journals and IEEE conference proceedings in these areas.

Chuang Lin is a Professor and the head of the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He received the Ph.D. degree in Computer Science from Tsinghua University in 1994. In 1985–1986, he was a Visiting Scholar with the Department of Computer Sciences, Purdue University. In 1989–1990, he was a Visiting Research Fellow with the Department of Management Sciences and Information Systems, University of Texas at Austin. In 1995–1996, he visited the Department of Computer Science, Hong Kong University of Science and Technology. His current research interests include computer networks, performance evaluation, network security, logic reasoning, and Petri net and its applications. He has published more than 200 papers in research journals and IEEE conference proceedings in these areas and has published three books. Lin is an IEEE senior member and the Chinese Delegate in IFIP TC6. He serves as the General Chair, ACM SIGCOMM Asia workshop 2005; the Associate Editor, IEEE Transactions on Vehicular Technology; and the Area Editor, Journal of Parallel and Distributed Computing.

Minghui Shi received a B.S. degree in 1996 from Shanghai Jiao Tong University, China, and an M.S. degree in 2002 from the University of Waterloo, Ontario, Canada, both in electrical engineering. He is currently working toward a Ph.D. degree at the University of Waterloo. His current research interests include wireless LAN/cellular network integration and network security.

Xuemin (Sherman) Shen received the B.Sc. (1982) degree from Dalian Maritime University (China) and the M.Sc. (1987) and Ph.D. degrees (1990) from Rutgers University, New Jersey (USA), all in electrical engineering. From September 1990 to September 1993, he was first with the Howard University, Washington D.C., and then the University of Alberta, Edmonton (Canada). Since October 1993, he has been with the Department of Electrical and Computer Engineering, University of Waterloo, Canada, where he is a Professor and the Associate Chair for Graduate Studies. His research focuses on mobility and resource management in interconnected wireless/wireline networks, UWB wireless communications systems, wireless security, and ad hoc and sensor networks. He is a coauthor of two books, and has published more than 200 papers and book chapters in wireless communications and networks, control and filtering. He was the Technical Program Co-Chair for IEEE Globecom '03 Symposium on Next Generation Networks and Internet, ISPAN '04, IEEE Broadnet '05, QShine '05, and is the Special Track Chair of 2005 IFIP Networking Conference. He serves as the Associate Editor for IEEE Transactions on Wireless Communications; IEEE Transactions on Vehicular Technology; ACM/Wireless Networks; Computer Networks; Wireless Communications and Mobile Computing (Wiley); and International Journal of Computers and Applications. He also serves as Guest Editor for IEEE JSAC, IEEE Transactions Vehicular Technology, IEEE Wireless Communications, and IEEE Communications Magazine. He received the Premier's Research Excellence Award (PREA) from the Province of Ontario, Canada for demonstrated excellence of scientific and academic contributions in 2003, and the Distinguished Performance Award from the Faculty of Engineering, University of Waterloo, for outstanding contribution in teaching, scholarship and service in 2002. He is a registered Professional Engineer of Ontario, Canada.