An Efficient and Sustainable Self-healing Protocol for Unattended Wireless Sensor Networks

Juan Chen[1], Hongli Zhang[1], Binxing Fang[1,3], Xiaojiang Du[2], Haining Yu[1], Xiangzhan Yu[1]

[1]Research Center of Computer Network and Information Security Technology, Harbin institute of technology,

Harbin, China, e-mail: juanchencs@gmail.com

[2]Dept. of Computer and Information Sciences, Temple University, Philadelphia, PA, USA, e-mail: dux@temple.edu

[3]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, e-mail: fangbx@cae.cn

***Abstract*—Due to the unattended operation nature, nodes in Unattended Wireless Sensor Networks (UWSNs) are susceptible to physical attacks. Once a sensor is compromised, the adversary will be able to learn all its secrets. While some previous works tried to address the node self-healing issue in UWSNs, little effort has been devoted to ensure the sustainability of node self-healing. In this paper, we present a novel sustainable node self-healing protocol for UWSNs. We generate unpredictable random data for key update and thus the node self-healing capability doesn't decrease when the number of attack rounds increases. We show both analytically and through simulation experiments that our protocol provides efficient and sustainable node self-healing capabilities with small overheads.**

Keywords - unattended wireless sensor networks; self-healing; sustainable

I. INTRODUCTION

Unattended Wireless Sensor Networks (UWSNs) have many commercial and military applications. Different from traditional WSNs, an UWSN is left unattended for most of the time after deployment. An UWSN has a mobile base station (BS) that visits the network with some frequency.

In UWSNs, the unattended feature makes nodes extremely vulnerable to attacks happened between visits of the mobile BS. An attacker may compromise nodes one by one, obtain their secrets and then leave the network without being noticed. In UWSNs, node compromise attack may cause severe damage to the network. First, if the node secrecy information is leaked, any cryptographic protocol that depends on the secrecy (e.g., keys) would become useless. Second, the attacker may decrypt and obtain important sensing data. Last but not least, it is easy for an attacker to delete data, forge data and authentication.

Data collected from a compromised sensor may be classified into two categories, based on the time of collecting them: (1) before compromise, and (2) after compromise. *Forward secrecy* means that, even if an attacker obtains the sensor's current secrets, he cannot decrypt (or forge authentication tags for) data collected and encrypted (or authenticated) before compromise. *Backward secrecy* means that, an attacker who obtains the sensor's current secrets cannot decrypt (or forge authentication tags for) data after compromise [3].

Forward security is relatively easy to obtain by key evolution such as hash function [6-8], which doesn't help on backward security. Pietro [9] provides both forward and backward secrecy by using public key cryptography. Unfortunately, Public key cryptography is not suitable for WSNs due to the large computational overhead [10]. DISH [1] and POSH [2] achieves both forward and backward security by key evolution and node cooperation. In DISH, each node requests for random data from randomly selected nodes and then updates its key based on the random data and its current key. Different from DISH, in POSH each node selects some nodes as the recipients and sends a random data to each of them. POSH does not need to send extra data-request message and hence achieves much lower communication cost than DISH. Unfortunately, in POSH some nodes may not receive any random data, which makes the node self-healing capability not as effective as DISH.

However, both POSH and DISH don't consider node failures and message losses, which are common in real sensor applications. Furthermore, both POSH and DISH generate random data by a Pseudo-Random Number Generator (PRNG). A PRNG is an algorithm that starts with a seed - and uses some function(s) to produce a sequence of values that appear random [3]. If an attacker compromises a sensor, he can obtain the PRNG algorithm and compute all subsequent random values. Under this attack, a PRNG cannot provide backward security either. An alternative way to per-sensor PRNGs is to use a True Random Number Generator (TRNG). Compared to PRNGs, TRNGs extract randomness from physical phenomena and hence the random numbers are non-deterministic and cannot be pre-computed. However, TRNG is only suitable for nodes equipped with extra hardware [4, 5]. TRNG is not suitable for small sensors. These challenges motivate our work.

