Research Journal of Applied Sciences, Engineering and Technology 5(3): 1048-1053, 2013
DOI:10.19026/rjaset.5.5061
ISSN: 2040-7459; E-ISSN: 2040-7467

Submitted: June 29, 2012 Accepted: July 28, 2012 Published: January 21, 2013

Research Article

A Dynamic Self-Healing Key Management Scheme for Wireless Sensor Networks Based on EBS

[1]Jianting Ning and [2]Xinchun Yin
[1]Department of Information Engineering, Yangzhou University, Yangzhou Jiangsu, China
[2]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing Jiangsu, China

Abstract: A self-healing mechanism in key management is an important means for large-scale clustering wireless sensor networks that enable non-revoked nodes use their private information and the received broadcast messages to recover the lost session keys on their own. In this study, we propose a dynamic self-healing key management scheme for large-scale clustering wireless sensor networks that is based on Exclusion Basis System (EBS). We use forward and backward key chains to form cluster session key chain for self-healing, take t-degree polynomial keys to replace the original keys used in EBS. The analysis shows that the proposed scheme has the properties of forward and backward secrecy and resisting to a collusion attack, which is suitable for resource-constrained wireless sensor networks.

Keywords: Dynamic key management, Exclusion Basis System (EBS), self-healing, wireless sensor networks

INTRODUCTION

Recently, as one of the core technologies of the Internet of Things, Wireless Sensor Networks (WSNs) is attracting more and more research interests because of its wide applications such as military operations, scientific explorations and so on. Among all security issues in WSNs, key management is a fundamental security issue for wireless sensor networks.

Staddon *et al*. (2002) first proposed self-healing key distribution schemes with revocation capability in WSNs in 2002 (Staddon *et al*., 2002). Blun Do *et al*. (2003) analyzed Staddon's schemes and showed that an adversary could though just broadcast messages to recover the group session key which proved that (Staddon *et al*., 2002) is not safe (Blun Do *et al*., 2003). Later on many self-healing key distribution schemes (Liu *et al*., 2003) based on Staddon *et al*. (2002) are proposed, Liu *et al*. (2003) proposed a novel method by combining the personal secret key distribution scheme with the self-healing technique to improve the scheme in Staddon *et al*. (2002). Dutta *et al*. (2007) proposed a self-healing group key distribution scheme based on one-way key chain (Dutta *et al*., 2007). Du and He proposed a self-healing key distribution with revocation which is claimed to resist to the collusion attack (Du and He, 2008). Bao and Zhang found the scheme in Du and He (2008) is not secure against the collusion attack and proposed a modified scheme (Bao and Zhang, 2011). Eltoweissy *et al*. (2004) proposed a combination of dynamic key management scheme EBS based on combinatorial optimization methodology (Eltoweissy *et al*., 2004). Kim *et al*. (2006) proposed an EBS and t-degree bivariate based polynomial group key management scheme (Kim *et al*., 2006).

In this study, we propose a dynamic self-healing key management scheme for large-scale clustering wireless sensor networks that is based on Exclusion Basis System (EBS). We use forward and backward key chains to form cluster session key chain for self-healing, take t-degree polynomial keys to replace the original keys used in EBS. The analysis shows that the proposed scheme has the properties of forward and backward secrecy and resisting to a collusion attack, which is suitable for resource-constrained wireless sensor networks.

LITRATURE REVIEW

Network and adversary model:

Network model: Research shows that cluster-style network topology is more suitable for large-scale energy-constrained wireless sensor networks. WSN nodes determine how clustering based on their location information or other criteria. We assume that the network nodes are divided into two categories:

- **Cluster head node:** This scheme assumes that, cluster head node' energy is sufficient to support the basic study requirements of each cluster and is

Corresponding Author: Jianting Ning, Department of Information Engineering, Yangzhou University, Yangzhou Jiangsu, China

assigned by system. They are mainly as a data gathering point within cluster, sent data to remote terminal after simple data processing. In addition to data aggregation, such nodes will be responsible for key distribution, updating, sensor nodes join and eviction, resistance against attacks and so on. A small number of cluster head node exists in the network.

- **Sensor nodes:** Compared to cluster head node, sensor node has less storage capacity, energy, weak computing power. It is responsible for sensing the external environment, to obtain and transfer data to the cluster head node belonged to after a simple treatment. Lots of these nodes distributed in the network.

