Self-Healing in Unattended Wireless Sensor Networks

ROBERTO DI PIETRO, Università di Roma Tre
DI MA, University of Michigan-Dearborn
CLAUDIO SORIENTE, Universidad Politécnica de Madrid
GENE TSUDIK, University of California, Irvine

Wireless sensor networks (WSNs) appeal to a wide range of applications that involve the monitoring of various physical phenomena. However, WSNs are subject to many threats. In particular, lack of pervasive tamper-resistant hardware results in sensors being easy targets for compromise. Having compromised a sensor, the adversary learns all the sensor secrets, allowing it to later encrypt/decrypt or authenticate messages on behalf of that sensor. This threat is particularly relevant in the novel unattended wireless sensor networks (UWSNs) scenario. UWSNs operate without constant supervision by a trusted sink. UWSN's unattended nature and increased exposure to attacks prompts the need for special techniques geared towards regaining security after being compromised.

In this article, we investigate cooperative self-healing in UWSNs and propose various techniques to allow unattended sensors to recover security after compromise. Our techniques provide seamless healing rates even against a very agile and powerful adversary. The effectiveness and viability of our proposed techniques are assessed by thorough analysis and supported by simulation results. Finally, we introduce some real-world issues affecting UWSN deployment and provide some solutions for them as well as a few open problems calling for further investigation.

1. INTRODUCTION

Wireless sensor networks (WSNs) are used for data collection in numerous application settings and scenarios, such as environmental monitoring, disaster relief, and homeland security. Security in WSNs presents a number of difficult and novel challenges due to the low cost and resource constraints of individual sensors.

The security research community's initial focus has been on WSNs operating in a real-time data collection mode: a trusted entity (usually called a sink) is assumed to be always present. Its presence allows nodes to submit measurements soon after sensing. While many WSNs operate in this model, some WSNs do not have the luxury of a constantly present sink. We refer to such networks as unattended WSNs or UWSNs [Di Pietro et al. 2008; Ma and Tsudik 2007]. In a UWSN, the sink periodically visits the network and collects data obtained and accumulated by sensors. Thus, a sensor cannot communicate with the sink at will. Instead, it collects data in situ and waits for the sink's next visit. A sensor's inability to off load data in real time exposes accumulated data to increased risks. This general model has been endorsed by national agencies [DARPA 2007] and private companies [Trident Systems 2010]. We believe it fits several network scenarios where the UWSN is deployed in a hostile environment and there is no online sink. One reason for not having an online sink might be the need to avoid having a single point of failure, as a sink is very likely a valuable attack target for the adversary. Another reason could be the network's large scale and/or remote location. In either case, there might be incentives for the sink to be itinerant, rather than fixed. For example, if the network is deployed along a national border to detect border crossing, the size of the monitored area precludes the possibility of having an online sink. The latter would be rather mounted on a vehicle of the border control corpse and would collect sensor measurements as it drives along the border.

Aside from the sink's constant availability, prior WSN security work typically assumed that there is an upper bound on the number of sensors that can be compromised during entire WSN deployment. Thus, most prior WSN security work focused on attack detection [Conti et al. 2008; 2009]. This is reasonable, since a constantly present sink can help to detect and react to attacks.

In a UWSN, the unattended nature of the network gives the adversary extra advantages. In particular, during sink absence, the adversary can roam the network and compromise sensors. Thus, adversarial models used in prior sensor network security cannot be used in evaluating security solutions for UWSNs. The powerful *mobile* adversary considered here, $\mu\mathcal{ADV}$, takes advantage of the unattended nature of the network: it compromises a number of sensors within a particular interval and moves on to another set of sensors in the next interval. We anticipate that the adversary might also compromise a number of sensors and stay put, but it would gain no advantage in doing so. A static adversary (i.e., the adversary that has been considered so far in WSN literature) would only learn measurements collected by the compromised sensors. Indeed, if intervals are much shorter than the time between successive sink visits and the adversary roams compromising different sets of sensors, it might subvert the entire network, gradually undermining security.

In this article, we focus on techniques to defend UWSNs against a *curious* mobile adversary whose goal is to learn data collected by sensors. We assume that the adversary does not have resources to set up its own network infrastructure or that corrupting sensors and reading their measurements is a more cost-effective solution [Luo et al. 2008]. When a curious mobile adversary compromises a sensor, it reads all memory and storage of that sensor and leaves no evidence of its presence. Once a sensor is compromised and the adversary learns its secrets, collected data—even if encrypted—become exposed.

If we assume that the adversary compromises a given sensor and maintains it under direct control for a certain length of time, data collected by that sensor can be partitioned into three categories based on the time of compromise: (1) before compromise, (2) during compromise, and (3) after compromise. Nothing can be done about the secrecy of data in category (2) since the adversary is fully in control. The challenge is in protecting the secrecy of data in categories (1) and (3), which requires, respectively, the following.

—*Forward secrecy*. The term *forward* means that category (1) data remains secret as time goes forward.
—*Backward secrecy*. The term *backward* means that category (3) data remains secret even though a compromise occurred before it was collected.

Contributions. This article investigates two cooperative self-healing strategies [Ma and Tsudik 2008; Di Pietro et al. 2008] that allow an UWSN to recover from compromise and maintain the secrecy of collected data. Because the cure comes from peer sensors, the network exhibits a self-healing property that emerges through collaboration of all nodes—something no individual node can provide. This distributed self-healing strategy provides a tunable probabilistic assurance of data secrecy across all nodes and compromise intervals. We completely characterize via analysis the features that capture this emergent property, that is, adversarial capability (number of nodes $\mu\mathcal{ADV}$ can compromise at the same time), amount of internode communication the UWSN can support, and number of data collection intervals between successive sink visits. We also show that, in this context, healing capabilities are subject to a 0-1 law. Results obtained from our analysis are supported by extensive simulations, showing that the proposed protocols are very effective in self-healing, despite the power of the mobile adversary. Finally, some issues related to UWSN deployment are addressed, while some open research problems calling for further investigation are introduced.

Organization. The article is organized as follows. We start surveying related work in Section 2 and provide an overview of existing techniques and point out why they cannot satisfy our security requirements in Section 3. We then present our network and adversarial model in Section 4 and show two self-healing schemes in Section 5. We analyze the effectiveness of the proposed schemes and present simulation results in Section 6. Next, we discuss certain limitations and potential improvements that pave the way for further research in the field in Section 7. The article concludes in Section 8.

2. RELATED WORK

Data secrecy is a fundamental security issue in sensor networks, and encryption is the standard way to achieve it [Hu and Sharma 2005]. Many research efforts have yielded techniques for establishing pairwise keys used to secure sensor-to-sensor and sink-to-sensor communication (e.g., [Eschenauer and Gligor 2002; Di Pietro et al. 2003; Du et al. 2004; Chan and Perrig 2005]). Local keys are updated periodically to mitigate the effect of sensor compromise. Mauw et al. [2006] proposed some techniques for providing forward-secure data authentication and confidentiality for node-to-sink communication.

Wang et al. [2009] limit the damages of secret-exposure in WSN using secret-sharing and erasure codes; their proposed scheme achieves secure and dependable data storage, allowing for dynamic integrity of sensed data. A work related to ours is Whisper [Naik et al. 2003], a protocol which provides both forward and backward secrecy for communication between pairs of sensors. However, Whisper's security relies on an unrealistic assumption that the adversary cannot compromise both sensors simultaneously. Also, every sensor must be equipped with a TRNG.

Recently, unattended sensors and sensor networks have become subjects of attention. The initial work [Ma et al. 2009] introduced the UWSN scenario, defined the mobile adversary, and discussed a number of challenges in the new scenario. Di Pietro et al. [2008] investigated simple techniques to counter attacks focused on erasing specific data. This work was later extended [Di Pietro et al. 2009a] to include the case where the adversary's goal is to indiscriminately erase all sensor data. The problem of data survivability in UWSN was also studied [Park and Shin 2005; Ruan et al. 2010; Ren et al. 2010, 2009; Vitali et al. 2011]. Another recent result [Di Pietro et al. 2009b] introduced simple cryptographic techniques for preventing the adversary from locating data that it aims to erase. Authentication in UWSN was studied in Di Pietro et al. [2009] and Yu et al. [2010]. Forward- and backward-secure data authentication in UWSNs was also explored [Ma 2008; Ma and Tsudik 2007; Di Pietro et al. 2009; Yavuz and Ning 2009]. Self-healing techniques for UWSNs where sensors are mobile have been proposed in Di Pietro et al. [2010a, 2010b]. Their protocols leverage node mobility to afford self-healing, while this article focuses on networks with static nodes.

