POSH: Proactive co-Operative Self-Healing in Unattended Wireless Sensor Networks

Roberto Di Pietro
Universitat Rovira i Virgili
roberto.dipietro@urv.cat

Di Ma
University of California, Irvine
dma1@ics.uci.edu

Claudio Soriente
University of California, Irvine
csorient@ics.uci.edu

Gene Tsudik
University of California, Irvine
gts@ics.uci.edu

Abstract

Unattended Wireless Sensor Networks (UWSNs) are composed of many small resource-constrained devices and operate autonomously, gathering data which is periodically collected by a visiting sink. Unattended mode of operation, deployment in hostile environments and value (or criticality) of collected data are some of the factors that complicate UWSN security.

This paper makes two contributions. First, it explores a new threat model involving a mobile adversary who periodically compromises and releases sensors aiming to maximize its advantage and overall knowledge of collected data. Second, it constructs a self-healing protocol that allows sensors to continuously and collectively recover from compromise. The proposed protocol is both effective and efficient, as supported by analytical and simulation results.

1 Introduction

In the last decade, sensor network security has rapidly gathered momentum and became a very popular topic of research. A wide range of sensor applications either already exist or are envisioned for the near future. One of the fundamental challenges stems from sensor resource limitations, e.g., storage, memory, bandwidth, computational ability, and, most importantly, battery power. This challenge applies to all aspects of sensor functionality, including security, which, in turn, complicates basic issues such as: key management, privacy, authentication, secure routing and intrusion detection.

The research community's original focus has been on wireless sensor networks (WSNs) operating in a real-time collection mode. In such WSNs, a trusted third party (collector or sink) is assumed to be always present. Sensors collect data – either periodically or on-demand – and transfer it to the sink. In this general setting, security issues have been thoroughly explored and a great body of literature has been accumulated.

Recently, a slightly different WSN paradigm has been identified. It envisions WSNs operating in unattended and hostile environments [8, 16]. In this unattended WSN (UWSN) setting, the sink visits the network with irregular and even unpredictable frequency. Consequently, each sensor must retain its data (measurements) for a considerable time.

Intervals between successive sink visits represent periods of vulnerability and incentivize attacks. While the sink is away, the adversary (ADV) can easily compromise a number of sensors, learn all possible information, and leave the network before the next sink visit. Specifically, we imagine an adversary who aims to learn or alter sensor data. It can simultaneously compromise up to a certain number of sensors (e.g., k out of n total). Once it compromises a set of sensors, it can choose to stay put or to move to a different set. We assume that there is a certain minimal time that the adversary needs to spend on all currently compromised sensors before it moves to the next set. To be realistic, we also assume that this time is much shorter than that between consecutive sink visits. Thus, the adversary has enough time intervals to migrate through different sets of sensors and undermine overall security of the UWSN.

This paper makes two main contributions. First, it explores an emerging adversarial model unique to UWSNs. Second, it presents a distributed protocol where unattended sensors, anticipating compromise, cooperate to maintain and restore security in the presence of a powerful mobile adversary. The new protocol is based on simple well-known cryptographic techniques. Results obtained from both analysis and simulations indicate that the proposed protocol, besides being efficient, is very effective in self-healing, despite

1060-9857/08 $25.00 © 2008 IEEE
DOI 10.1109/SRDS.2008.23

IEEE computer society

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 26, 2009 at 10:33 from IEEE Xplore. Restrictions apply.

the agility and best efforts of the mobile adversary.[1]

Organization: Next section surveys related work. Then, Section 3 presents our network assumptions, adversarial model and notation. Section 4 presents the cooperative self-healing protocol and Section 5 provides details on the adversarial behavior based on the network defense mechanism. Then, Section 6 constructs an analytical model and presents simulation results. Next, Section 7 compares our approach with alternatives and discusses certain limitations and potential improvements. The paper concludes with the summary and future work in Section 8.

2 Background and Related Work

A number of key distribution protocols have been proposed in order to bootstrap secure communication in WSNs, e.g., [10], [11], [5] and [4]. Most such protocols are secure and efficient. However, as shown in [14], sensor compromise is a real threat and the cost of tamper-resistance for commodity sensors remains prohibitively high. Clearly, key distribution alone is insufficient to maintain security if sensors can be easily compromised.

With respect to compromise of a given sensor, collected data can be partitioned in three categories, based on the time of collection: (1) before compromise, (2) during compromise, and (3) after compromise. If encryption and authentication keys are fixed, once a sensor is compromised, ADV can decrypt data in any category and create valid authentication tags for any fraudulent data. Clearly, nothing can be done to protect data in category (2), since, during compromise, ADV has full control of the sensor. Whereas, compromise of data in categories (1) and (3) is not unavoidable. We thus state two well-known security properties:

- **Forward secrecy:** ADV can not learn any keys used to decrypt and/or authenticate category (1) data.
- **Backward secrecy:** ADV can not learn any keys used to decrypt and/or authenticate category (3) data.

Forward secrecy is relatively easy to obtain through periodic key evolution [3]. Assuming that time is divided into fixed-length rounds, at the end of each round r, the next round key is computed as: $K^{r+1} = \mathcal{H}(K^r)$, where $\mathcal{H}(\circ)$ is a public one-way collision-resistnat hash function. (K^r is then securely erased.) This way, ADV who compromises a sensor at round $r+1$, learns K^{r+1} but can not compute K^r due to one-wayness of $\mathcal{H}(\circ)$. However, ADV can easily compute all $K^{r'}$ for $r' > r$ by using $\mathcal{H}(\circ)$; this negates backward secrecy.

[1] For example, in a 400-node UWSN where ADV simultaneously compromises up to $k = 50$ sensors, at least 298 always remain secure.

In contrast, it is much harder to attain backward secrecy. Doing so typically requires a trusted third party, in the form of either tamper-resistant hardware on each sensor, or constantly present intrusion-immune centralized key server.