Adversary model: For distributed in an open and hostile environment, sensor nodes face communication monitor, Sybil attack and so on. Among them, the collusion attack arising from node capturing is a direct threat to the security of key systems. How to resist collusion attack, to prevent as captured node increases, adversary with more key information until owns and explains the whole key system, is to be considered an important issue in WSNs key management scheme.

Forward and backward key chains: In short, Randomly select key seeds KF_0, KB_0, one-way hash function H (.), H_B (.), d numbers $\delta_1, \delta_2, \ldots, \delta_d$. Then we can get the corresponding forward key chain $KF_1, KF_2, \ldots, KF_d$. for *d* sessions through:

$$KF_j = H(KF_{j-1}, \delta_j) = \ldots = H^{j-2}(KF_1, \delta_2)$$

$H^{j-1}(KF_0, \delta_1)$ $(1 \le j \le d)$. For session *1*, the forward key is $KF_1 = H(KF_0, \delta_0)$, the backward key seed is $KB_1^0 = H(KB_0, \delta_1)$ and the backward key chain is $\{KB_1^0\}$. For session 2, the forward key is $KF_2 = H(KF_1, \delta_2)$, the backward key seed is $KB_1^0 = H(KB_1, \delta_2)$ and the backward key chain is $\{KB_2^0, KB_2^1\}$. For session j $(1 \le j \le d)$, the forward key is $KF_j = H(KF_{j-1}, \delta_j)$, the backward key seed is $KB_j^0 = H(KF_{j-1}, \delta_j)$, then the backward key chain is $KB_j^0, KB_j^1, \ldots, KB_j^{j-1}$, *j=1, 2,... d,* where, $KB_j = KB_j^{j-1} = H_B(KB_j^{j-2}) = \ldots = H_B^{j-1}(KB_j^0)$. For session j, the cluster session key is $CK_j = KF_j + KB_{d-j+1}$, as showed in Fig. 1.

Dynamic key management scheme EBS: Let n, k and m be positive integers, such that k>1, n>m. An Exclusion Basis System of dimension (n, k, m), denoted by EBS (n, k, m), is a collection Γ of subsets of [1, n] = {1, 2,…, n} such that for every integer $t \in [1, n]$ the following 2 properties hold:

Fig. 1: The structure of forward and backward key chains

- t is in at most k subsets of Γ
- There are exactly m subsets

say A1, A2,…, Am, in Γ such that $U_{i=1}^{m} A_i = \{1,2, \ldots, n\}\{t\}$. (That is, each element t is excluded by a union of exactly m subsets in Γ).

We take the EBS *(n, k, m)* described above as a wireless sensor network dynamic key management method, n is the number of nodes, k is the number of administrative keys and m is the number of rekeying messages. A set of (k + m) administrative keys is used to support a set of n nodes and each node is assigned a distinct combination of k keys. A node can be simply admitted to the group by assigning one of the unused set of k keys out of the total of C (k + m, k), i.e., $(K + m)!/(k!\, m!)$, distinct combinations. Ejection of a compromised node can be performed by broadcasting replacement of the k keys that the evicted node knows using the m keys.

NEW SCHEME FOR WSNS

In this study, we present a dynamic EBS-based key management scheme with the property of self-healing. The skeleton of self-healing in this scheme is mainly showed in Fig. 2.

System initialization: Assume that the life cycle of a communication is divided into d sub-sessions. And the cluster key in every session will be updated periodically. At system initialization, the system select key seeds KF_O, KB_O, on-way hash functions $H(.)$, $H_B(.)$ and $H_1(.)$ and d numbers $\delta_1, \delta_2, \ldots, \delta_d$. randomly. Then system randomly chooses numbers $\alpha_1, \alpha_2, \ldots, \in F_q$ for each session, then generate the forward key chain $\{KF_1, KF_2, \ldots KF_d\}$ and the corresponding *d* backward key chains $\{KB_j^0, KB_j^1, \ldots, KB_j^{j-1}\}$. $(1 \le j \le d)$ as described above. After strict registration and authentication, each sensor node N_a within cluster will be assigned its unique Id_a, forward hash function $H(.)$ and one-way hash function $H_1(.)$ which used for updating. And each node will be allocated a key-buffer of length L (kb (L), …, kb (1)) and two key-slots.