Mobile adversaries have been studied in the cryptographic literature for a long time [Ostrovsky and Yung 1991]. An entire branch of cryptography, called *Proactive Cryptography*, has been developing cryptographic techniques that preserve security in the presence of such a mobile adversary [Frankel et al. 1997; Rabin 1998]. However, the goal of the cryptographic mobile adversary is to discover some system-wide secret (usually a decryption or a signature key) predistributed via secret-sharing techniques among the system components. Whereas, in a UWSN, the mobile adversary's goal is to read, erase, or modify data collected by unattended sensors. Consequently, research results in proactive cryptography do not apply to the problem at hand. Also, the resource-constrained capabilities of sensors makes the domain of UWSN very different from that in proactive cryptography, thus motivating radically different solutions.

The result most related to this article is that of Canetti and Herzberg [1994] in which a generic deterministic scheme is proposed that maintains secrecy in the presence of a mobile adversary. In this scheme, a node must communicate with all other nodes to update its state at each round. This might work for small wired networks, but due to communication overhead, the proposal would not scale to the envisaged UWSN size. Also, since sensor communication is wireless and broadcast in nature, eavesdropping is easy, which makes our analysis very different from that in Canetti and Herzberg [1994].

An adversarial model for WSN similar to the one considered in this article, was studied in Luo et al. [2008], where the adversary is *parasitic* and *unobtrusive*. That is, the adversary tries to learn sensor collected data but does not modify sensor code.

3. SIMPLE NON-COOPERATIVE SCHEMES

We now briefly examine a few noncooperative schemes to achieve forward and/or backward secrecy. More detailed descriptions of these schemes can be found in Ma and Tsudik [2008].

3.1. Simple Symmetric Key Scheme

A scheme based on symmetric cryptography assumes that each sensor s_i shares a key K_i^0 with the sink. At round r, s_i collects data d_i^r and encrypts it to produce $E_i^r = Enc(K_i^r, d_i^r, r, s_i)$.

It then computes the next round key as $K_i^{r+1} = F(K_i^r)$ where $F(\cdot)$ is a suitable pseudorandom function (PRF).[1] Finally, it securely deletes K_i^r and moves to the next round. During its next visit, the sink collects all ciphertexts produced by s_i and decrypts them to retrieve data. Since the sink knows K_i^0, it can easily recompute all keys used by s_i in each round.

In this simple scheme, forward secrecy is achieved through one-way key update. However, backward secrecy is lacking: once the adversary compromises s_i at round r and learns K_i^r, it can compute any future key $K_i^{r'}$ $(r' > r)$ by mimicking the key evolution process.

[1]In practice, a PRF can be constructed from a block cipher, such as AES (AES is a good pseudorandom permutation). Other even more practical constructions of PRFs deployed in standards use "good" MAC functions, such as HMAC [Bellare et al. 1996].

Table I. Notation

n	number of sensors in the network
k	number of compromised sensors at any round
s_i	sensor i
r, r'	collection round indices
t	degree of collaboration
K_i^r	s_i' key at round r
$F(\cdot)$	pseudorandom function
$H(\cdot)$	one-way collision-resistant hash function
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
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
$c_{i_j}^r$	jth contribution received by s_i at round r

3.2. Simple Public Key Scheme

In this scenario, the sink has a long-term public key, PK_{sink}, known to all sensors. At round r, sensor s_i collects data d_i^r and encrypts it to produce $E_i^r = Enc(PK_{sink}, R_i^r, d_i^r, r, s_i)$, where R_i^r is a one-time random value included in each randomized encryption operation, as specified in the OAEP+ quasi-standard [Shoup 2000]. Upon its next visit, the sink collects all ciphertexts produced by s_i and decrypts them with its private key SK_{sink}. Since sensors have no secrets of their own, no critical information is exposed upon compromise. Thus, the only way $\mu\mathcal{ADV}$ can determine cleartext data is by guessing and trying to encrypt it with the sink's public key, PK_{sink}. Since the value of sensor data can have a small range, defending against a guessing attack requires the use of randomized encryption.

On the one hand, if random numbers are obtained from a true random number generator (TRNG), both forward and backward secrecy can be trivially achieved. On the other hand, if random numbers are obtained from a pseudorandom number generator (PRNG), the resulting security is similar to the preceding simple symmetric key scheme, that is, forward secrecy is attained but backward secrecy is lacking.

3.3. Key-Insulated and Intrusion-Resilient Schemes

There are certain advanced cryptographic techniques that provide both forward and backward secrecy. They include key-insulated [Dodis et al. 2002] and intrusion-resilient [Dodis et al. 2003, 2004] encryption schemes. In both models, secret key update requires a separate secure entity, typically in the form of a remote trusted server or local tamper-resistant hardware. However, such schemes are unsuitable for UWSNs, since there is no trusted server and pervasive secure hardware is unrealistic.

3.4. Summary

Having reviewed simple intuitive approaches, we observe that except for the public key scheme used in conjunction with all sensors equipped with TRNGs, none of the schemes achieves both forward and backward secrecy. However, we believe that per-sensor TRNG is not viable for current and emerging sensor networks due to resource and cost constraints.

4. NETWORK AND ADVERSARY MODEL

This section presents our network and adversary model. Symbols used in the rest of the article are reflected in Table I.

4.1. Network Model

We assume a homogeneous UWSN with n sensors uniformly distributed over a fixed geographical area. The sink periodically visits the UWSN to collect data.

We define the time between two consecutive visits by the sink as an *unattended epoch*. Time within each unattended epoch is divided into fixed collection intervals, and each sensor collects a single data unit per round.[2] Round synchronization can be implemented via any well-known technique, for example, that of Ganeriwal et al. [2005].

There is a system-wide parameter, v, denoting the maximum number of collection rounds in an unattended epoch. We assume sensors have enough storage to accommodate $O(v)$ data units.[3]

The UWSN is always connected, and any two sensors can communicate either directly or via peers, according to the underlying routing protocol. However, messages can be lost and sensors can fail.

Each sensor can perform cryptographic hashing, symmetric key encryption, and, optionally, public key encryption (but not decryption).[4] Also, each sensor has a PRNG initialized with a unique secret seed shared with the sink.

Finally, every time the sink visits the UWSN, it empties all sensor storage, securely re-initializes secret seed values for all sensors, and resets the round counter.

We assume the adversary is not present during sink visits to the network. Since sensor secrets are refreshed, $\mu\mathcal{ADV}$ cannot retain control over sensor-across multiple unattended epochs; hence, our ultimate goal is intrusion-resilience within one unattended epoch.

4.2. Adversarial Model

$\mu\mathcal{ADV}$'s main goal is to learn as many sensor secrets (keys or other secret material) as possible. An adversary with similar goals was introduced in Luo et al. [2008]. To ease exposition, we assume that the adversarial round is the same as the sensor collection round. At the end of each round, $\mu\mathcal{ADV}$ picks a set of sensors to compromise next. At the start of the next round, it releases the currently compromised sensors and compromises the new set.

$\mu\mathcal{ADV}$ can simultaneously compromise at most k sensors ($k \ll n$) per round, similarly to the adversary in Luo et al. [2008] and Kong and Hong [2003]. However, $\mu\mathcal{ADV}$ has no restriction on the location of sensors to compromise, that is, compromised sensors might be not clustered. While a sensor is compromised, $\mu\mathcal{ADV}$ reads its storage/memory and listens to all incoming and outgoing communication. However, $\mu\mathcal{ADV}$ is not a global eavesdropper: it can only monitor traffic germane to currently compromised nodes.

We also stress that $\mu\mathcal{ADV}$ does not interfere with sensors' behavior, that is, we deal with an *unobtrusive*, as defined in Luo et al. [2008]. This is in order to stay undetected for as long as possible. Actually, any modification to the sensor code can be later discovered by the sink using techniques [Park and Shin 2005; Seshadri et al. 2004; Yang et al. 2007].

If the sink does not recognize malicious activity during its absence, the adversary can benefit from multiple attacks to the network over periods of sink absence.

[2]We use the term round or interval interchangeably.

[3]This assumption will be later relaxed as we trade off space requirement for improved robustness.

[4]In the context of resource-constrained sensor networks, we are interested in encryption schemes that allow for efficient encryption, while we might tolerate more expensive decryption operations at the sink. For example, the RSA public key encryption scheme allows for efficient encryption using small public exponents.