In this paper we propose an Efficient and Sustainable Self-Healing (ESSH) protocol that helps sensors recover from node-compromise-attack with high probability. The main contributions are summarized below:

1) ESSH is effective and sustainable. ESSH is effective since compromised nodes can recover with high probability. ESSH uses random numbers obtained by unpredictable random data generation scheme instead of a PRNG. Furthermore, ESSH is sustainable as the node self-healing capability doesn't decrease when the number of attack rounds increases.

978-1-4673-0921-9/12/$31.00 ©2012 IEEE

2) ESSH is lightweight. ESSH uses hash functions and symmetric cryptography and hence it has low computation cost. Extensive analyses and simulation show that: the communication cost of ESSH is comparable to that of POSH; the communication cost of ESSH is only about 50% of DISH.
3) ESSH performs well in unreliable UWSNs. With a random data compensation scheme, ESSH can handle real network issues, such as message losses and node failures.

The remainder of this paper is organized as follows. Section II outlines the network and attack model. Section III presents the node self-healing protocol. Section IV provides our performance analysis. Section V gives simulation results. Finally, Section VI concludes this paper.

II. Network and Attack Model

A. Network Model

We consider an UWSN with n homogeneous sensors. The unattended sensors are scattered over a region to execute pre-determined tasks such as data gathering. Two sensors can communicate with each other either directly or via intermediate nodes. Time is divided into rounds, and nodes are synchronized.

Each node is scheduled to collect, encrypt and then store exactly one sensing data per round. Different from the traditional WSNs, the BS in UWSNs is mobile and it visits the network at infrequent intervals. BS is supposed to visit the network every e rounds. However, BS might cancel a visit for its safety considerations. Once entering the network, BS collects sensing data from each node, re-initialize secret seed values for each node and reset the round counter to 1. During the interval between two BS visits, sensors are left unattended and they may be attacked.

B. Attack Model

We consider a powerful attacker. The attacker's goal is to learn as many nodes' privates as possible while keeping himself unobservable. He may decrypt stored data, forge data and authentication using obtained keys without being noticed. More specifically, we consider an attacker with following capabilities:

- *Resource rich* - The attacker has adequate computation capability. He can also move at will.
- *Local monitoring* - The attacker can eavesdrop messages within its transmission range.
- *Active attack* - Each round, the attacker may select g nodes, attack them and obtain all their privates.

We consider an attack strategy the same as in DISH [1]. Each round the attacker chooses to attack g sensors that have not been compromised in the past. If all sensors have been attacked before, he chooses sensors that were attacked a long time ago, because with high probability these sensors may obtain secure privates by a self-healing protocol.

III. The efficient and Sustainable Self-healing Protocol

A. Motivation

First, we give some definitions.

Definition 1: A V-data is a random data that is received by a node for key update.

Definition 2: A secure V-data is a V-data that is unknown to the attacker.

Definition 3: An unsecure V-data is a V-data that is directly obtained or indirectly inferred (i.e., known) by an attacker.

Definition 4: p_r denotes the node self-healing capability in round r (r>0). p_r is defined as the probability that a node receives at least one secure V-data in round r.

DISH/POSH generates random data by a PRNG. Thus, once a node is attacked, it will never provide secure V-data. Recall that the attacker can infer the node's future keys by the obtained PRNG. Depending on whether a node is controlled by an attacker and whether a node can generate secure V-data, a node could be in one of the following four states:

- **Healthy**: The node has never been attacked and it can provide secure V-data to other nodes.
- **Sub-healthy**: The node has been attacked but it has updated its key based on some received secure V-data. However, it can only provide unsecure V-data.
- **Sick**: The node is controlled by an attacker and any V-data comes from it is not secure.
- **Released**: The node has been attacked and it hasn't received a secure V-data since then. Hence, its future keys can be inferred by the attacker and it only provides unsecure V-data.

Table I lists the status (secure or unsecure) of a V-data provided by a node in different states under DISH and POSH. We can see from Table I that only nodes in Healthy state can provide secure V-data.