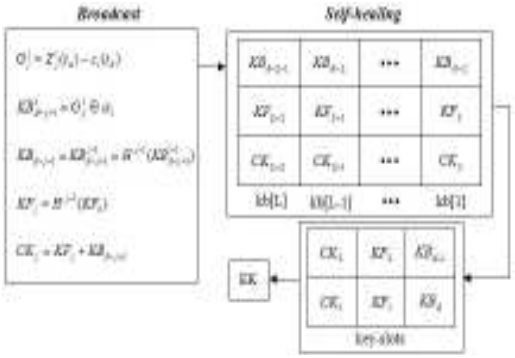

Fig. 2: The skelecton of self-healing

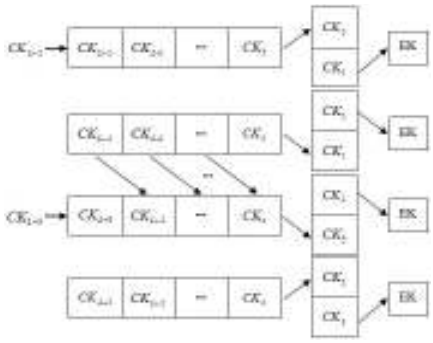

Fig. 3: Management of CKs

Key pre-distribution: Cluster head node chooses two polynomials of degree t at random $f(x) = f_0 + f_1 x + \cdots + f_t x^t$ and $s(x) = s_0 + s_1 x + \cdots + s_t x^t$, which is based on (t + 1, n) threshold secret sharing technique. Then cluster head node randomly generates a number of k + m polynomial administrative keys $f_n(x) = \sum_{i=0}^{t} f_i x^i$, n = 1, 2,…, k + m and distributes unused set of k polynomial keys out of the total of C (k + m, k) to each sensor node within cluster according to a pre-generated random EBS matrix. Mean while, the cluster head node will broadcast $BN_a = \{L|KF_0|KB_{d+L-1}|s_d(a)|...|s_{L+3}(a)|\delta_1|...|\delta_L$. Sensor node N_a receive and decrypt B_{N_a} to get L, KF_O, KB_{d-L-1}, $S_d(a)$,..., $s_{L+3}(a)$, $\delta_1, \delta_2, \ldots, \delta_L$. Then N_a calculates $CK_j = KF_j + KB_{d-j+1}$, $(1 \le j \le L+2)$, stores $\{CK_{L+2},...,CK_3\}$ and $\{CK_2, CK_1\}$ in the key-buffer and key-slots, respectively. CK_1 is used for the present cluster session key and when the timer expires, switches the active key to CK_2 and right move CK_s as showed in Fig. 3.

Broadcast: Assume that $N = \{N_1, N_2, \ldots, N_P\}$ is the set of all active sensor nodes for j-th session, where, p is the number of active user in session j. Let $T = \{t_1, t_2, \ldots, t_P\}$ be the set of all active users' secret values in j-th session. In session j, cluster head node generates a masking key sequence $\{G_j^1, G_j^2, \ldots, G_j^{j-1}, G_j^j\}$ where, $G_j^i = KB_{d-j+1}^i \oplus \alpha_i$ (j = 1, 2,…, d; i = 1, 2,…, j), then broadcasts the following message:

$$B_0 = \{z_j^i(x)_{= A_j^i(x)G_j^i + s_i(x)\}} \cup \{E_{KB_{d-j+1}^0}(\delta_1), E_{KB_{d-j+1}^1}(\delta_2)$$

$E_{KB_{d-j+1}^{j-1}}(\delta)\}$, i = 1, 2,… j, where, S_j^i is a randomly number to mask G_j^i, $\frac{S_j^i}{T_j} \notin T = \{t_1, t_2, \ldots, t_P\}$, and $A_j^i(x) = 1 - (S_j^i x - T_j)\prod_{n=1}^{p} x - t_n$. $N_i \subset N = \{N_1, N_2, \ldots, N_p$ Receives the j-th broadcast message B_j, N_i can evaluate $A_j^i(x) = 1$ by using its secret value t_i. For any revoked user, however, the $A_j^i(x)$ is a random value.