ALGORITHM 1: SEND

```
$\{R_{i_1}, \ldots, R_{i_t}\} \leftarrow_{\$}$ ;
$\{s_{i_1}, \ldots, s_{i_t}\} \leftarrow_{\$}$ ;
for ($j \leftarrow 1$ to $t$) do
    Send $R_{i_j}$ to $s_{i_j}$
end
```

ALGORITHM 2: RECEIVE

```
$S = \emptyset$ ;
while ($roundTimer$) do
    Receive $c_{i_j}$ from $s_{i_j}$ ;
    Add $c_{i_j}$ to $S$ ;
end
Sort $S$ ;
$K_i^{r+1} = F(K_i^r)$ ;
for ($j \leftarrow 1$ to $|S|$) do
    $K_i^{r+1} = F(K_i^{r+1} || c_{i_j})$ ;
end
```

5. SELF-HEALING STRATEGIES

We now present two cooperative self-healing schemes that provide forward and backward secrecy.

As discussed in Section 3, since forward secrecy can be easily achieved through key evolution, our main goal is backward secrecy within one unattended epoch. In other words, we need a previously compromised sensor to compute a new (next-round) key unknown to $\mu\mathcal{ADV}$ and thus regain security.

The main idea behind our techniques is for each sensor to serve as a source of randomness for other sensors. The key observation is that a sensor not currently occupied by $\mu\mathcal{ADV}$ but with a compromised state[5] can regain security by computing a new secure state if it obtains at least one "infusion" of secure randomness from a peer sensor whose secret state is not compromised. Since our goal is to allow a sensor to obtain backward secrecy (recover from prior compromise), an infusion of secure randomness achieves exactly this goal.

Using our cooperative self-healing approach, nodes can either explicitly solicit random contributions from others or volunteer their own random contributions to peers. Based on this difference, we propose two concrete schemes: *PUSH* [Di Pietro et al. 2008] and *PULL* [Ma and Tsudik 2008]. In the former, each node voluntarily supplies a random set of peers with its random contributions. Whereas, in the latter, each node asks for contributions from a random set of peers. Although similar, these two schemes are inherently different in regards to healing rates and behaviors, as described in the next sections.

5.1. *PUSH*

At each round, every sensor runs Algorithm 1 and Algorithm 2. In SEND, based on its current PRNG state, each sensor picks a random set of t peers and, for each of them, generates and sends a pseudorandom value. In RECEIVE, each sensor receives contributions from others and uses these contributions (together with its current key)

[5]In other words, its current key is known to $\mu\mathcal{ADV}$.

ALGORITHM 3: RECEIVE

```
$S = \{s_{i_1}, \ldots, s_{i_t}\} \leftarrow_\$$ ;
for ($j \leftarrow t$ to $|S|$) do
    Send $REQ$ to $s_{i_j}$ ;
end
$S = \emptyset$ ;
while ($roundTimer$) do
    Receive $c_{i_j}$ from $s_j$ ;
    Add $c_{i_j}$ to $S$ ;
end
Sort $S$ ;
$K_i^{r+1} = F(K_i^r)$ ;
for ($j \leftarrow 1$ to $|S|$) do
    $K_i^{r+1} = F(K_i^{r+1} || c_{i_j})$ ;
end
```

ALGORITHM 4: SEND

```
while ($roundTimer$) do
    Receive $REQ$ from $s_j$ ;
    $c_{ji} \leftarrow_\$$ ;
    Send $c_{ji}$ to $s_j$ ;
end
```

as inputs to the key evolution function used to compute the next round's key. In both algorithms, we denote with $\leftarrow_\$$ an invocation of a sensor's PRNG.

5.2. *PULL*

In this protocol, each sensor runs Algorithm 3 and Algorithm 4. In SEND, a sensor waits for requests for randomness and replies, if asked. In RECEIVE, a sensor selects t random peers and asks each of them for a random contribution. Having received all contributions, a sensor uses them (together with its current key) as inputs to the key evolution function used to compute the next round's key. In both algorithms we denote with $\leftarrow_\$$ an invocation of a sensor's PRNG.

6. ANALYSIS AND SIMULATION RESULTS

In this section, we analyze the effectiveness of the proposed cooperative self-healing schemes, that is, the number of healed sensors they can afford. We also develope a simulator to further confirm and validate analytical results. For all configuration of parameters taken into account, the difference between analytical and simulated data was negligible.[6] Hence, for the sake of readability, we will only provide figures of our simulations. Before presenting our analysis and simulation results, we state our system abstraction and refine the adversarial strategy.

6.1. Sensor States—Coloring

To simplify our analysis, at each round, we partition sensors into three distinct groups. A sensor can be red, yellow, or green, as defined next.

[6]In 98% of our experiments , the difference between the number of healed nodes and the outcome of the analysis was less than 1. The largest difference we experienced was 4, that is, less than 1% of the network size.

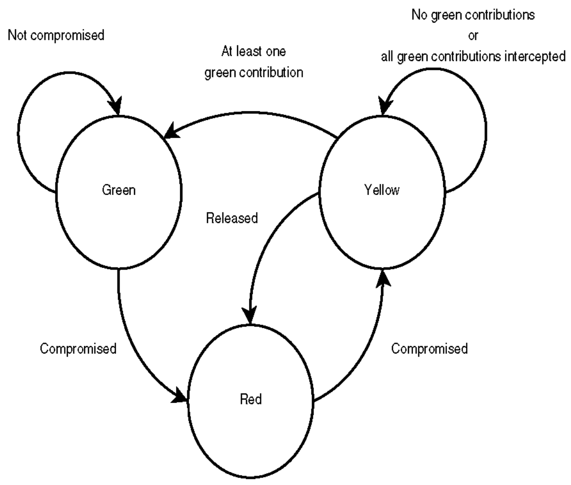

Not compromised
At least one green contribution
No green contributions or all green contributions intercepted
Green
Yellow
Released
Compromised
Compromised
Red

Fig. 1. Sensor state transition diagram.

—*Red sensors (R^r).* A sensor is red if it is currently (in round r) controlled by $\mu\mathcal{ADV}$. This means that $\mu\mathcal{ADV}$ knows its current key and eavesdrops on all incident communication. $\mu\mathcal{ADV}$ also learns the next-round key, since it is computed at the end of the current round.

—*Yellow sensors (Y^r).* A sensor is yellow if it has been compromised in round $r' < r$ and subsequently released by $\mu\mathcal{ADV}$ later. However, its current key is still known by $\mu\mathcal{ADV}$ (but $\mu\mathcal{ADV}$ cannot eavesdrop on it.)

—*Green sensors (G^r).* A sensor is green if its current key is unknown to $\mu\mathcal{ADV}$. Either it has never been compromised or it was "healed" through our protocols.

Given this coloring scheme, it is easy to see that (1) a green sensor remains green until $\mu\mathcal{ADV}$ compromises it; (2) a red sensor cannot become green without becoming yellow first; and (3) a yellow sensor can become green if and only if it receives at least one contribution from a green sensor that was not eavesdropped upon by $\mu\mathcal{ADV}$. The state transition diagram in Figure 1 captures these axioms and illustrates how sensors change colors.

Since knowledge of sensor keys allows $\mu\mathcal{ADV}$ to decrypt data previously encrypted under these keys, $\mu\mathcal{ADV}$'s goal is to maximize the number of both red and yellow sensors. To this end, we analyze the effectiveness of the proposed protocols in terms of the number of green sensors present at each round.

Evolutionary model approximation. This model just introduced completely depicts the possible evolution among *states* (colors). In the following, we will provide an analytical representation. To this goal, we need to introduce some (limited) approximations.

(1) *Inter-round dependency.* Note that the exposed model introduces a dependency of the actual state of the system (at round r) on the previous state of the system (at round $r-1$). Consider, for instance, the number of green sensors at the beginning of round r. This number depends on the number of green nodes at the beginning of round $r-1$ and the ones remaining at the end of round $r-1$. To simplify the analysis, we will approximate the number of green sensors at the beginning of round r with its expected value nP_G^{r-1}.

(2) *Absorbing states.* No single state in Figure 1 is an absorbing one. However, note that if the number of green sensors drops to zero, no "healing randomness" will be spread throughout the network, and all sensor will remain either yellow or red. Hence, if the number of green sensors drops below $\mu\mathcal{ADV}$'s compromising capability ($G^r \leq k$), next round will lead the system to be composed of red and yellow sensors only. Since this last step in the network evolution is of little interest, in the following analysis, we assume the network enjoys a number of green sensors higher than k—this hypothesis being verified by all rounds except the last one.

6.2. Adversarial Strategies

The adversarial model summarized in Section 4.2 omits both the criteria and the strategies $\mu\mathcal{ADV}$ uses to migrate between successive sets of compromised sensors. This is because $\mu\mathcal{ADV}$'s behavior (given its goal) would be based on the specific defense mechanism employed by the UWSN.