TABLE I. V-DATA PROVIDED BY A NODE IN DIFFERENT STATES

	DISH	POSH
Healthy	secure	secure
Sub-healthy	unsecure	unsecure
Released	unsecure	unsecure
Sick	unsecure	unsecure

Fig.1 is the state transition diagram for DISH/POSH. It can be seen from Fig.1 that once a node is compromised, the attacker can predict its future key by using PRNG obtained from the node. Hence, even if received a secure V-data, the compromised node can only transit to the Sub-healthy state but not the Healthy state. With more rounds of attacks, in DISH/POSH the number of nodes that can provide secure V-data decreases. After several rounds, no healthy node exists in the network and no secure V-data can be provided. As a result, the node self-healing capability decreases to zero. Therefore, DISH/POSH cannot provide sustainable self-healing capability for a WSN.

To address this issue, we propose the ESSH protocol that provides sustainable node self-healing capability.

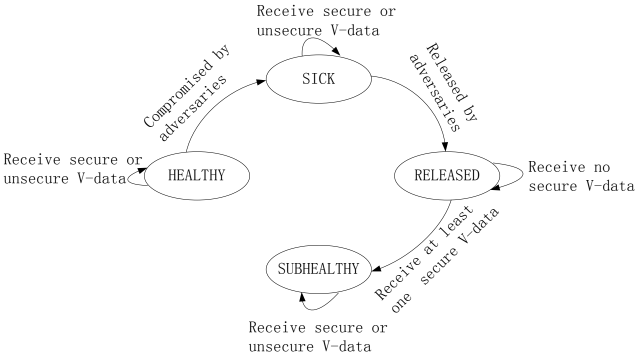
Receive secure or unsecure V-data
SICK
Compromised by adversaries
Released by adversaries
Receive secure or unsecure V-data
HEALTHY
RELEASED
Receive no secure V-data
Receive at least one secure V-data
SUBHEALTHY
Receive secure or unsecure V-data

Fig.1. State transition for DISH/POSH

B. Overview of ESSH

Different from POSH/DISH, in ESSH a V-data is jointly generated by several nodes instead of one node. For example, in round *r*+1, the V-data comes from node *i* to *j* is generated by all nodes in $R_{i,j}$, where $R_{i,j}=\{i,...\}$ denotes the set of nodes on the route from *i* to *j*. For $\forall l \in R_{i,j}$, *l* contributes a value $T_r^{(l)}$ to the V-data. $T_r^{(l)}$ is computed by some random physical parameters of node *l,* such as the number of error bits in the received message, and the message transmission delay in round *r*. The attacker cannot predict these parameters. Hence, $T_r^{(l)}$ and the V-data are unpredictable.

Specifically, our ESSH protocol consists of three phases: pre-deployment, random data receiving/forwarding and random data compensate. In the random data receiving/forwarding phase, ESSH generates secure V-data by the unpredictable random data generation scheme. Therefore, a compromised node can receive a secure V-data with high probability. And also the recover ability of a node doesn't decrease as the number of attack rounds increases. In the random data compensation phase, with the random data compensation scheme, ESSH can cope with real network issues (e.g., message losses and sensor failures) and further improves node self-healing ability.

C. Pre-deployment

Before deployment, each node *i* is preloaded with several parameters: random number β_i, hash function H_1 and pair-wise key k_i^0, where k_i^0 is the initial key shared between *i* and the BS. k_i^0 is updated every round and it is re-initialized when BS visits the network again. Node *i* is also preloaded with a routing table T_i including the next hop from *i* to any node in the network.

D. Random Data Receiving/Forwarding

In this phase, nodes generate V-data through cooperation and update their keys by received V-data.

Data Receiving Phase. In round *r*, node *i* (as a volunteer) generates *t* recipients by H_1 and sends each of them a V-data Generation (VDG) message. A VDG message mainly includes a V-data field and is used to generate unpredictable V-data.