Several cryptographic tools provide either only forward secrecy [1], [2] or both forward and backward secrecy [15], [9]. Unfortunately, they impose heavy overhead, unsuitable for resource-constrained sensors. In the context of WSNs, some protocols have been proposed that rely on key evolution to achieve forward secrecy (e.g., [7, 18]) but none provides backward secrecy. Peer cooperation for key computation has been proposed in [6], to provide security against a passive adversary. Naik, et al. [17] presented a technique to provide forward and backward secrecy for pairwise keys (shared by two sensors). The security of this technique rests on the assumption that both sensors can not be compromised at the same time.

A more recent result [16] suggests using key evolution and sensor cooperation to provide forward and backward secrecy, respectively. The proposed protocol (DISH) involves each sensor sharing an initial key with the sink. At any time, sensors are either *healthy* or *sick*. Healthy sensors are currently not compromised and their current keys are unknown to the adversary. Whereas, the adversary knows current keys of all *sick* sensors (due to current or past compromise of those sensors). Forward secrecy is obtained via key evolution. To gain backward secrecy, each sensor requests random contributions from a set of randomly selected peers and computes its next key based on its prior key and all received randomness. With this cooperative approach, a healthy sensor always remains healthy, as long it is not directly compromised; however, a sick sensor can become healthy if it receives randomness from healthy peers. DISH has been evaluated analytically and via simulations; one notable observation is that the ratio of sick to healthy sensors tends to stabilize after a few rounds. As we discuss below, although DISH exhibits certain important shortcomings, it serves as a stepping stone for our work in this paper.

3 Preliminaries

This section presents our network assumptions, adversarial model and notation.

3.1 System and Network assumptions

We assume a homogeneous UWSN where sensors are uniformly distributed over a certain geographical area. Salient details of our network model are as follows:

- **Periodic Data Collection:** Time is divided into equal and fixed collection rounds and each sensor collects a single data unit per round. Round synchronization can

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 26, 2009 at 10:33 from IEEE Xplore. Restrictions apply.

be implemented via any well-known technique, e.g., [13, 12].

- **Unattended Operation:** An itinerant sink periodically visits the UWSN to collect sensed data. There is a system-wide parameter – v – denoting the maximum number of collection rounds between successive sink visits.
- **Communication:** The UWSN is always connected and any two sensors can communicate either directly or through peers, according to the underlying routing protocol. However, messages can be lost and sensors can fail.
- **Storage:** Each sensor has enough storage for $O(v)$ data units.
- **Cryptographic Capabilities:** Each sensor can perform cryptographic hashing and symmetric key encryption. Also, each sensor has a Pseudo-Random Number Generator (PRNG) initialized with a unique secret seed shared with the sink.
- **Re-initialization:** Every time the sink visits the UWSN, it securely re-initializes secret seed values for all sensors and resets the round counter.

3.2 Adversarial Model

The envisaged adversary has the following features:

- **Goal:** ADV's main goal is to learn as many sensor secrets (keys or other keying material) as possible. These might be later used to decrypt encrypted data or to compute authentication tags for fraudulent data.
- **Compromise Power:** ADV can compromise at most $0 < k < \frac{n}{2}$ sensors at any round.[2]
- **Periodic Operation:** Time is divided into equal and fixed compromise rounds. At the end of each compromise round, ADV picks a subset of up to k sensors to compromise in the following round. (The latter may be the same or overlapping with the currently compromised subset). At the start of each round, the adversary atomically releases the subset from the previous round and compromises the new subset.
- **Compromise Round & Collection Round:** for ease of exposition and without loss of generality, we assume that the compromise and collection rounds have the same duration. Moreover, and also without loss of generality, we assume that they are synchronized, i.e., both types of rounds start and end at the same time.

[2]Note that if ADV controls at least $\frac{n}{2}$ sensors per round, in two rounds, it can trivially control the entire network.

- **Sensor Compromise:** ADV can read all storage/memory and listen to all communication of each compromised sensor.
- **Topology Knowledge:** ADV knows the entire topology of the UWSN.
- **Minimal Disruption:** ADV generally does not interfere with sensors' behavior. This in order to stay undetected for as long as possible. In particular, it does not delete, delay or introduce messages.
- **Defense Awareness:** ADV is fully aware of any scheme or algorithm that the UWSN uses to defend itself.

3.3 Notation

Our notation is reflected in Table 1. Some of the items are clarified later, in Section 4.

Table 1. Notation

n	total number of sensors
s_i	sensor i
r, r'	collection round indices
t	number of peers each sensor helps at any round
K_i^r	s_i' key at round r
$\mathcal{H}(\circ)$	one-way collision-resistant hash function
$\mathcal{S}$	set of all sensors in the UWSN
v	number of rounds between successive sink visits
R^r	set of red sensors at round r
Y^r	set of yellow sensors at round r
G^r	set of green sensors at round r
R_i^r	set of red contributions received by s_i at round r
Y_i^r	set of yellow contributions received by s_i at round r
G_i^r	set of green contributions received by s_i at round r
W_i^r	set of all contributions sent to s_i at round r
Z_i^r	set of all contributions received by s_i at round r
c_{ij}^r	$j-th$ contribution received by s_i at round r

4 POSH: Proactive co-Operative Self-Healing

In this section we present POSH (Proactive co-Operative Self-Healing) scheme. The main idea behind POSH is for each sensor to serve as a source of randomness for other sensors. The obvious observation is that: a sensor whose randomness is compromised (i.e., whose current key is known by ADV) can regain security and compute a new key unknown to ADV, if it obtains at least one "infusion" of secure randomness from a peer sensor whose randomness is not currently compromised. Put another way, since our

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 26, 2009 at 10:33 from IEEE Xplore. Restrictions apply.

goal is to allow a sensor to obtain backward secrecy (recover from prior compromise), an infusion of secure randomness achieves exactly that.