While $\mu\mathcal{ADV}$ can arbitrarily choose the set of sensors to compromise, its best strategy is to compromise as many green nodes as possible at any round. However, since $\mu\mathcal{ADV}$ is not a global eavesdropper, it cannot determine current status of all nodes. Nevertheless, $\mu\mathcal{ADV}$ knows the identities of nodes that have ever been compromised. Thus, moving in a round-robin fashion, it can keep compromising green sensors for the first $\lceil \frac{n}{k} \rceil$ rounds. However, once a node has been compromised and later released, $\mu\mathcal{ADV}$ has no certainty on that node's status in following rounds. Thus, a realistic $\mu\mathcal{ADV}$ might inadvertently pick some yellow sensors in the next round compromise set.

Despite this, we consider a model of *Informed Adversary* ($\mathcal{INF} - \mu\mathcal{ADV}$), which is always aware of the current set of green nodes. Although unrealistic, we believe that $\mathcal{INF} - \mu\mathcal{ADV}$ represents the upper bound in adversarial capability. It thus serves as a reference point for the worst-case analysis. If our protocols perform well in the presence of $\mathcal{INF} - \mu\mathcal{ADV}$, they would perform considerably better with a more limited (and realistic) adversary.

A more realistic adversary can still compromise green sensors for the first $\lceil \frac{n}{k} \rceil$ rounds, moving in a round-robin fashion. Then, it can apply some heuristics to maximize its chances of selecting green nodes. Based on the heuristic that earliest-compromised nodes had the most rounds to recover secrecy, $\mu\mathcal{ADV}$ might keep its round-robin strategy, even after round $\lceil \frac{n}{k} \rceil$. We refer to this adversary type as $\mathcal{RR} - \mu\mathcal{ADV}$. In the next sections, we show the difference in performance between $\mathcal{INF} - \mu\mathcal{ADV}$ and $\mathcal{RR} - \mu\mathcal{ADV}$.

Critical parameters for the success of our self-healing techniques are the ratio $\frac{k}{n}$ and the degree of collaboration among sensors, t. Another important factor is the adversary's ability to eavesdrop on exchanged contributions. A single green contribution can help a yellow sensor to regain secret state, only if the contribution is not eavesdropped by $\mu\mathcal{ADV}$ via compromised (red) sensors. Local eavesdropping is determined by the network layout. Also, routing algorithms can clearly influence $\mu\mathcal{ADV}$'s eavesdropping power and the results of proposed self-healing techniques. We decided to make our analysis independent of such factors, yet realistic. To this extent, we define p as the probability of any message being eavesdropped by the adversary.

6.3. *PUSH*

We define P_G^r (P_Y^r, P_R^r) as the probability that $s_i \in G^r$ ($s_i \in Y^r, s_i \in R^r$) at the end of round r, while β^r is the probability that a yellow sensor at the beginning of round r does not become green in the next round.

A sensor is red at the end of round r if it was compromised at the beginning of the same round. Recall that $\mathcal{INF} - \mu\mathcal{ADV}$ only compromises green sensors. Thus, we have the following.

$$\begin{aligned}
P_R^r &= P(s_i \in R^r | s_i \in G^{r-1}) \cdot P(s_i \in G^{r-1}) + P(s_i \in R^r | s_i \in Y^{r-1}) \cdot \\
&\quad \cdot P(s_i \in Y^{r-1}) + P(s_i \in R^r | s_i \in R^{r-1}) \cdot P(s_i \in R^{r-1}) \\
&= P(s_i \in R^r | s_i \in G^{r-1}) \cdot P(s_i \in G^{r-1}) \\
&= \frac{k}{n \cdot P_G^{r-1}} \cdot P_G^{r-1} \\
&= \frac{k}{n}.
\end{aligned}$$

A sensor is yellow at the end of round r if it was yellow at the start of that round and did not become green during the same round. A sensor is yellow at the start of a round if it was either yellow or red at the end of the previous one. Thus, we have

$$P_Y^r = (P_R^{r-1} + P_Y^{r-1})\beta^r.$$

At round r, a yellow sensor does not become green by the contribution of one green peer if (1) it is not chosen as a recipient by that green peer, or (2) the contribution is intercepted by $\mathcal{INF} - \mu\mathcal{ADV}$. The probability of (1) is $1 - \frac{t}{n-1}$, while the probability of (2) is $\frac{t \cdot p}{n-1}$. During round r, the average number of green nodes is $n \cdot P_G^{r-1} - k$, as there were, on average, $n \cdot P_G^{r-1}$ green nodes at the start of the round and k of them were later compromised. Thus, leveraging the approximation introduced in Section 6.1, β^r can be computed as

$$\beta^r = \left(1 - \frac{t}{n-1} + \frac{t \cdot p}{n-1}\right)^{n \cdot P_G^{r-1} - k} = \left(1 - t \cdot \frac{1-p}{n-1}\right)^{n \cdot P_G^{r-1} - k}.$$

Finally we have the following.

$$\begin{aligned}
P_G^r &= 1 - P_R^r - P_Y^r \\
&= 1 - P_R^r - (P_R^{r-1} + P_Y^{r-1})\beta_r \\
&= 1 - \frac{k}{n} - (P_R^{r-1} + P_Y^{r-1})\beta_r \\
&= 1 - \frac{k}{n} - (P_R^{r-1} + P_Y^{r-1}) \cdot \left(1 - t \cdot \frac{1-p}{n-1}\right)^{n \cdot P_G^{r-1} - k}.
\end{aligned}$$

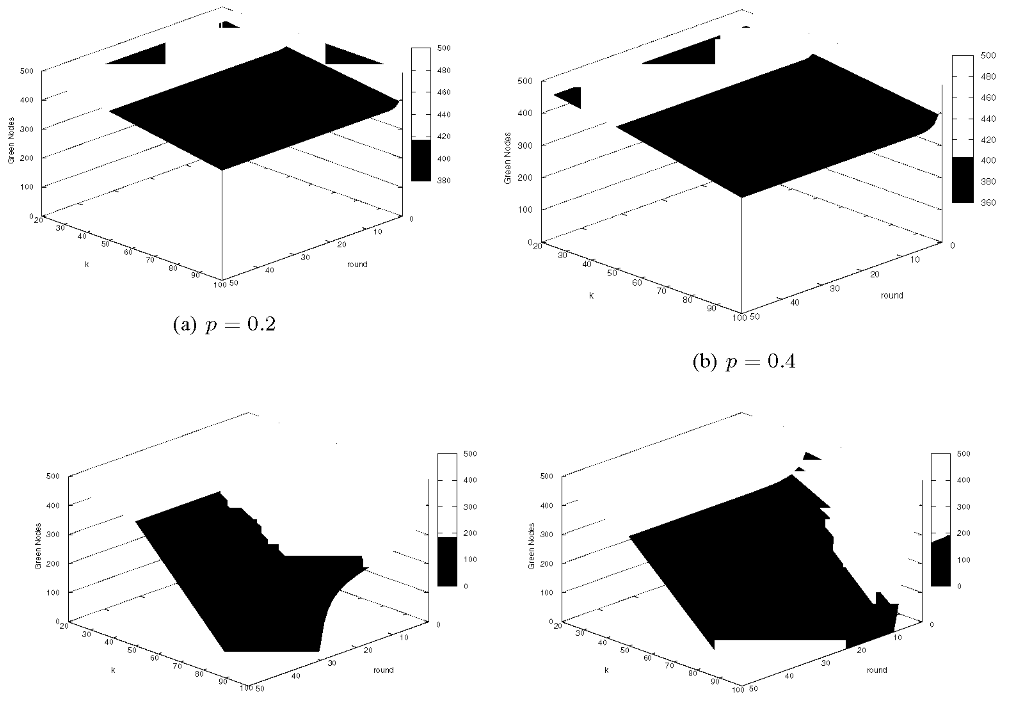
Recall that at round 0, $P_Y^0 = P_R^0 = 0$, while at round 1, $P_Y^1 = 0$, $P_R^1 = \frac{k}{n}$. We can then compute P_G^r for any round.

Figure 2 shows our simulation results[7] for a network of 500 nodes, $t = 4$, and k ranging between 20 and 100. As shown in Figure 2, if $\mathcal{INF} - \mu\mathcal{ADV}$ eavesdrops on 20% or 40% of the messages (Figure 2(a) and Figure 2(b)), the number of green sensors decreases in the first rounds and then remains constant. However, if $\mathcal{INF} - \mu\mathcal{ADV}$ increases its eavesdropping power, it eventually subverts the whole network, as shown in Figure 2(c), and Figure 2(d).