In POSH/DISH, an attacker can predict a compromised node's future key by obtained PRNG. Thus once being attacked, the node cannot provide secure V-data. If a V-data is generated by different nodes using unpredictable value (e.g. a value related to the network performance) instead of a PRNG, an attacker cannot predict the V-data. So, we propose a simple but effective random data generation scheme: an unpredictable random data generation scheme based on network performance (URDG-NP).

In URDG-NP, at the end of each round, each node say *i* calculates a random number $T_r^{(i)}=F(e_1,e_2,\ldots)$, where e_u ($u\geq 1$) denotes a parameter related to *i*'s local network performance such as the average message transmission latency, message retransmission times, the number of error bits from received message, etc. *F* is a randomly chosen multivariate function. Since $T_r^{(l)}$ changes with the network performance which is undetermined, $T_r^{(l)}$ is unpredictable. In order to enhance the unpredictability of a V-data, V-data is computed as the accumulation of T_r from nodes which the VDG message passes through. For example, if node *j* receives a V-data *v*' comes from *i* in round *r*+1, then *v*' is computed as $\sum_{l\in R_{i,j}} T_r^{(l)}$.

Different from POSH/DISH, in ESSH if a node is compromised in some round and then released; it can still contributes unpredictable value to generate secure V-data in cooperation with other nodes. Therefore, a compromised node in ESSH can recover with high probability and its self-healing ability will not decrease (even disappear) with the increase of attack rounds.

Data Forwarding Phase. When a node, say *i* receives *t*' (*t*'≥0) VDG messages in round *r*, *i* updates its pair-wise key for the next round by Eq. (1)

$$k_i^{r+1} = H(k_i^r||V_i[1]||\ldots||V_i[t']) \tag{1}$$

where $V_i[q]$ ($1\leq q\leq t$) denotes the *q*-th V-data received by *i*. If node *i* has not received any V-data, *i* can obtain V-data by the random data compensation scheme in the next phase. After that, *i* encrypts and then stores the received V-data and sensing data by k_i^{r+1}.

E. Random Data Compenstation

As we introduced before, each node selects some recipients randomly in ESSH. So, some nodes might receive no VDG message in a round. Furthermore, network issues such as node failures or message losses might also cause VDG message missing. We propose a simple random data compensation scheme. If node *i* doesn't receive any VDG message in round *r*, *i* selects *t* nodes and sends a V-data Request message to each of them. Once a node receives a V-data Request message, it sends a VDG message back to *i*.

IV. PERFORMANCE ANALYSIS

ESSH has low computation cost using hash function and symmetric cryptography. Therefore, we only analyze the communication cost and security performance for ESSH in this section.

A. Communication Cost

The communication cost is the total number of transmissions of a process. The communication cost of ESSH includes two parts.

1) In the random data receiving/forwarding phase, each node sends a VDG message to each of t recipients. The communication cost is

$$nt\bar{d} \qquad (2)$$

where n is the total number of nodes in the network and $\bar{d}$ denotes the average shortest hop between any two nodes.

2) In the random data compensation phase, if a node receives no VDG message, it sends t V-data Request messages to t randomly chosen nodes. Hence, the communication cost is $2npt\bar{d}$, where p is the probability that a node doesn't receive a VDG message. For arbitrary two nodes, say i and j ($j \neq i$), j receives a VDG message from i with probability $t/(n\text{-}1)$ in round r. So, j cannot receive a VDG message from i with probability 1- $t/(n\text{-}1)$. Then, the probability that j cannot receive a VDG message from any node in round r is $[1-t/(n-1)]^{(n-1)}$. Thus, the communication cost in the random data compensation phase is

$$C_{ESSH}(t)=2nt\bar{d}[1-t/(n-1)]^{(n-1)} \qquad (3)$$

Combining Eq.(2) and Eq.(3), we have that the communication cost for ESSH is $C_{ESSH}(t)=nt\bar{d}+2nt\bar{d}[1-t/(n-1)]^{(n-1)}$.