As mentioned in Section 3.1, when the sink leaves the UWSN, the round counter is reset to 1. All secrets are re-initialized and each sensor s_i shares a symmetric key K_i^1 with the sink. At this point, ADV knows no sensor keys whatsoever.

Starting in round 1, ADV breaks into k sensors and read all keys. Then, based on some strategy, it migrates to a different set of sensors, learning their keys. This process repeats until the sink's next visit. At any time, we identify three sets of sensors:

- *Red sensors (R^r)* are currently (in round r) controlled by ADV. We assume that it is always the case that $|R^r| = k$
- *Green sensors (G^r)* are those that have either never been compromised or regained their security through POSH. (Note that, if $G^r = \emptyset$, ADV remains in full control of the UWSN for all subsequent rounds.)
- *Yellow sensors (Y^r)* are those that have been compromised in some round $r' < r$ and their current keys are known to ADV.

Recall that we want a sensor to recover from past compromise and somehow compute a key unknown to ADV. In other words, backward secrecy is our primary goal (since, as mentioned earlier, forward secrecy is trivial to achieve). The gist of the POSH scheme is for sensors, at each round, to provide each other with contributions (derived from their PRNG-s). Each sensor, having received some such contributions, uses them together with its prior key (or keys) to compute a key for the next round.

Specifically, each sensor produces t pseudo-random values with its PRNG, and sends each value to a randomly selected recipient. We use the term *sponsor* to denote a sensor who contributes a value to a peer. The recipient uses all contributions from its *sponsors* as inputs to the one-way function used for key evolution. In more detail, to update its key at the end of round r, s_i computes:

$$K_i^{r+1} = \mathcal{H}(K_i^r || c_{i_1}^r || \ldots || c_{i_j}^r)$$

where $c_{i_j}^r$ is a contribution received during current round. Note that, since sponsors select contribution recipients at random, it is possible for a sensor to receive no contributions. This is a consequence of the probabilistic nature of POSH. (We discuss below how to pick t such that the probability of receiving no contributions is negligible.)

We also note that all contributions generated by red and yellow sensors are known to ADV and thus can not help any recipient in obtaining backward secrecy. (However, neither do they hurt since a green sensor can not become yellow due to any number of contributions from red or yellow sponsors.) On the other hand, contributions by green sensors are unknown to ADV. If a yellow sensor receives a single contribution from a green sensor, ADV can not learn the former's next key, i.e., backward secrecy is attained and the yellow sensor becomes green. Another feature is that a green sensor can not become yellow unless it first becomes red (is compromised). A yellow sensor s_i retains its color if ADV can compute K_i^{r+1}, which is the case as long as all inputs to $\mathcal{H}(\circ)$ are known, i.e.:

- K_i^r is known (since $s_i \in Y^r$)
- Each contribution $c_{i_j}^r$ received by s_i comes from a sponsor in $Y^r \cup R^r$

To summarize, we state the following features:

1. A red sensor at round r, remains red (if ADV does not leave) or becomes yellow (if ADV migrates) at round $r + 1$.
2. A yellow sensor at round r, becomes green at round $r + 1$ if it receives at least one green contribution. It becomes red if ADV chooses to compromise it, and it remains yellow otherwise.
3. A green sensor at round r becomes red at round $r + 1$ only if ADV compromises it directly. Otherwise, it remains green.

The state transition diagram of a single sensor is shown in Figure 1.

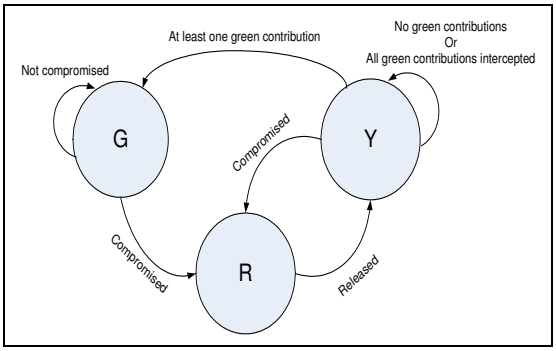


Figure 1. POSH Sensor State Transition Diagram.

5 Adversarial Strategies

The adversarial model summarized in Section 3.2 omits the criteria and strategy ADV uses to migrate between successive sets of compromised sensors. This is because ADV's

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 26, 2009 at 10:33 from IEEE Xplore. Restrictions apply.

behavior (given its goal) would be based on the specific defense mechanism employed by the UWSN. In case of POSH, ADV naturally strives to maximize the set of yellow sensors – Y^r.

We distinguish between two species of the adversary, based on its knowledge of the current set of yellow (or, equivalently, green) sensors. On one extreme, we can imagine ADV who is aware of the exact set of yellow sensors. On the other, we have ADV who assumes that all non-red sensors are potentially green (which may or may not be the case in POSH). We refer to the former as the Informed Adversary[3] or INF-ADV for short, and the latter – as the Round-Robin Adversary or RR-ADV for short.

INF-ADV migration strategy is to randomly select and compromise k sensors from the set G^r. We note that INF-ADV is very powerful. In fact, it might not be realistic in many UWSN settings employing POSH. However, we believe that it represents the upper-bound in adversarial capability. Thus, if POSH performs well in the presence of INF-ADV, it would perform considerably better with a more limited ADV.

RR-ADV migration strategy is to randomly select and compromise k sensors from the set $\mathcal{S}\backslash R^r$. However, to avoid backtracking, this adversary moves by first hopping through the entire population of sensors (which takes $\lceil \frac{n}{k} \rceil$ rounds) and then, starts over with the same pattern. Thus, RR-ADV represents the lower-bound in adversarial capability.