6.4. PULL

In *PULL*, the game between the UWSN and the $\mathcal{INF} - \mu\mathcal{ADV}$ changes only in terms of sensor cooperation specifics, that is, β^r is different. Here, each sensor asks for contributions from a set of randomly chosen peers. A yellow requester becomes green if at least one of the requested peers is green and the contribution is not intercepted.

[7]Values were averaged over 1,000 trials.



(a) $p = 0.2$ (b) $p = 0.4$

(c) $p = 0.6$ (d) $p = 0.8$

Fig. 2. Simulation results for *PUSH* ($n = 500, t = 4$).

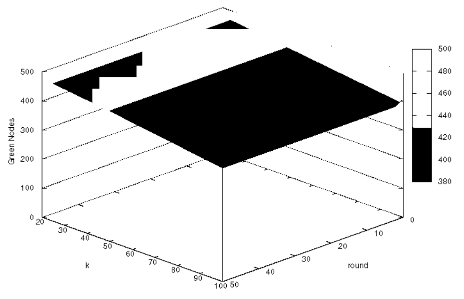
To estimate β_r, we consider that a yellow sensor does not get healed because (a) it selects t sponsors from $R^r \cup Y^r$, or (b) it selects $0 < i \leq t$ sponsors from G^r, but their contributions are intercepted. Event (a) happens with probability $P(a) = \frac{\binom{n \cdot P_S^{r-1}+k-1}{t}}{\binom{n-1}{t}}$, while event (b) has probability $P(b) = \sum_{i=1}^{t} p^i \cdot \frac{\binom{n\cdot(1-P_S^{r-1})-k}{i}\binom{n\cdot P_S^{r-1}+k-1}{t-i}}{\binom{n-1}{t}}$.

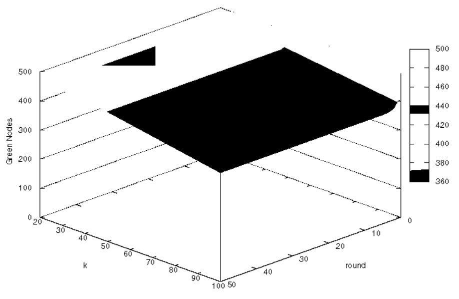
Thus we have the following.

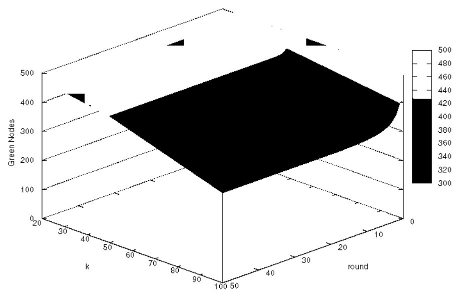
$$\begin{aligned}\beta^r &= \frac{\binom{n\cdot P_S^{r-1}+k-1}{t}}{\binom{n-1}{t}} + \sum_{i=1}^{t} p^i \cdot \frac{\binom{n\cdot(1-P_S^{r-1})-k}{i}\binom{n\cdot P_S^{r-1}+k-1}{t-i}}{\binom{n-1}{t}} \\ &= \sum_{i=0}^{t} p^i \cdot \frac{\binom{n\cdot(1-P_S^{r-1})-k}{i}\binom{n\cdot P_S^{r-1}+k-1}{t-i}}{\binom{n-1}{t}}.\end{aligned}$$

Finally, we have

$$\begin{aligned}P_G^r &= 1 - P_R^r - P_Y^r \\ &= 1 - P_R^r - \left(P_R^{r-1} + P_Y^{r-1}\right)\beta_r \\ &= 1 - \frac{k}{n} - \left(P_R^{r-1} + P_Y^{r-1}\right)\beta_r \\ &= 1 - \frac{k}{n} - \left(P_R^{r-1} + P_Y^{r-1}\right) \cdot \sum_{i=0}^{t} p^i \cdot \frac{\binom{n\cdot(1-P_S^{r-1})-k}{i}\binom{n\cdot P_S^{r-1}+k-1}{t-i}}{\binom{n-1}{t}}.\end{aligned}$$


(a) $p = 0.2$


(b) $p = 0.4$


(c) $p = 0.6$

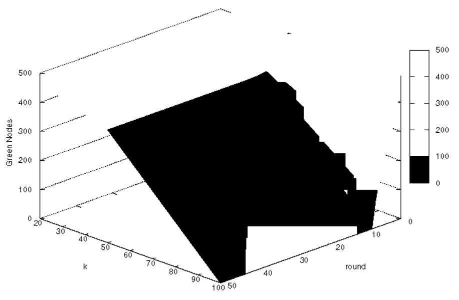

(d) $p = 0.8$

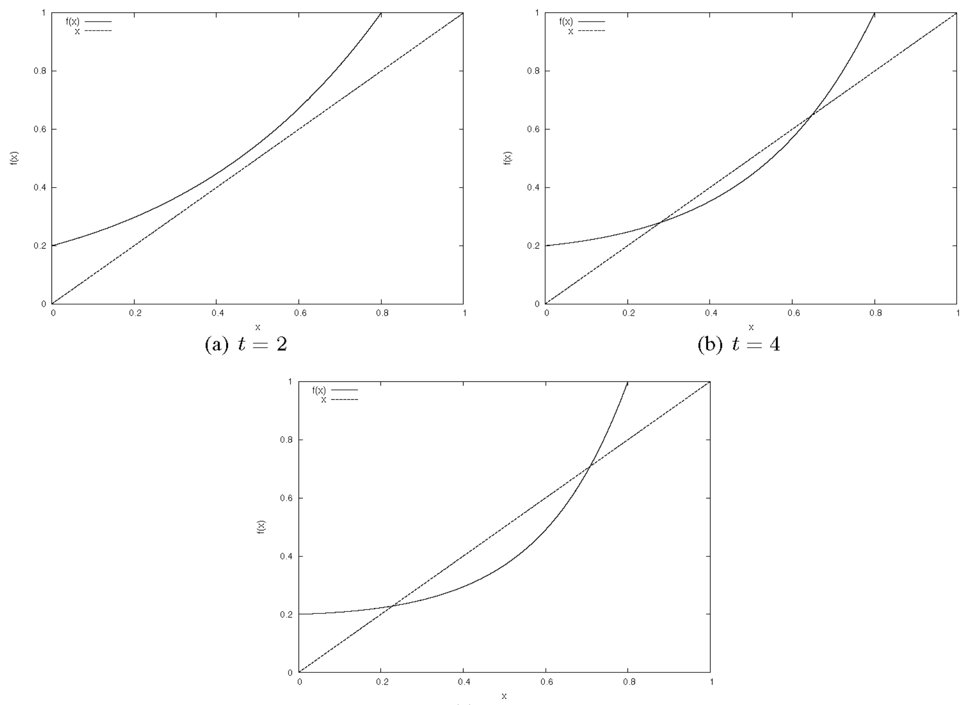
Fig. 3. Simulation results for *PULL* ($n = 500, t = 4$).

Figure 3 shows our simulation results of *PULL* under the same set of parameters as in *PUSH*. The number of green sensors remains stable, even if $\mathcal{INF} - \mu\mathcal{ADV}$ eavesdrops on 60% of all messages. However, note that the number of messages required by the *PULL* protocol is twice the number of messages required by the *PUSH* protocol. A comparison between *PUSH* and *PULL* focusing on this specific aspect is provided in Section 6.7.

6.5. Threshold Phenomena (0-1 law)

From the plots in Figures 2 and 3, it is easy to see that both *PUSH* and *PULL* exhibit a threshold phenomenon. The few first rounds show a decrease in the number of green nodes proportional to $\mu\mathcal{ADV}$'s compromise and message interception power. After this initial phase, if system parameters (n, k, t, p) are below a threshold configuration, the number of green nodes remains stable thereafter. Regardless of $\mu\mathcal{ADV}$'s strategy, there are always enough green sensors to heal some yellow sensors so that the size of G^r remains stable. However, if parameters cross the threshold, $\mu\mathcal{ADV}$ becomes powerful enough to counter the healing activity; it eventually compromises the entire network. Thereafter, collaborative self-healing is useless.