Cluster session key and self-healing key recovery: Assume Suppose that sensor node N_a joins in the cluster in session I and not revoked in session $(1 \le i \le j$ n j, then it can recover the cluster session key CK_j from the broadcast message B_j as follows:

- N_a computes $G_j^i = Z_j^i(t_{d)-}s_i(t_d)$ where, $A_j^i(t_{d)} = 1$
- N_a evaluates $KB_{d-j+1}^i - G_j^i \oplus \alpha_i$
- N_a computes all the future $\{KB_{d-j+1}^i, KB_{d-j+1}^{i+1}, \ldots, KB_{d-j+1}^{j-1}\}$ through the one-way hash function $H(.)$, then get $KB_{d-j+1} = KBd{-}j{+}1j{-}1 = Hj{-}iKBd{-}j{+}1i{-}1$. Meanwhile, N_a calculates the forward key $KB_j = H^{j-i}(KF_o)$ by using the preloaded key seed KF_o and one-way hash function $H(.)$. Thus, gets the cluster session key $CK_j = KF_j{+}KB_{d\text{-}j+1}$ for j-th session;
- N_a can decrypt $\{E_{KB_{d-j+1}^{i-1}}(\delta_i), E_{KB_{d-j+1}^{i-1}}(\delta_{i+1}), \ldots, E_{KB_{d-j+1}^{i-1}}(\delta_i)\}$ by using the corresponding keys $KB_{d-j+1}^i, = KB_{d-j+1}^{j+1}, \ldots, = KB_{d-j+1}^{j-1}$, thus getting the corresponding self-healing keys $\{\delta_i, \delta_{i+1}, \ldots, \delta_j\}$. If N_a has already obtained KB_{d-j+1} from B_j, he can recover all the session keys $KB_{d\text{-}l+1}$ (j<l<j) with KB_{d-j+1} and the self-healing keys $\{\delta_i, \delta_{i+1}, \ldots, \delta_j\}$.

Key update: The administrative and session keys need to be updated periodically or on-demand within cluster in order to improve system's security. The cluster head node broadcasts update packets:

$$B_j = \{\dot{f}(x)\} \cup \{Z_j(x) = A_j(x)G_j + \dot{s}(x)\} \cup \{E_{KB_{d-j+1}^0}(\delta_1)$$

$E_{KB_{d-j+1}^{1}}(\delta_2), \ldots, E_{KB_{d-j+1}^{j-1}}(\delta_j)\}$, where, $\dot{f}(x) \sum_{i=0}^{t} \dot{f}_i x^i$ $\dot{s}(x) = \sum_{i-0}^{t} \dot{s}_i x^i$, $\dot{f}_i = H_1(f_i)$, $\dot{s}_i = H_1(s_i)$. Each node receives and decrypts the packet, calculates $\dot{f}(x)$ to replace the original $f(x)$, thus completing the administrative key update. Also, each node can decrypt the packet and calculate to get $CK_j = KF_j + KF_{d-j+1}$ like described above. And CK_j will be put in the key-buffer and switches the active-key like showed in Fig. 3.

Add new sensor node: To maintain a good connectivity in the network, new nodes need to be added into network to replace dead nodes and. First system detects EBS matrix for all the cluster distributions and find out the smallest number of nodes n among whole clusters which meets $n \leq C(k+m,k)$, then assigns one of the unused set of k keys to the new sensor node and preload its unique Id, thus admitting the new node. Meanwhile, the new node will stores forward hash function $H(\cdot)$ and one-way hash function $H_1(.)$used for updating. And the new node will be allocated a key-buffer of length L (kb (L),…, kb (1) and 2 key-slots. If all clusters meet $n \leq C(k+m,k)$, then the system will assign a new cluster head node H_{new} to form a new cluster, the new cluster head node H_{new} will distribute the key to the new sensor node as described above.