We claim that it is in the best interest of both INF-ADV and RR-ADV to exhibit the same behavior – and achieve the same results – for the first $\lceil \frac{n}{k} \rceil$ rounds: compromise a new (not previously "visited") set of k sensors per round.[4] Starting with round $\lceil \frac{n}{k} \rceil + 1$, INF-ADV occupies only sensors in G^r, while RR-ADV compromises those that are in $\mathcal{S}\backslash R^r$).

6 Analysis & Simulations

In this section we present analytical and experimental simulation results.

6.1 Analysis

To evaluate the healing rate provided by POSH, we analyze the number of green sensors at any round. We stress that INF-ADV is a very powerful adversary and thus its effectiveness corresponds to the lower-bound of security in POSH. Recall that INF-ADV knows the exact set of green sensors at all times and selects $R^{r+1} \subseteq G^r$.

A yellow sensor remains yellow if it receives no green contributions or if all received green contributions are intercepted by the adversary. Let $p_a = \frac{t}{n-1}$ be the probability of a given sensor being one of the sponsored sensors selected by a peer. Let E_1 be the event: *a sensor receives no green contributions* and E_2 be the event *all green contributions received by a sensor are intercepted by INF-ADV*. We then have:

$$Pr\{E_1\} = (1 - p_a)^{|G^r|}$$

Note that E_2 is influenced by the routing algorithm and UWSN topology. To make our analysis independent from these parameters, we define p as the probability that any contribution sent from a sponsor to a recipient is eavesdropped by INF-ADV.

The probability of a given sensor being chosen as a recipient of a contribution by s_i (out of $|G^r|$ green sensors) is:

$$p(i, |G^r|) = \binom{|G^r|}{i} p_a^i (1 - p_a)^{|G^r|-i}$$

Hence, we have:

$$Pr\{E_2\} = \sum_{i=1}^{|G^r|} p(i, |G^r|) p^i$$

and the probability of a yellow sensor not becoming green can be expressed as the union of E_1 and E_2, i.e.:

$$\begin{aligned} Pr_y &= Pr\{E_1 \vee E_2\} \leq Pr\{E_1\} + Pr\{E_2\} \\ &= (1 - p_a)^{|G^r|} + \sum_{i=1}^{|G^r|} p(i, |G^r|) p^i \end{aligned}$$

Thus, the expected number of green sensors at round r is:

$$E[|G^{r+1}|] \geq |G^r| + (1 - Pr_y)|Y^r| - k \qquad (1)$$

We now plot Equation 1 for a set of sensitive parameters. For all plots in this paper, we consider a UWSN composed of 400 sensors. Figure 2(a) shows the expected number of green sensors when INF-ADV eavesdrops on 20% of contribution messages in each round. For $\frac{k}{n} < \frac{123}{400} = 0.3075$, the number of green sensors decreases for the first few rounds and then remains stable. Indeed, at round 1 there are $n - k$ green sensors. At any round $r \geq 2$, there are always $2k + w$ compromised sensors: k red (currently occupied), k yellow which were red in the previous round, as well as w – yellow which have been compromised in any round between 1 and $r - 2$ and have not been *healed* yet. As the plot shows, there are always enough green sensors, and, therefore, enough $(t * |G^r|)$ green contributions. This limits w, regardless of the INF-ADV's efforts.

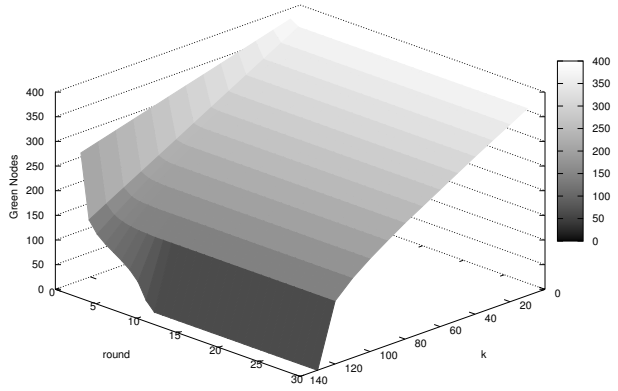
Note that, if $\frac{k}{n} >= 0.3075$, the number of green sensors becomes insufficient to thwart INF-ADV ; consequently, it eventually winds up in control of the entire UWSN.

[3] Latin: "Adversarius Sapiens"

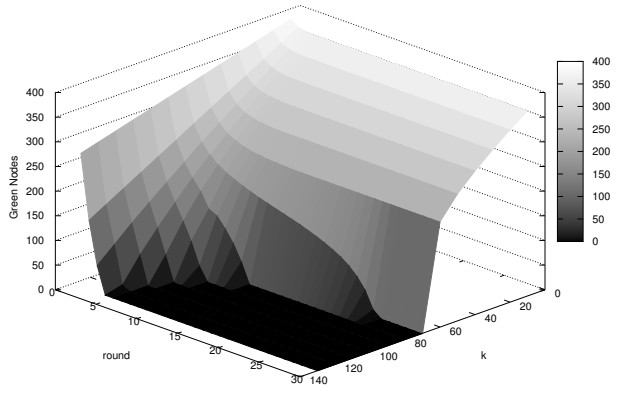
[4] At k sensors per round, it takes ADV $\lceil \frac{n}{k} \rceil$ rounds to compromise each sensor once.

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 26, 2009 at 10:33 from IEEE Xplore. Restrictions apply.

Figure 2(b) shows the same UWSN facing INF-ADV who eavesdrops on 80% of the traffic. As expected, the increase in eavesdropping rate causes the threshold value for $\frac{k}{n}$ to drop to $\frac{66}{400}$. That is, if $k > 66$, INF-ADV can compromise all sensors after a few rounds.