To characterize the threshold behavior, we define the set of *sick* sensors (either yellow or red) as $S^r = R^r \cup Y^r$. We hence define the probability of a node being sick as $P_S^r = P_R^r + P_Y^r$ and investigate the configuration of system parameters under which the network reaches steady state. We use *PUSH* as an example of illustrating the threshold phenomena (as for *PULL*, the threshold phenomena can be addressed via a



(a) $t = 2$ (b) $t = 4$ (c) $t = 6$

Fig. 4. Fixpoint of $f(\cdot)$ for *PUSH* ($n = 500, k = 100, p = 0.4$).

similar argument). P_S^r can be represented as the following.

$$\begin{aligned} P_S^r &= 1 - P_G^r\frac{k}{n} + P_S^{r-1}\cdot\left(1 - t\cdot\frac{1-p}{n-1}\right)^{n\cdot(1-P_S^{r-1})-k} \\ &= 1 - P_G^r\frac{k}{n} + P_S^{r-1}\cdot\left(1 - t\cdot\frac{1-p}{n-1}\right)^{n\cdot(1-P_S^{r-1})-k}. \end{aligned}$$

Since $P_S^0 = 0$ and $P_S^1 = \frac{k}{n}$ we can define a recursive function $f(\cdot)$ as follows.

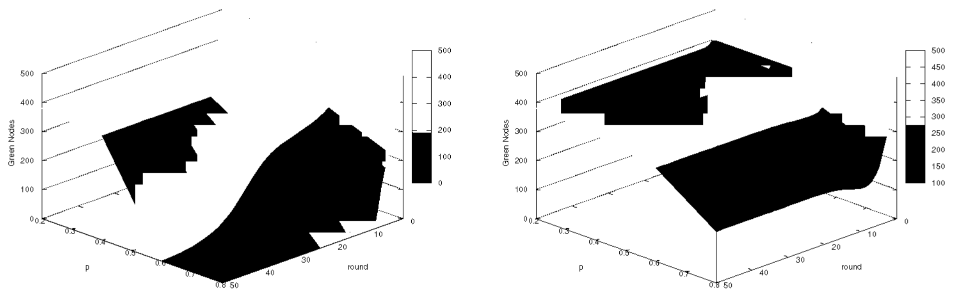
$$P_S^r = \begin{cases} 0 & \text{if } r = 0, \\ f(0) = \frac{k}{n} & \text{if } r = 1, \\ P_S^r = f(P_S^{r-1}) & \text{if } r > 1, \end{cases}$$

where

$$f(x) = \frac{k}{n} + x\cdot\left(1 - t\cdot\frac{1-p}{n-1}\right)^{n\cdot(1-x)-k}.$$

The network reaches a steady state if P_S^r equals the *fixpoint* of $f(\cdot)$. Given the perceived threat factors as $\frac{k}{n}$ and p, it is easy to find the smallest t such that there exists a solution $0 < \tilde{x} < 1$ to equation $f(x) = x$. This value is the minimum t for which the network reaches a steady number of green nodes.

Figure 4 plots $f(\cdot)$ against the straight line $y = x$ for different values of t, while n, k, and p remain constant. The first plot (Figure 4(a)) with $t = 2$ does not exhibit a fixpoint, as $f(\cdot)$ never intersects $y = x$. That is, $\mu\mathcal{ADV}$ is expected to turn the number



(a) $\mathcal{INF} - \mu\mathcal{ADV}$ (b) $\mathcal{RR} - \mu\mathcal{ADV}$

Fig. 5. Comparison between $\mathcal{INF} - \mu\mathcal{ADV}$ and $\mathcal{RR} - \mu\mathcal{ADV}$ in *PUSH* ($n = 500, k = 100, t = 4$).

of green nodes to 0. However, if t is increased to 4 (Figure 4(b)), the curve intersects $y = x$ in two points, and the first, $\tilde{x}$, yields P_r^S for which the network is in steady state. In other words, the number of green nodes first drops to $n \cdot (1 - \tilde{x})$ and then remains stable. As shown in Figure 4(c), by increasing t, we can decrease the fixpoint of $f(\cdot)$ and, consequently, increase the number of green nodes at steady state. However, increasing t involves more overhead. Once we find the minimum t for which $f(\cdot)$ has a fixpoint, it might not be advantageous to increase the degree of collaboration. However, since our analysis leverages the assumption of Section 6.1, the required t might be slightly larger than the computed one.

6.6. $\mathcal{INF} - \mu\mathcal{ADV}$ vs. $\mathcal{RR} - \mu\mathcal{ADV}$

The analysis provided in the previous section considers an adversary that is always aware of the status of each sensor and compromises green nodes only (if any). As explained in Section 6.2, such an adversary is not realistic, and it is studied as a benchmark for our self-healing protocols. A more realistic adversary would not have knowledge of the status of all the sensors and would, rather, use some heuristic to pick the set of sensors to compromise during the next round. $\mathcal{RR} - \mu\mathcal{ADV}$, uses compromise time as a rationale for a simple yet meaningful heuristic: a sensor compromised earlier in the past has more chances to become green than a sensor compromised later on in time. Thus, $\mathcal{RR} - \mu\mathcal{ADV}$ partitions the network into $\lceil \frac{n}{k} \rceil$ sets of k sensors and moves through them in a round-robin fashion. Figure 5 shows the results of simulation of *PUSH* against $\mathcal{INF} - \mu\mathcal{ADV}$ and $\mathcal{RR} - \mu\mathcal{ADV}$. Since the latter might choose to compromise sensors that are not green, network resilience performance is sensibly better.

Hence, on the one hand, when dealing with a realistic adversary, the desired degree of resilience can be achieved with a reduced degree of cooperation as compared to an omniscient yet unrealistic adversary. On the other hand, the UWSN could be tuned to adopt the worst case represented by the $\mathcal{INF} - \mu\mathcal{ADV}$, hence offering a stronger resilience against a realistic yet unknown adversary.

6.7. *PUSH* and *PULL* Comparison

Message Overhead. The number of messages exchanged is an important overhead factor for resource-constrained UWSNs. Security of the proposed collaborative approaches is based on sensors' ability to communicate with any peer in the network. In general, the number of messages generated by any sensor during one round of *PUSH* is t; the energy spent to forward messages on behalf of other peers is highly dependent on the network shape, topology, background noise, sensor position, etc.. *PULL* requires twice

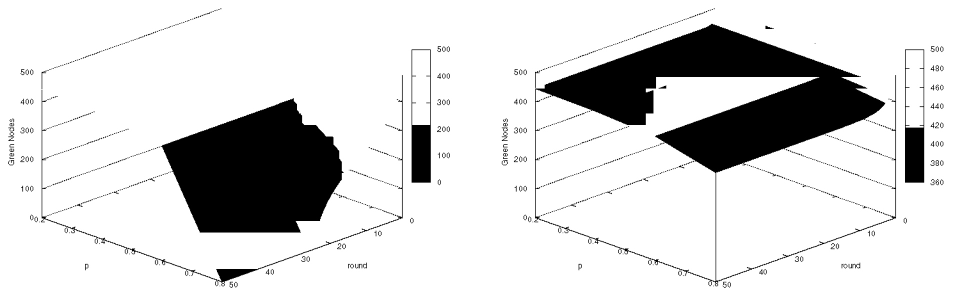

(a) $PULL$ - $t = 2$ (b) $PUSH$ - $t = 4$

Fig. 6. Comparison between *PUSH* and *PULL* ($n = 500, k = 50$) given a fixed number of messages per round.

as many messages as *PUSH*, because it needs two messages—request and reply—for each contribution. *PUSH* entails a single (unsolicited) contribution message. Given the same number of messages per round, Figure 6 shows the difference between the two schemes with $t = 4$ for *PUSH* and $t = 2$ for *PULL*. While the network always reaches a stable number of green sensors in *PUSH*, a powerful $\mu\mathcal{ADV}$ eventually compromises all sensors in *PULL*.

Contribution Assurance. In *PULL*, all sensors receive the same number (t) of contributions. In *PUSH*, sponsored peers are picked randomly, and (for small values of t) contributions can reflect a small unbalance in their allocation to peers. This is why *PULL* performs better than *PUSH* for small values of t. If t is small, there might be sensors that receive no contributions at all in *PUSH*. Conversely, there might be sensors that receive far more than t contributions.

We note that the differences in self-healing between the two schemes are negligible as long as all sensors receive at least one contribution. In particular, for *PUSH*, t must be tuned so that each sensor gets at least one contribution. For example, if $t = ln(n) + c$, the probability[8] of a sensor receiving no contribution is upper bounded by e^{-c}.