Node ejection: When system detects an abnormal sensor node N_i, response should be made by system immediately. According to EBS, we need to update all k administrative keys $E_i^{P_j}$, j = 1, 2,…, k, $p_j \in \{1, 2, \ldots, k+m\}$ that N_i owns. So the cluster head node broadcasts m data packets: $E_i^{q_s}(E_i^{q_1}(\dot{f_{p_1}}), E_i^{p_2}(\dot{f_{p_2}})), \ldots, E_i^{p_k}(\dot{f_{p_k}})$, s = 1, 2,… m $\{q_1, q_2, \ldots\ldots, q_m\} \in \{1,2,\ldots\ldots,k+m\} - \{p_1, p_2, \ldots\ldots, p_k\}$ within the cluster, $\dot{f}Pi$ Piis the new administrative key that is generated by using the one-way hash function $H_1(.)$. According to EBS matrix, since N_i has not been assigned by $E_i^{q_s}$, it cannot decrypt any data packets, thus unable to update the key and being ejected from the network. According to EBS matrix, the other nodes could be assigned one or more $E_i^{q_s}$, so they can decrypt data packet to get update information to form newly polynomial keys, thus being retained in the network.

RESULT ANALYSIS

Computational overhead: We build our quantitative analysis of the proposed scheme's performance according to steady-state distributions of 2-dimensional Markov chain. Let p_L = pr {B_j is lost}, P_F = Pr {B_j authentication fails $|B_j$ is received} and P_s = 1-p_L-P_F. Also, let $p(I, j)$ denote the steady-state probability of state (I, j) and $p_\varepsilon(k)$ the probability that there were exactly k empty slots. Then, we can get:

$$p_\varepsilon(k) = (p_L + p_F)^k \cdot p(0,0), k = 0,\ldots,d$$

$$p(0,0) = \frac{1-(p_L+p_F)}{1-(p_L+p_F)^{d+1}}$$

Since the sensor nodes have relatively limited computing resources, so we only consider the computational overhead of sensor nodes. The cost of computing of each node include polynomial evaluate, sum operation and Hash operation and the computational overhead of polynomial operation is relatively small and fixed, so we mainly focus on the hash computation to analyze the computational overhead. Assume that E(N_H) is the expected number of hash computations per updating, when there are k empty slots in key-buffer, we have:

$$E(N^H) = \sum_{k=0}^{d-1}(k+1)(1-p_L)(p_L+p_F)^k p(0,0) + (d+1)$$

(p_L + p_F) $d.p(o, o)$. Compared to the proposed scheme LiSP, we have the same expected number of hash computations after the node receives the key update message with Lisp. Moreover, our scheme is based on EBS and we take the t-degree polynomial keys to replace the original keys used in EBS, which makes our scheme more effective than LiSP against collusion attack. Figure 4 gives the relationship between the expected number of hash computations per updating and the length of key-buffer, which shows that we only execute small hash computation even if under a highly lose of wireless channel.

Communication overhead: The overhead of communication between the cluster head node and the sensor nodes is mainly about 2 parts: the cost of init key C_{init} and the cost of update key C_{update}. When the d sub-sessions circle is finished or the L key-buffer is empty, the cluster head node needs to re-initialize, in other case the cluster head node only to broadcast to update periodically, We take n is the ratio of the key C_{update}, so we can get the expected communication cost $E_{comm} = n.[\frac{1}{d} + (p_L+p_F)^d.\ p(0,0) + \sum_{t=0}^{d-1}(p_L+p_F)^t$. Figure 5 shows the relationship between the expected communication cost and the length of key-buffer, which indicates that the longer the length of the key-buffer, the smaller the communication overhead requires.

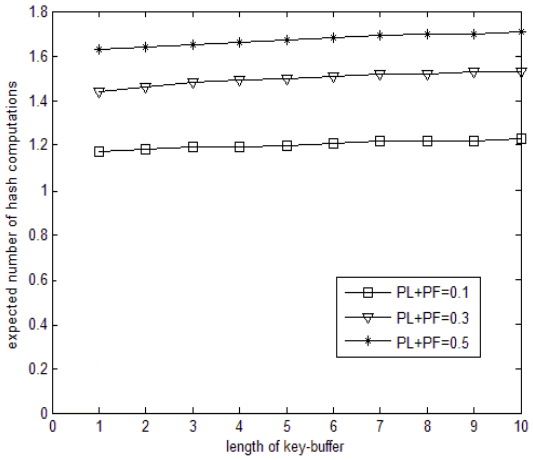

Fig. 4: The relationship between the expected number of hash computations per updating and the length of key-buffer