(a) $n = 400, t = 6, p = 0.2$



(b) $n = 400, t = 6, p = 0.8$

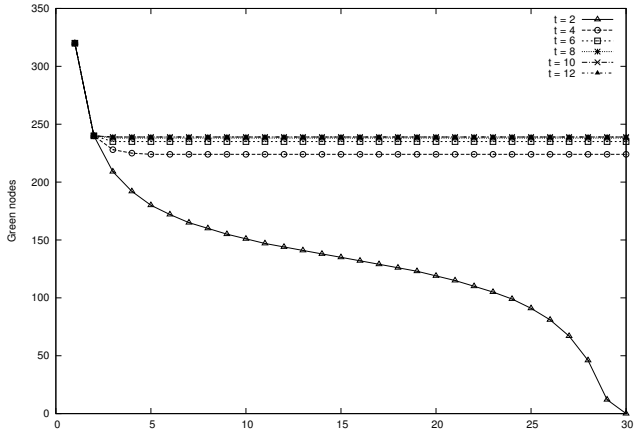
Figure 2. Analysis of INF-ADV**.**

For the previous graphs we set the number of contributions made by each sensor $t = 6$. From a straight-forward application of the well-known *Coupons Collector's Problem*, this value of t guarantees (for a network of 400 sensors) that each sensor is helped by at least one peer, with overwhelming probability.

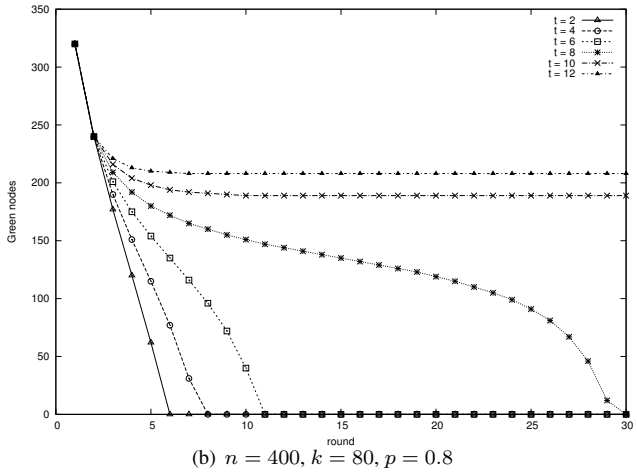
The value of t also determines the communication overhead of POSH, since the grand total of $t * n$ messages traverse the UWSN at the end of each round. We observe that, provided that $p < 1$ and $k < \frac{n}{2}$, it is in principle possible to come up with a sufficiently high value of t such that each sensor is guaranteed to receives a green contribution not eavesdropped on by INF-ADV. This way, the magnitude of G^r would always remain $|G^r| = n - 2k$. However, this would also involve very high communication overhead.

As shown in Figure 3(a), if $\frac{k}{n}$ and p are below the threshold (i.e., it is never the case that $G^r = \emptyset$), using $t > 6$ offers little extra in terms of security. This is simply because the UWSN already contains close to the optimal number of green sensors ($|G^r| = n - 2k$). However, if $t < 6$, many sensors (especially yellow ones) do not receive any contributions and the number of green sensors eventually drops to zero.

In contrast, if $\frac{k}{n}$ and p are above the threshold, setting $t > 6$ dramatically improves performance. As shown if Figure 3(b), given $\frac{k}{n} = \frac{80}{400}$ and $p = 0.8$, if $t = 6$ INF-ADV eventually turns all sensors red or yellow. Whereas, if $t > 6$, $|G^r|$ decreases slowly ($t = 8$) yet eventually reaches zero, while for $t = 10$ or $t = 12$, $|G^r|$ reaches steady state with non-empty G_r. (We also observe the obvious: as t gets bigger, the size of G^r grows commensurately, up to $n-2k$.)



(a) $n = 400, k = 80, p = 0.2$



(b) $n = 400, k = 80, p = 0.8$

Figure 3. Analysis of the effect of t.

The above analysis leads us to conclude that, for a wide

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 26, 2009 at 10:33 from IEEE Xplore. Restrictions apply.

range of parameters, POSH is secure against an ADV who controls a considerable fraction of sensors and eavesdrops on a large fraction of inter-sensor traffic. Nevertheless, if adversarial capabilities increase beyond a certain threshold, the number of green sensors eventually shrinks to nought, i.e., ADV compromises backward secrecy of the entire UWSN.

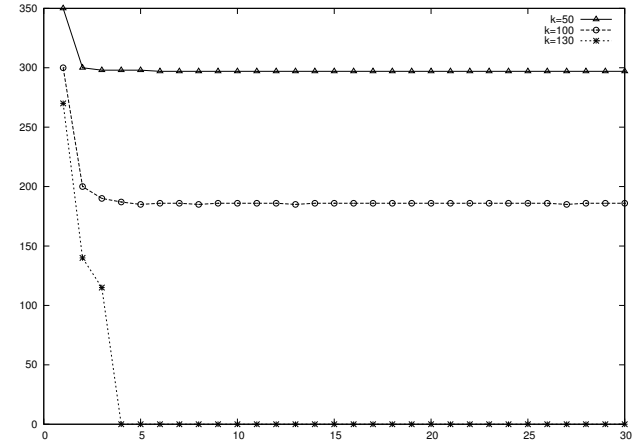
6.2 Simulations

To confirm and further validate analytical results, we developed a simulator and run numerous experiments. For each, we define the network parameters (n, t), ADV's power (k, p) and ran the simulator, until: (1) the UWSN has no more green sensors (ADV's success), or (2) $|G^r|$ reaches a steady state (UWSN's success).