In DISH, each node obtains t random V-data by sending t data request message. Thus the communication cost is $C_{DISH}(t)=2nt\,\bar{d}$. Compared with DISH, the communication cost of ESSH is reduced by

$$\begin{aligned}&[C_{DISH}(t)\text{-}\ C_{ESSH}(t)]/C_{DISH}(t)\\&=\{2nt\bar{d}\text{-}nt\bar{d}\text{-}2nt\bar{d}[1-t/(n-1)]^{(n-1)}\}/2nt\bar{d}\\&=0.5\text{-}[1-t/(n-1)]^{(n-1)}\end{aligned} \qquad (4)$$

Let $f(x)=[1-t/(x-1)]^{(x-1)}$, the derivative of $f(x)$ with respect to x is

$$f'(x)=f(x)\{ln[1-t/(x-1)]+t/(x-t-1)\} \qquad (5)$$

Let $z=t/(x\text{-}1)$ and $z\in(0,1)$, then we have that

$$ln[1-t/(x-1)]+t/(x-t-1)=ln(1\text{-}z)+z/(1\text{-}z) \qquad (6)$$

Let

$$g(z)=ln(1\text{-}z)+z/(1\text{-}z) \qquad (7)$$

and the derivative of $g(z)$ with respect to z is $g'(z)=z/(1-z)^2>0$. It is obvious that $g(z)$ is monotonically increasing. As $z>0$, we thus have $g(z)>g(0)=0$. Combining Eq. (5), (6) and (7), we obtain $f'(x)>0$. Therefore, $f(x)$ is monotonically increasing too and we have

$$\begin{aligned}f(n)&=[1-t/(n-1)]^{(n-1)}\\&<\lim_{n\to+\infty}[1-t/(n-1)]^{(n-1)}\\&=\lim_{n\to+\infty}[1+\frac{1}{(1-n)/t}]^{[(1-n)/t](-t)}\end{aligned} \qquad (8)$$

Let

$$y=(1-n)/t \qquad (9)$$

Combining Eq. (8) and Eq. (9) gives

$$\begin{aligned}f(n)&=[1-t/(n-1)]^{(n-1)}\\&<\lim_{n\to+\infty}[1+\frac{1}{y}]^{y(-t)}=e^{-t}\end{aligned} \qquad (10)$$

according to the exponential limit equation $\lim_{n\to\infty}[1+1/n]^n=e$. Then, combining Eq.(4) and (10), we have

$$[C_{DISH}(t)\text{-}\ C_{ESSH}(t)]/C_{DISH}(t)>0.5\text{-}e^{-t} \qquad (11)$$

Different from DISH, nodes in POSH don't have to send extra data request message. Thus the communication cost for POSH is $C_{POSH}(t)=nt\,\bar{d}$. Compared with POSH, the communication cost of ESSH increases

$$\begin{aligned}&[C_{ESSH}(t)\text{-}C_{POSH}(t)]/C_{POSH}(t)\\&=\{nt\bar{d}+2nt\bar{d}[1-t/(n-1)]^{(n-1)}\text{-}nt\bar{d}\}/nt\bar{d}\\&=2[1-t/(n-1)]^{(n-1)}<2e^{-t}\end{aligned} \qquad (12)$$

According to Eq. (11), we conclude that the communication cost of ESSH decreases at least 0.5-e^{-t} (nearly 50%) compared with DISH. Meanwhile, compared with POSH, the communication cost of ESSH increases only $2e^{-t}$ by Eq. (12). If $t<6$, many sensors do not receive any random data in POSH and hence the node self-healing ability is not good [2]. Therefore, we set t=6 as POSH does. When t=6, the communication cost of ESSH decreases 49.75% compared with DISH and increases only 0.5% compared with POSH. Therefore, ESSH incurs small communication cost.

B. Security Analysis

In this section, we will analyze the security performance of ESSH.