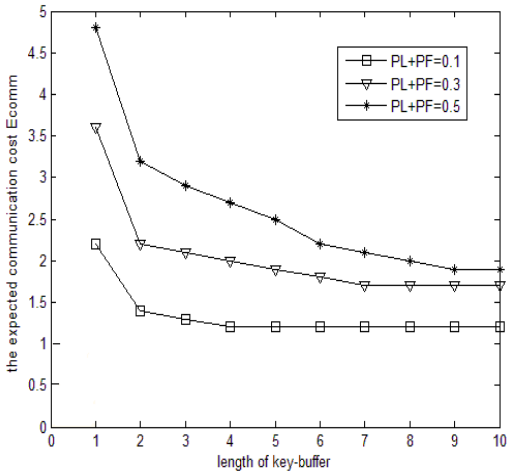

Fig.5: The relationship between the expected communication cost and the length of key-buffer

Forward secrecy: Suppose that R_j is the set of the nodes revoked in and before session j. For the broadcast message:

$$B_j = \{Z_j^i(x) = A_j^i(x)G_j^i + s_i(x)\} \cup \{E_{KB_{d-j+1}^0}(\delta_1), E_{KB_{d-j+1}^1}$$

$E_{KB_{d-j+1}^{j-1}}(\delta_j)\}$ as mentioned above and a node needs to obtain the corresponding self-healing key δ_j which is randomly chosen and an active node's secret to get the j-th cluster session key $CK_j = KF_j + KF_{d-j+1}$. The corresponding self-healing key δ_j is encrypted by the corresponding backward session key KB_{d-j+1}^{j-1}, which $KB_{d-j+1}^{j-1} = G_j^{j-1} \oplus \alpha_{j-1}$. But for any revoked node $N_R \in R_j$, cannot calculate the masking keys G_j^{j-1}, because $A_j^i(x)$ is a random value for N_R. Moreover, for any revoked node $N_R \in R_j$ do not have α_{j-1}. Therefore, the nodes in R_j cannot get the values of the self-healing keys to obtain the future group session keys. Meanwhile, $f(x)$, $S(x)$ are t-degree polynomial based on $(t + 1, n)$ threshold secret sharing technique which need $t + 1$ points to recover. The above analysis shows that the proposed scheme is forward secure.

Backward secrecy: Suppose that U_{j+1} is the set of the nodes which join in session $j + 1$. For users in U_{j+1} can only get the current backward session key KB_{d-j} compute CK_{j+1} which is the last key in the cluster key chain from the broadcast B_{j+1}. Thus users in U_{j+1} can only get the current self-healing key δ_{j+1}. Since one-way hash function is irreversible, it is computationally infeasible for any user in U_{j+1} use KB_{d-j} and δ_{j+1} to compute the previous cluster key CK_j. Meanwhile, $f(x)$, $S(x)$ are t-degree polynomial based on $(t + 1, n)$ threshold secret sharing technique which need $t + 1$ points on the polynomial $f(x)$, to recover. $\dot{f}(x)$ is the result of $f(x)$ after one-way hash operation, since one-way hash function is irreversible, so it is hard to recover the previous polynomial $f(x)$. So nodes in U_{j+1} can not get any information about the previous cluster session key or administrative key. The above analysis shows that the proposed scheme is backward secure.

Resistance to a collusion attack: Suppose that R_j is the set of the nodes revoked in and before session j_1+1 and U_{j_2} is the set of the nodes that join from session j_2. Assume that $N_R \in R_j$ colludes with $N_{j_2} \in U_{j_2}$, they need the self-healing keys between δ_{J_1} and δ_{J_2} to recover $KB_{d-j+1} = KB_{d-j+1}^{j-1}$ which could be used to compute CK_j $(j_1 < j < j_2)$. For the equation $Z_{j_2}^{j_1}(x) = A_{j_2}^{j_1}(x)G_{j_2}^{j_1} + s_{j_1}(x)$ from B_{j_2}, U_{j_2} needs the value of $s_{j_1}(x)$ to obtain $G_{j_2}^{j_1}$. And N_{j_2} can get $K_{d-j_2+1}^{j_1-1}$ with N_R's secret α_{j_1}. Thus, N_R and N_{j_2} can get $\{K_{d-j_2+1}^{j_1-1}, \ldots, K_{d-j_2+1}^{j_2-1}\}$ and $\{s_{j_1}, \ldots, s_{j_2}\}$. But they cannot obtain $s_{j_1}(x)$ unless they can recover the secret polynomial $s(x)$ which based on $(t + 1, n)$ threshold secret sharing technique. Therefore, they cannot recover the self-healing keys between s_{j_1} and s_{j_2}. Thus they cannot get the backward session keys $KB_{d-j+1} = KB_{d-j+1}^{j-1}$ to calculate CK_j without the corresponding self-healing keys. Therefore, the proposed scheme can resist to the collusion attack. Compared to Lisp, SHELL, LOCK, results from