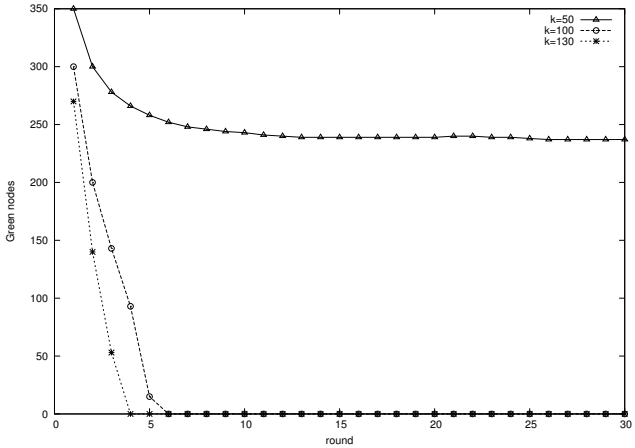
Figure 4 shows that simulation results match ther analytical counterparts. To assess how p influences network evolution, note that, in Figure 4(a) (for $p = 0.2$) ADV eventually compromises the whole UWSN for $k = 130$. However, if $p = 0.8$ (as in Figure 4(b)), ADV reaches the same goal with $k = 100$.

As mentioned in Section 6.1, the choice of t dramatically affects security. This is confirmed by simulations in Figure 5.

Figure 6 compares the behaviors of INF-ADV and RR-ADV. If the configuration of $\frac{k}{n}$ and p allows the UWSN to keep a constant number of green sensors, there is no appreciable difference between the two adversarial flavors. Whereas, if ADV has enough power to eventually subvert the whole network, INF-ADV reaches the goal faster (in fewer rounds). These results are interesting, since they show that POSH guarantees, for an appropriate choice of parameters, UWSN resilience to INF-ADV . Hence, there is no need to resort to more complex protocols. On the other hand, there is still some margin for improvement if ADV is powerful enough to eventually subvert the entire UWSN.



(a) $n = 400, t = 6, p = 0.2$



(b) $n = 400, t = 6, p = 0.8$

Figure 4. Simulation of INF-ADV**.**

7 Discussion

In this section we investigate how POSH compares to the earlier-proposed DISH scheme [16] and discuss some limitations of our proposal as well as ways to mitigate them.

7.1 Self-Healing: Push vs. Pull

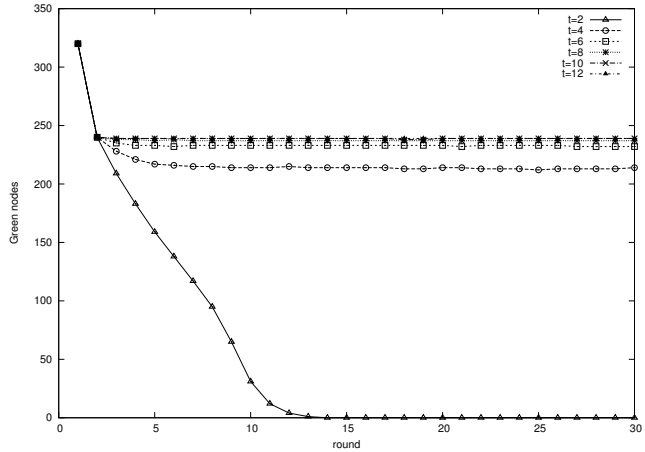
POSH has some notable differences with respect to the earlier DISH scheme [16]. We claim that POSH provides higher security against INF-ADV. From here on, we refer to the DISH scheme as the *pull* model because, in it, sensors explicitly request contributions from peers, while we refer to POSH as the *push* model, since it involves sponsors volunteering their contributions.

In both models, since local randomness is acquired through a PRNG, ADV can reconstruct the randomness of any yellow sensor. In particular, this means that ADV knows all contributions of all currently yellow sensors. This is where the similarity ends.

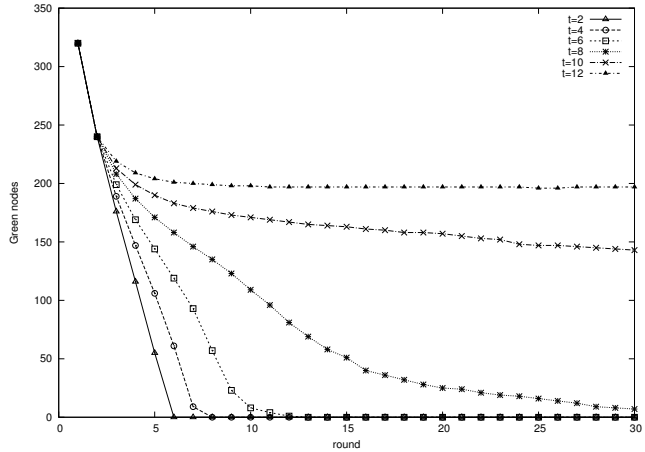
In the pull model, ADV knows all contributions sent by yellow sensors to other sensors; however, it might not know the recipients unless they are also yellow or the contribution message is intercepted. It also knows which sponsors each yellow sensor asked for a contribution. For example, at the end of round r, ADV can release a red s_i (let it become yellow) and continue monitoring s_i's key evolution. Since, before release, ADV copies the current state of s_i, it knows K_i^r as well as the set of s_i's future sponsors. ADV can then compromise all green sponsors (if any) and acquire their contributions to s_i. This allows ADV to compute K_i^{r+1}, i.e., s_i remains yellow.

In the push model, ADV knows all contributions sent by

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 26, 2009 at 10:33 from IEEE Xplore. Restrictions apply.