Targeting Specific Sensors. In *PULL*, $\mu\mathcal{ADV}$ could constantly keep control of a specific, previously compromised sensor state, even while being away from that sensor. Indeed, $\mu\mathcal{ADV}$ knows which peers each yellow (and red) sensor will ask for a contribution. For example, at the end of round r, $\mu\mathcal{ADV}$ can release a red s_i (let it become yellow) and continue monitoring s_i's key evolution; indeed, before releasing it, $\mu\mathcal{ADV}$ copies the current state of s_i, and it knows K_i^r as well as the set of s_i's future sponsors. $\mu\mathcal{ADV}$ can compromise all green sponsors (if any) among the s_i's future sponsors and acquire their contributions to s_i. This allows $\mu\mathcal{ADV}$ to compute K_i^{r+1}, that is, s_i remains yellow.

The control capability of $\mu\mathcal{ADV}$ is annihilated by *PUSH*, because $\mu\mathcal{ADV}$ can not predict the behavior of any green sensor. In particular, $\mu\mathcal{ADV}$ does not know which t peers a given green sensor will sponsor. Thus, once $\mu\mathcal{ADV}$ releases a previously red s_i, it can not determine with certainty the set of sensors that might contribute to K_i^{r+1}. Indeed, each green sensor (if any) has the same probability $1/(n-1)$ to contribute to s_i. $\mu\mathcal{ADV}$ has only two ways to be sure that s_i continues evolving in a yellow state: (a) there are no green sensors in the UWSN, or (b) $\mu\mathcal{ADV}$ must control the area around s_i, in order to eavesdrop on all incoming messages—something that could not be always practical for real-world $\mu\mathcal{ADV}$.

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

7. COOPERATION DRAWBACKS AND IMPROVEMENTS

If the sensor secret state (or, in particular, keys) evolves, as in Section 3.1, the sink can use the secret it shares with sensor s_i at round 0 to mimic the secret state evolution process and compute subsequent round secret states, so that it can decrypt any ciphertext produced by s_i at any round. However, if collaboration among sensors is used to update the secret state, the sink must be aware of which interactions occurred among sensors, and this must hold for each round. In other words, since the sink knows the initial state of each sensor, it can recompute the state of each s_i (notably, K_i^r) at every round. This assertion holds as long as communication is perfect, that is, all messages are delivered in a timely fashion and sensors do not fail. Unfortunately, this is not the case in real-word deployments where messages can go lost because of the unreliable communication medium, while sensors can fail due to unexpected malfunctioning, battery exhaustion, or deliberate destruction, to cite a few. We now attempt to relax these ideal-world assumptions. Without loss of generality, we will only discuss the preceding issues with regards to *PUSH*. A similar argument could be developed for *PULL*.

7.1. Unreliable Communication & Reliable Sensors

In the following, we assume that while sensors still do not fail, messages are delivered with some probability smaller than 1 (we deal with sensors failure in next section). In other words, a contribution sent in round r to s_i might not be received by the latter. 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 actually received by s_i at the same round. Then, s_i would compute $K_i^{r+1} = F(K_i^r||Z_i^r)$, while the sink would compute $K_i^{r+1} = F(K_i^r||W_i^r)$. Clearly, the sink needs extra information to correctly compute K_i^{r+1}. A simple and effective solution would be for s_i to store, for each round r, a set of IDs: $\mathcal{ID}_i^r = \{j|s_j \textit{ has a contribution in } Z_i^r\}$. With this amendment, storage overhead is kept linear, that is, $O(v * t)$, where v is the maximum number of unattended collection rounds and t is the maximum number of contributions received within a round. The advantage is that once the sink learns $\mathcal{ID}_i^r$, it can correctly compute K_i^r.

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.

7.2. Unreliable Communication & Unreliable Sensors

If we now relax the assumption that sensors do not fail, learning $\mathcal{ID}_i^r$ for a given secret state update might not be enough for the sink to recompute K_i^{r+1}. 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 network, it can not reconstruct the status of s_j at round r and, as a result, can not compute K_i^{r+1}. This demonstrates the interdependence among sensors' secret states and reveals that both *PULL* and *PUSH* are fragile with respect to sensors failures.

A viable and secure solution is to introduce public key cryptography. While public key cryptography has been regarded as a poor match for resource-constrained sensors, recent studies [Wander et al. 2005] show that public key cryptography can be sporadically used in WSNs to afford high security at reasonable costs.

Assuming that the sink has a well-known permanent public key PK_{sink}, both *PUSH* and *PULL* would operate in almost the same manner, as previously described. The only difference is that s_i would encrypt its secret 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 "$PK_{sink} + K_i^r$" approach seems to simultaneously solve all problems stemming from both unreliable communication and unreliable sensors. However, it incites Deletion-n-Replace attacks. That is, $\mu\mathcal{ADV}$ can simply delete any data found on sensors and then replace them with arbitrary values. This motivates us to look at extra security mechanisms to prevent such attacks.

7.3. Defending Against Delete-n-Replace Attack

Delete-n-replace attacks could be avoided using Forward Secure Aggregation Authentication (*FssAgg*) techniques [Ma and Tsudik 2007] along with the "$PK_{sink} + K_i^r$" approach. The augmented scheme has the following system operations.

(1) *Initialization Stage.* Before deployment, sensor s_i uses its initial key K_i^0 shared with the sink to generate a MAC $\mu_{0,0}$ over a dummy data item d_i^0 known by the sink. The generation of $\mu_{0,0}$ is necessary, since it prevents $\mu\mathcal{ADV}$ to erase all data found on s_i.

(2) *Data Generation Stage.* At round r, upon sensing data d_i^r, sensor s_i does the following.
   (a) Encrypts d_i^r with K_i^r: $E_i^r = Enc_{K_i^r}(d_i^r)$.
   (b) Computes $\mu_{0,r} = H(\mu_{0,r-1}||MAC(K_i^r, E_i^r))$, where MAC is a generic algorithm for producing a message authentication code.
   (c) Encrypts K_i^r with PK_{sink}: $CK_i^r = Enc(PK_{sink}, K_i^r)$.
   (d) Generates K_i^{r+1} through *PUSH* or *PULL*.

   A sensor's storage at the end of round r contains the following.
   $[(d_i^0), (E_i^1, CK_i^1), \ldots, (E_i^r, CK_i^r), \mu_{0,r}, K_i^{r+1}, PK_{sink}]$.

(3) *Data Processing Stage.* At round r, the sink approaches the network and gets $[(d_i^0), (E_i^1, CK_i^1), \ldots, (E_i^{r'}, CK_i^{r'}), \mu_{0,r'}]$ from a sensor. It performs the following operations.
   (a) Checks whether $r = r'$. If not, $\mu\mathcal{ADV}$ has compromised the sensor and either inserted fake data or deleted existing data. The sink simply discards the data and moves to the next sensor.
   (b) Otherwise, decrypts $CK_i^{r'}$ using its private key SK_{sink} to get $K_i^{r'}$ for $r' = 1, \ldots, r$.
   (c) Computes $\mu'_{0,r} = H(\,H(\ldots\, H(MAC(K_i^0, d_i^0)||MAC(K_i^1, d_i^1))\ldots)||MAC(K_i^r, d_i^r))$ and compares whether $\mu'_{0,r} = \mu_{0,r}$. If not, $\mu\mathcal{ADV}$ has compromised the sensor and replaced some faked data.
   (d) Otherwise, uses K_i^r to decrypt E_i^r and extracts sensor readings.

In this augmented scheme, forward and backward secrecy are achieved through *PUSH* or *PULL*. In the following, we discuss $\mu\mathcal{ADV}$'s attempt to mount delete-n-replace attacks. Suppose $\mu\mathcal{ADV}$ compromises sensor s_i at round r_1, then releases the sensor at round $r_1 + 1$, and comes back to compromise the sensor again at time r_2.

—If the sensor's state is green at $r_2 - 1$, $\mu\mathcal{ADV}$ cannot modify any data entry before r_2, including all data entries generated after round r_1.
—If the sensor's state is yellow at $r_2 - 1$, $\mu\mathcal{ADV}$ can modify and forge data entries between r_1 and r_2, since $\mu\mathcal{ADV}$ knows all the keys during these intervals, as well as $\mu_{0,r}$.

So for $\mu\mathcal{ADV}$ to have a chance to modify pre-compromise data, it must recompromise the sensor before it becomes green.

8. CONCLUSION

This article provides several contributions. First, we revised a recently introduced model (by the same authors): unattended wireless sensor networks (UWSNs). Further, we provided a thorough analysis of two self-healing mechanisms for UWSNs; this analysis led to a full, characterization of the proposed models. In particular, other than being interesting on its own, this analysis allows for the tuning of the security parameters of the model (and related overhead) to the security threat the UWSN is supposed to face. Another result is the first characterization of a 0-1 law for this type of model. Extensive simulations support our findings. We also highlighted implementation issues which UWSNs are subject to and provided a few solutions to tackle these issues. However, further research is still needed to fully explore the introduced model, as well as mechanisms, related threats, and consequent security solutions.