***Theorem* 1**: Node self-healing ability doesn't change with the growth of attack rounds in ESSH.

Proof: For arbitrary two nodes, say i and j ($j\neq i$), j receives a VDG message from i with probability $t/(n\text{-}1)$ in round r. So, j cannot receive a VDG message from i with probability 1- $t/(n\text{-}1)$. Then, the probability that j receives t' VDG messages in round r is

$$P_{VDG}(t')=C_{n-1}^{t'}[t/(n-1)]^{t'}[1-t/(n-1)]^{n-t'-1} \qquad (13)$$

If j has ever been compromised, j can recover on condition that j receives at least one secure V-data. Let $p_{health}(t')$ denote the probability that j receives at least one secure V-data from t' VDG messages. Then, according to Def. 4, we have that

$$P_r=P_{VDG}(0)p_{health}(t)+\sum_{t'=1}^{n-1}P_{VDG}(t')\,p_{health}(t') \qquad (14)$$

As each V-data is generated by all nodes on the route $R_{i,j}$ from i to j, the V-data is unsecure only if the last node in $R_{i,j}$ is controlled by the attacker currently. The probability that the last node in $R_{i,j}$ is under control by an attacker is g/n. Thus a V-data is unsecure with probability g/n. We thus have

$$p_{health}(t')=1-(g/n)^{t'} \qquad (15)$$

Combining Eq. (13), (14) and (15) gives

$$\begin{aligned}P_r=&[1-t/(n-1)]^{n-1}[1-(g/n)^t]\\&+\sum_{t'=1}^{n-1}C_{n-1}^{t'}[t/(n-1)]^{t'}[1-t/(n-1)]^{n-t'-1}[1-\\&(g/n)^{t'}]\end{aligned} \qquad (16)$$

Eq. (16) shows that P_r is not a function of r. We thus conclude that node self-healing ability in ESSH doesn't change with varying r in ESSH. □

In POSH/DISH, once a node has been compromised, it cannot provide secure V-data. When $1\leq r<\lceil n/g\rceil$, the number of nodes in Healthy state is $H(r)=n\text{-}gr$. When $r\geq\lceil n/g\rceil$, all nodes in the network have been attacked and hence $H(r)$=0. In DISH, each node receives t V-data in a round. A compromised node j transits to Sub-healthy state with probability

$$p_r=\begin{cases}1-(gr/n)^t & 1\leq r<\lceil n/g\rceil\\ 0 & r\geq\lceil n/g\rceil\end{cases} \qquad (17)$$

As for POSH, the probability that each node receives *t*' (*t*'≥0) V-data is $P_{V-\text{data}}(t^{'})$=$C_{n-1}^{t^{'}}[t/(n-1)]^{t^{'}}[1-t/(n-1)]^{n-t^{'}-1}$. A compromised node transits to Sub-healthy state if it receives at least one secure V-data. Hence, the node self-healing probability is

$$p_r=\begin{cases}1-\sum_{t^{'}=1}^{n-1}P_{V-\text{data}}(t^{'})\,(gr/n)^{t^{'}} & 1\le r<\lceil n/g\rceil \\ 0 & r\ge\lceil n/g\rceil\end{cases} \qquad (18)$$

We use the same parameters as in POSH [2]: *n*=400, *t*=6 and *g*=80, and by Eq. (14), we have

$$P_r=P_{VDG}(0)p_{health}(t)+\sum_{t^{'}=1}^{n-1}P_{VDG}(t^{'})\,p_{health}(t')$$
$$>P_{VDG}(0)p_{health}(6)+\sum_{t^{'}=1}^{399}P_{VDG}(t^{'})\,p_{health}(t^{'})$$
$$>0.9995$$

Therefore, compromised nodes in ESSH can transit to Healthy state with a high probability. Fig. 2 shows the node self-healing ability of ESSH, DISH and POSH with the increase of attack rounds. It can be seen from Fig. 2 that ESSH provides high and sustainable node self-healing ability regardless of *r*.