(a) $n = 400, k = 80, p = 0.2$



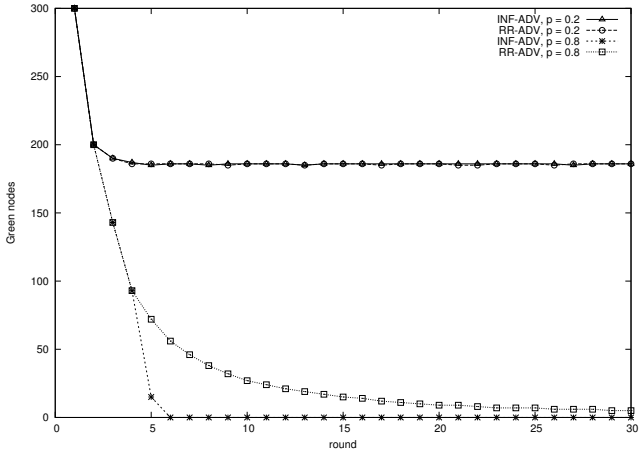
(b) $n = 400, k = 80, p = 0.8$

Figure 5. Simulation with different values of t.

yellow sensors to other sensors as well as all recipients of these contributions. The above example does not succeed with the push model, because ADV can not predict the behavior of green sensors. In particular, ADV does not know which t sensors a given green sensor will sponsor. Thus, once ADV releases a previously red s_i, it can not determine with certainty the set of sensors that might contribute to K_i^{r+1}. (Of course, ADV might determine that some red or yellow sensors will contribute to s_i. However, it can not be certain that no green sensor will also contribute.) Thus, to ensure that s_i remains yellow, ADV must either keep it red (i.e., continue occupying it) or control the area around it, in order to eavesdrop on all incoming messages.

Message overhead. The push model requires half of the messages required in the pull model. This is because the latter needs two messages – a request and a reply – for each contribution. The push model only involves a single contribution message. As shown in Figure 7, with the same number of messages per round, the push model performs better. In particular, if ADV can subvert all sensors in the pull model, with the push model the UWSN survives the same ADV with a high percentage of green sensors.

Figure 6. Comparison between INF-ADV and RR-ADV (for $n = 400$, $k = 100$, $t = 6$).



Contribution assurance. In the pull model, each sensor explicitly picks its own t sponsors. This guarantees that all sensors receive the same number of contributions.[5] In the push model, sponsored peers are picked at random and contributions might not be evenly spread. Thus, t must be tuned so that each sensor gets at least one contribution. For example, if $t = ln(n) + c$ the probability[6] that a sensor receives no contributions is e^{-c}.

7.2 Cooperation Drawbacks

We now identify some limitations of the proposed approach and explore ways of addressing them.

Unreliable Communication & Reliable Sensors. Since the sink knows the initial PRNG seed and initial key of each sensor, it can re-compute the state of each s_i (notably, K_i^r) at every round. This assertion holds as long as communication is perfect, i.e., all messages are delivered in a timely fashion and sensors do not fail. We now attempt to relax this ideal-world assumption.

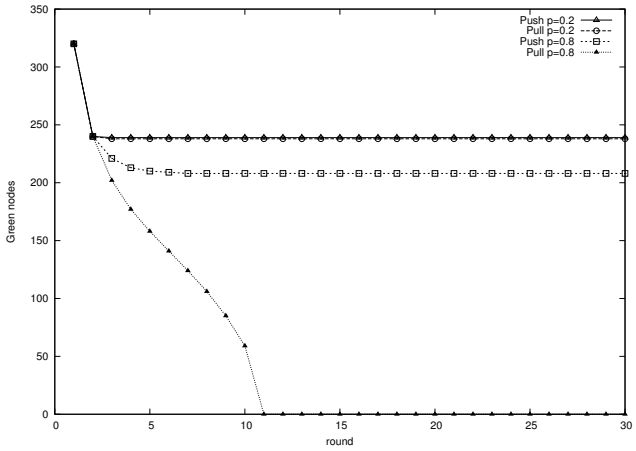
First, we assume that, while sensors still do not fail, messages are delivered with some probability $P \leq 1$. In other words, a contribution sent in round r to s_i might not be

[5] Assuming, of course, that all messages are delivered.

[6] Derived from the Coupon Collector's Problem.

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 26, 2009 at 10:33 from IEEE Xplore. Restrictions apply.

Figure 7. Comparison between the *Push* and the *Pull* models with respect to INF-ADV **($n = 400$ $k = 80$ $t = 6$).**



received by the latter. It is easy to see that this would desynchronize the key evolution process between s_i and the sink.

Let W_i^r be the set of contributions *sent* to s_i at round r and let $Z_i^r \subseteq W_i^r$ be the set of contributions *received* by s_i at the same round. Then, $K_i^{r+1} = \mathcal{H}(K_i^r||Z_i^r)$, while the sink would compute the same value as: $K_i^{r+1} = \mathcal{H}(K_i^r||W_i^r)$. Clearly, the sink would need extra information to correctly compute K_i^r.

A simple and effective solution, would be for s_i to store, for each round r, a set of ID-s: $\mathcal{ID}_i^r = \{j|s_j \ has \ a \ contribution \ in \ Z_i^r\}$. With this amendment, storage overhead for each sensor becomes linear, i.e., $O(v*t)$ where v is the maximum number of unattended collection rounds. Once the sink learns $\mathcal{ID}_i^r$, it can correctly K_i^r.

However, note that $\mathcal{ID}_i^r$ only provides information about the sponsors of s_i at round r. It does not give any information about the actual contributions. If, at round $r' > (r+1)$, ADV compromises s_i, it learns $\mathcal{ID}_i^r$, but can not compute K_i^{r+1}.

Unreliable Communication & Unreliable Sensors. If we now relax the assumption that sensors do not fail, learning $\mathcal{ID}_i^r$ for a given key update, might not be enough for the sink to re-compute K_i^r. For example, suppose that s_j sponsors s_i at round r, and then s_j fails at some later round. When the sink visits the UWSN, it can not reconstruct keys computed by s_j and, as a result, can not compute K_i^{r+1}. This demonstrates the inter-dependence among sensors' keys and reveals that both pull and push model are not robust in presence of sensor failures.

Addressing Sensor Failures. As discussed above, in the presence of unreliable communication, both DISH [16] and the proposed POSH schemes require each sensor to store additional information for each key update. If, in addition, sensor failures are allowed, both schemes are not robust due to sink's potential inability to recompute round keys.