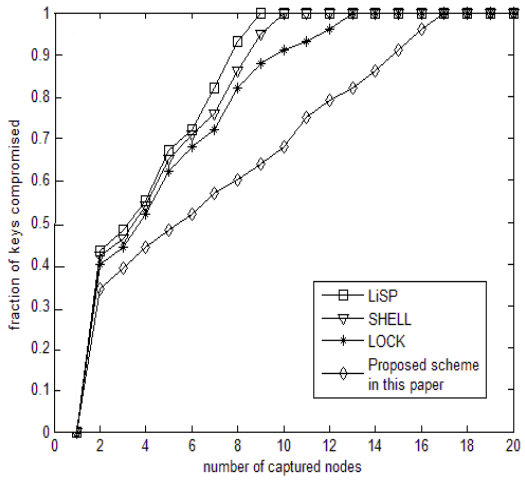

Fig. 6: Relationship between the number of captured nodes and the fraction of keys compromised (m = 5)

simulation and analysis indicate that the proposed scheme in this study is more resilient against node capture and collusion attack and Fig. 6 gives the relationship between the number of captured nodes and the fraction of keys compromised.

CONCLUSION

We propose a dynamic key management scheme for wireless sensor networks with the property of self-healing. We take t-degree polynomial keys to replace the original keys used in EBS, use forward and backward key chains and broadcast polynomial key to achieve self-healing, forward and backward secrecy and resisting to a collusion attack. Meanwhile, this scheme has a small calculation and communication overhead, which is efficient and secure for resource-constrained wireless sensor networks.

ACKNOWLEDGMENT

The authors wish to thank the helpful comments and suggestions from my teachers and colleagues in Yangzhou University. This study was supported in part by National High Technology Research and Development Program of China (863 Program) (2007AA0124487), National Natural Science Foundation of China (60473012) and Colleges and Universities of Jiangsu Province Plans to Graduate Research and Innovation (CXLX11_1008).

REFERENCES

Bao, K.H. and Z.F. Zhang, 2011. Collusion Attack on a Self-Healing Key Distribution with Revocation in Wireless Sensor Networks. In: Chung, Y. and M. Yung (Eds.), WISA 2010. LNCS 6513, Springer, Heidelberg, pp : 221-233.

Blun Do, C., P. D'Arco and M. Listo, 2003. A flaw in a self-healing key distribution scheme. Proceeding of Information Theory Workshop, Paris, pp: 163-166.

Du, W. and M.X. He, 2008. Self-healing Key Distribution with Revocation and Resistance to the Collusion Attack in Wireless Sensor Networks. In: Baek, J., F. Bao, K. Chen and X. Lai (Eds.), Prov Sec LNCS. Springer, Heidelberg, 5324: 345-359.

Dutta, R., E. Chang and S. Mukhopadhyay, 2007. Efficient Self-healing Key Distribution with Revocation for Wireless Sensor Networks Using One Way Hash Chains. In: Katz, J. and M. Yung (Eds.), ACNS 2007. LNCS, Springer, Heidelberg, 4521: 385-400.

Eltoweissy, M., H. Heydari, L. Morales and H. Sadborough, 2004. Combinatorial optimization of group key management (J): Special issue on network security. J. Netw. Syst. Manag, 12(1): 33-50.

Kim, J., J. Cho, S. Jung and T. Chung, 2006. An energy-efficient dynamic key management in wireless sensor networks (A). Proceedings of the 8th International Conference on Advanced Communication Technology (C). Phoenix Park, Korea, pp: 2148 - 2153.

Liu, D., P. Ning and K. Sun, 2003. Efficient self-healing key distribution with revocation capability. Proceeding of the 10th ACM CCS, pp: 27-31.

Staddon, J., S. Miner, M. Franklin, D. Balfanz, M. Malkin and D. Dean, 2002. Self-healing key distribution with revocation. Proceeding of IEEE Symposium on Security and Privacy, pp: 241-257.