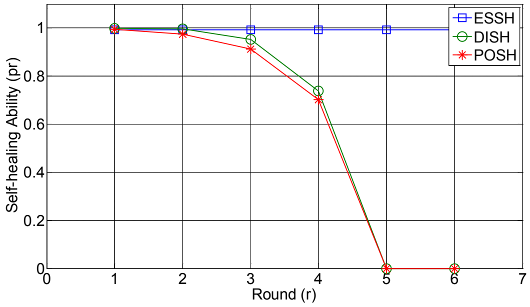

Fig.2: Self-healing probability comparison

V. SIMULATION RESULTS

We evaluate the performance of ESSH by an event-driven sensor network simulator written in C++. For uniform sensor deployment, we divide the monitored area into small grids and place one node in each grid. For fair comparison, our simulation uses the same setting as POSH [2]: a sensor network of 400 nodes with *g*=80.

Fig. 3 plots the communication cost of ESSH, POSH and DISH for varying *t*. As shown in Fig. 3, for ESSH, POSH and DISH the communication cost increases with the increase of *t*. The communication cost of ESSH is very close to that of POSH. This is consistent with our analysis in section IV. Fig. 3 also shows that compared with DISH ESSH reduces the communication cost by 50%.

In Fig. 4, we compare the number of nodes that receive at least one secure V-data under the three protocols. We can see that at the beginning the numbers of nodes in the three protocols are very close. However, as the attack round increases, the number of nodes in both POSH and DISH decreases significantly. After five rounds, nodes in both POSH and DISH cannot receive any secure V-data. This is because all nodes have been attacked after five rounds and no nodes in the network can provide secure V-data. On the other hand, we observe that almost all nodes in ESSH can receive secure V-data regardless of the attack round.

Fig. 5 shows that ESSH performs much better than POSH and DISH, in term of the number of self-healing nodes each round. It can be seen that the increase of attack rounds has a significant impact on POSH and DISH: The number of self-healing nodes quickly decreases with the increase of attack rounds. After five rounds, no nodes in both POSH and DISH can recover. On the other hand, the number of self-healing nodes in ESSH doesn't change much.

Fig. 6 shows the number of nodes in Healthy state. Note that for POSH and DISH, nodes in either Healthy or Sub-healthy state are considered as healthy nodes in our experiment. Fig. 6 shows that with the increase of attack rounds, the number of healthy nodes in POSH and DISH decreases dramatically, and after ten rounds there is no healthy node in the network. In contrast, the curve for ESSH is almost flat and the number of healthy nodes in ESSH is always more than that in POSH and DISH. This is no surprise since almost all nodes in ESSH can receive at least a secure V-data each round and thus most of nodes remain in healthy state or can return to healthy state.