One seemingly simple way to address both unreliable communication and unreliable sensors is forgo key synchronization with the sink. Instead, each sensor could simply store each contribution it receives in every round. (Or, it could store a function of all such contributions, in order to save space). Then, the sink would not have to re-compute each sensor's secure state for each round – it would simply obtain directly from sensors upon the next visit. Unfortunately, this approach is insecure since ADV can compromise a sensor and also learn these contributions.

A more viable and secure solution is to introduce public key cryptography. Assuming that the sink has a well-known permanent public key PK_{sink}, the PUSH scheme would operate in almost the same manner as described above. The only difference is that s_i would encrypt its round-specific key K_i^r under PK_{sink} using some suitable public key encryption technique (e.g., Elliptic Curve ElGamal or RSA). However, s_i would still encrypt data sensed in round r via conventional encryption under K_i^r. This simple modification appears to simultaneously solve all problems stemming from both unreliable communication and unreliable sensors. In particular, sensors no longer have to keep any lists of contributors' ID-s. Although extra public key encryption per round is certainly not negligible, it is most likely cheaper (as far as storage and eventual transmission to the sink) than each sensor keeping $O(t * v)$ storage for contributors' ID-s.

8 Conclusion

This paper explored a new an adversarial model geared for UWSNs. It also proposed a distributed self-healing scheme (POSH) based on proactive cooperation among sensors. Together with key evolution, POSH provides both forward and backward secrecy in the presence of a powerful mobile adversary. Both analytical and experimental results show that POSH is very effective. This paper also identified certain limitations of all approaches based on sensor cooperation and showed how to address them. Finally, it paved the way to future work on the general topic of security in UWSNs. Other UWSN-geared mobile adversary models, varying in both goals and power are possible and remain to be explored.

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 26, 2009 at 10:33 from IEEE Xplore. Restrictions apply.

9 Acknowledgments

Roberto Di Pietro is with the UNESCO Chair in Data Privacy, and the author is the solely responsible for the views expressed in this paper, which do not necessarily reflect the position of UNESCO nor commit that organization. Roberto Di Pietro is also with Università di Roma Tre, Dipartimento di Matematica.
This work was partly supported by The Spanish Ministry of Science and Education through projects TSI2007-65406-C03-01 "E-AEGIS" and CONSOLIDER CSD2007- 00004 "ARES", and by the Government of Catalonia under grant 2005 SGR 00446; the IST project SENSEI (Integrating the Physical with the Digital World of the Network of the Future, Grant Agreement Number: 215923). It was also supported in part by an award from the US Army Research Office (ARO) contract W911NF0410280 as well as a grant from the US Fulbright Foundation.

References

[1] M. Bellare and S. Miner. A forward-secure digital signature scheme. *Advances in Cryptology – CRYPTO'99*, 99:431–448, 1999.

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

[3] M. Bellare and B. S. Yee. Forward-security in private-key cryptography. In *CT-RSA*, pages 1–18, 2003.

[4] C. Castelluccia and A. Spognardi. RoK: a robust key predistribution protocol for multi-phase wireless sensor networks. In *IEEE Securecom'07*, 2007, to appear.

[5] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *S&P '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, page 197, Washington, DC, USA, 2003. IEEE Computer Society.

[6] M. Conti, R. Di Pietro, and L. Mancini. ECCE: enhanced cooperative channel establishment for secure pair-wise communication in wireless sensor networks. *Ad Hoc Networks*, 5(1):49–62, 2007.

[7] R. Di Pietro, L. Mancini, and S. Jajodia. Providing secrecy in key management protocols for large wireless sensors networks. *Ad Hoc Networks*, 1(4):455–468, 2003.

[8] R. Di Pietro, L. Mancini, C. Soriente, A. Spognardi, and G. Tsudik. Catch me (if you can): data survival in unattended sensor networks. In *IEEE PerCom'08*, pages 185–194, 2008.

[9] Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-insulated public key cryptosystems. In *EUROCRYPT'02*, pages 65–82, 2002.

[10] W. Du, J. Deng, Y. Han, P. Varshney, J. Katz, and A. Khalili. A pairwise key predistribution scheme for wireless sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(2):228–258, 2005.

[11] L. Eschenauer and V. Gligor. A key-management scheme for distributed sensor networks. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47, 2002.

[12] S. Ganeriwal, D. Ganesan, M. Hansen, M. Srivastava, and D. Estrin. Rate-adaptive time synchronization for long-lived sensor networks. *ACM SIGMETRICS Perform. Eval. Rev.*, 33(1):374–375, 2005.

[13] S. Ganeriwal, S. Čapkun, C. Han, and M. Srivastava. Secure time synchronization service for sensor networks. In *WiSe'05: 4th ACM workshop on Wireless security*, pages 97–106, 2005.

[14] C. Hartung, J. Balasalle, and R. Han. Node compromise in sensor networks: the need for secure systems. Department of Computer Science University of Colorado at Boulder, Tech. Rep CU-CS-990-05, 2005.

[15] G. Itkis. Intrusion-resilient signatures: generic constructions, or defeating strong adversary with minimal assumptions. In *SCN'02:Third Conference on Security in Communication Networks*, 2002.

[16] D. Ma and G. Tsudik. DISH: distributed self-healing (in unattended sensor networks). Cryptology ePrint Archive, Report 2008/158, 2008.

[17] V. Naik, A. Arora, S. Bapat, and M. Gouda. Whisper: local secret maintenance in sensor networks. *Workshop on Principles of Dependable Systems*, 2003.

[18] B. Przydatek, D. Song, and A. Perrig. SIA: secure information aggregation in sensor networks. In *ACM SenSys'03*, pages 255–265, 2003.

Authorized licensed use limited to: IEEE Xplore. Downloaded on February 26, 2009 at 10:33 from IEEE Xplore. Restrictions apply.