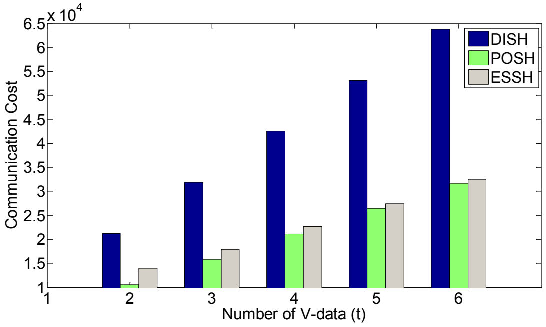

Fig.3: Comparison of communication cost

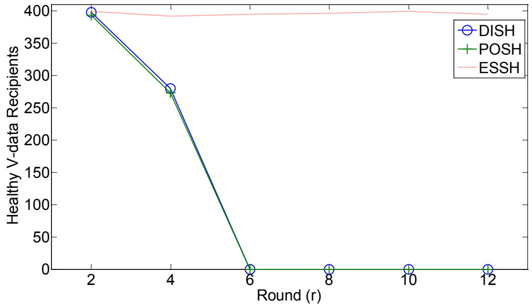

Fig.4: Number of nodes receiving secure V-data vs. attack rounds

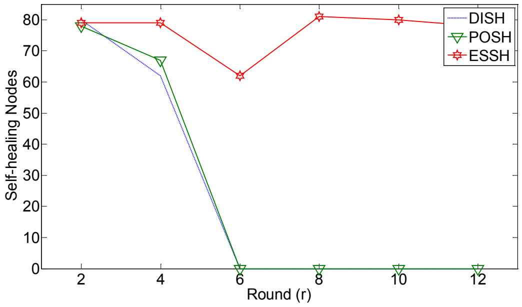

Fig.5: Number of self-healing nodes comparison

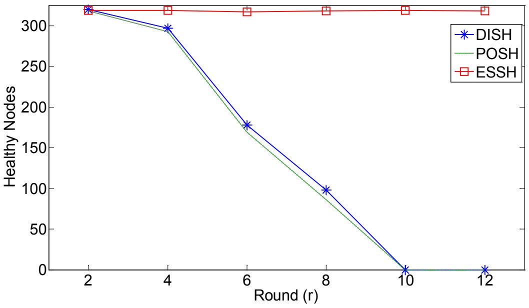

Fig.6: Number of healthy nodes vs. attack rounds

In summary, our simulation results show that ESSH provides efficient and sustainable node self-healing ability while incurring small overhead.

VI. Conclusion

In this paper, we studied the node compromise attack in UWSNs and we presented an efficient and sustainable self-healing (ESSH) protocol. We showed analytically and through simulation experiments that ESSH is efficient, sustainable and lightweight. Furthermore, ESSH copes well with reliable issues in UWSNs, e.g., message losses and sensor failures.

Acknowledgment

This research was supported in part by the China National Basic Research Program (973 Program) under grants 2011CB302605, the China National High Technology Research and Development Program (863 Program) under grant 2010AA012504 and 2011AA010705, the National Natural Science Foundation of China under grant 61073194, 61173145 and 61173144; and by the US National Science Foundation under grants CNS-0963578, CNS-1002974, CNS-1022552, and CNS-1065444, as well as the US Army Research Office under grant W911NF-08-1-0334.

References

[1] D. Ma and G. Tsudik, “Dish: Distributed self-healing,” in *Proc. of International Symposium on Stabilization, Safety, and Security of Distributed Systems* (*SSS*’08), 2008.

[2] R. Di Pietro, D. Ma, C. Soriente, and G. Tsudik, “POSH: Proactive cooperative self-healing in unattended wireless sensor networks,” in *Proc. of IEEE Symposium on Reliable Distributed Systems* (*SRDS*’08), 2008.

[3] D. Ma, C. Soriente and G. Tsudik. “New Adversary and New Threats: Security in Unattended Sensor Networks,” *IEEE Network*, vol. 23, no. 2, pp. 43-48, 2009.

[4] R. Latif, and M. Hussain, “Hardware-Based Random Number Generation in Wireless Sensor Networks(WSNs),” in *Proc. of ISA*, 2009.

[5] A. Suciu, D. Lebu and K. Marton, “Unpredictable Random Number Generator Based on Mobile Sensors,” in *Proc. of IEEE International Conference on Intelligent Computer Communication and Processing* (*ICCP*), 2011.

[6] M. Bellare and B. Yee, “Forward integrity for secure audit logs,” Technical Report, Computer Science and Engineering Department, University of San Diego, November, 1997.

[7] R. Dutta, Y. D. Wu, and S. Mukhopadhyay, “Constant storage selfhealing key distribution with revocation in wireless sensor network,” in *Proc. of IEEE International Conference on Communications* (*ICC*’07), 2007, pp. 1323–1328.

[8] M. Bellare and A. Palacio, “Protecting against key-exposure: strongly key-insulated encryption with optimal threshold,” *Appl. Algebra Eng. Commun. Comput*. vol. 16, no. 6, pp. 379–396, 2006.

[9] R. Di Pietro, G. Oligeri, C. Soriente and G. Tsudik, “Intrusion-Resilience in Mobile Unattended WSNs,” in *Proc. of IEEE INFOCOM*’10, 2010.

[10] W. Liu, R. Luo, and H. Yang, “Cryptography Overhead Evaluation and Analysis for Wireless Sensor Networks,” in *Proc. of International Conference on Communications and Mobile Computing*, 2009.