REFERENCES

Bellare, M., Canetti, R., and Krawczyk, H. 1996. Keying hash functions for message authentication. In *Proceedings of the 16th Annual International Cryptology Conference (Crypto'96).*

Canetti, R. and Herzberg, A. 1994. Maintaining security in the presence of transient faults. In *Proceedings of the 14th Annual International Cryptology Conference (Crypto'94).* 425–438.

Chan, H. and Perrig, A. 2005. Pike: Peer intermediaries for key establishment in sensor networks. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05).* 524–535.

Conti, M., Di Pietro, R., Mancini, L. V., and Mei, A. 2008. Emergent properties: Detection of the node-capture attack in mobile wireless sensor networks. In *Proceedings of the 1st ACM Conference on Wireless Network Security (WiSec'08).* 214–219.

Conti, M., Di Pietro, R., Mancini, L. V., and Mei, A. 2009. Mobility and cooperation to thwart node capture attacks in manets. *EURASIP J. Wireless Commun. Netw.* DOI:10.1155/2009/945943.

DARPA, I. 2007. Bba 07-46 landroids broad agency announcement. http://www.darpa.mil/ipto/solicit/baa/BAA-07-46 PIP.pdf.

Di Pietro, R., Ma, D., Soriente, C., and Tsudik, G. 2008. POSH: Proactive co-operative self-healing in unattended sensor networks. In *Proceedings of the 27th IEEE International Symposium on Reliable Distributed Systems (SRDS'08).* 185–194.

Di Pietro, R., Mancini, L., and A.Mei. 2003. Random key assignment for secure wireless sensor networks. In *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'03).* 62–71.

Di Pietro, R., Mancini, L., Soriente, C., Spognardi, A., and Tsudik, G. 2008. Catch me (if you can): Data survival in unattended sensor networks. In *Proceedings of the 6th IEEE International Conference on Pervasive Computing and Communications (PerCom'08).* 185–194.

Di Pietro, R., Mancini, L., Soriente, C., Spognardi, A., and Tsudik, G. 2009a. Data security in unattended wireless sensor networks. *IEEE Trans. Comput. 58,* 11, 1500–1511.

Di Pietro, R., Mancini, L., Soriente, C., Spognardi, A., and Tsudik, G. 2009b. Playing hide-and-seek with a focused mobile adversary in unattended wireless sensor networks. *Ad Hoc Netw. 7,* 8, 1463–1475.

Di Pietro, R., Oligeri, G., Soriente, C., and Tsudik, G. 2010a. Intrusion-resilience in mobile unattended WSNs. In *Proceedings of the 29th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'10).* 2303–2311.

Di Pietro, R., Oligeri, G., Soriente, C., and Tsudik, G. 2010b. Securing mobile unattended WSNs against a mobile adversary. In *Proceedings of the 29th IEEE International Symposium on Reliable Distributed Systems (SRDS'10).* 11–20.

Di Pietro, R., Soriente, C., Spognardi, A., and Tsudik, G. 2009. Collaborative authentication in unattended WSNs. In *Proceedings of the 2nd ACM Conference on Wireless Network Security (WiSec'09).* 237–244.

Dodis, Y., Franklin, M., Katz, J., Miyaji, A., and Yung, M. 2003. Intrusion-resilient public-key encryption. In *Proceedings of the Cryptographers' Track at the RSA Conference (CT-RSA'03).* 19–32.

Dodis, Y., Franklin, M., Katz, J., Miyaji, A., and Yung, M. 2004. A generic construction for intrusion-resilient public-key encryption. In *Proceedings of the Cryptographers' Track at the RSA Conference (CT-RSA'04).* 81–98.

Dodis, Y., Katz, J., Xu, S., and Yung, M. 2002. Key-insulated public key cryptosystems. In *Proceedings of the 21st International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt'02)*. 65–82.

Du, W., Deng, J., Han, Y., Chen, S., and Varshney, P. 2004. A key management scheme for wireless sensor networks using deployment knowledge. In *Proceedings of the 23th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'04)*. 586–597.

Eschenauer, L. and Gligor, V. 2002. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02)*. 41–47.

Frankel, Y., Gemmel, P., MacKenzie, P., and Yung, M. 1997. Proactive rsa. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'97)*. 440–454.

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

Ganeriwal, S., Čapkun, S., Han, C., and Srivastava, M. 2005. Secure time synchronization service for sensor networks. In *Proceedings of the 4th ACM Workshop on Wireless Security (WiSec'05)*. 97–106.

Hu, F. and Sharma, N. 2005. Security considerations in ad hoc sensor networks. *Ad Hoc Netw. (Elsevier) 3,* 1, 69–89.

Kong, J. and Hong, X. 2003. Anodr: Anonymous on demand routing with untraceable routes for mobile ad-hoc networks. In *Proceedings of the 4th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'03)*. 291–302.

Luo, J., Papadimitratos, P., and Hubaux, J.-P. 2008. Gossicrypt: Wireless sensor network data confidentiality against parasitic adversaries. In *Proceedings of the 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'08)*. 441–450.

Ma, D. 2008. Practical forward secure sequential aggregate signatures. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS'08)*. 341–352.

Ma, D., Soriente, C., and Tsudik, G. 2009. New adversary and new threats: Security in unattended sensor networks. *IEEE Network 23,* 2, 43–48.

Ma, D. and Tsudik, G. 2007. Extended abstract: Forward-secure sequential aggregate authentication. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'07)*. 86–91.

Ma, D. and Tsudik, G. 2008. DISH: Distributed self-healing in unattended wireless sensor networks. In *Proceedings of the 10th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS'08)*. 47–62.

Mauw, S., van Vessem, I., and Bos, B. 2006. Forward secure communication in wireless sensor networks. In *Proceedings of the 3rd International Conference Security in Pervasive Computing (SPC'06)*. 32–42.

Naik, V., Arora, A., Bapat, S., and Gouda, M. 2003. Whisper: Local secret maintenance in sensor networks. In *Proceedings of the Workshop on Principles of Dependable Systems (PoDSy'03)*.

Ostrovsky, R. and Yung, M. 1991. How to withstand mobile virus attacks. In *Proceedings of the 10th ACM Symposium on Principles of Distributed Computing (PODC'91)*. 51–59.

Park, T. and Shin, K. 2005. Soft tamper-proofing via program integrity verification in wireless sensor networks. *IEEE Trans. Mobile Comput. 4,* 3, 297–309.

Rabin, T. 1998. A simplified approach to threshold and proactive RSA. In *Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'98)*. 89–104.

Ren, W., Ren, Y., and Zhang, H. 2010. Secure, dependable and publicly verifiable distributed data storage in unattended wireless sensor networks. *Sci. China Info. Sci. 53,* 5, 964–979.

Ren, Y., Oleshchuk, V. A., and Li, F. Y. 2009. Secure and efficient data storage in unattended wireless sensor networks. In *Proceedings of the 3rd International Conference on New Technologies, Mobility and Security (NTMS'09)*. 1–5.

Ren, Y., Oleshchuk, V. A., and Li, F. Y. 2010. A scheme for secure and reliable distributed data storage in unattended wsns. In *Proceedings of the Global Communications Conference (GLOBECOM'10)*. 1–6.

Ruan, Z., Sun, X., Liang, W., Sun, D., and Xia, Z. 2010. Cads: Co-operative anti-fraud data storage scheme for unattended wireless sensor networks. *Info. Techno. J. 9,* 7, 1361–1368.

Seshadri, A., Perrig, A., Van Doorn, L., and Khosla, P. 2004. Swatt: Software-based attestation for embedded devices. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'04)*. 272–282.

Shoup, V. 2000. Oaep reconsidered. Cryptology ePrint Archive, Report 2000/060.

Trident Systems. 2010. Tridents family of unattended ground sensors. http://www.tridsys.com/white-unattended-ground-sensors.htm.

Vitali, D., Spognardi, A., and Mancini, L. 2011. Replication schemes in unattended wireless sensor networks. In *Proceedings of the 4th International Conference on New Technologies, Mobility and Security (NTMS'11)*. 1–5.

WANDER, A., GURA, N., EBERLE, H., GUPTA, V., AND SHANTZ, S. C. 2005. Energy analysis of public-key cryptography for wireless sensor networks. In *Proceedings of the 3rd IEEE International Conference on Pervesive Computing and Communication*. 324–328.

WANG, Q., REN, K., LOU, W., AND ZHANG, Y. 2009. Dependable and secure sensor data storage with dynamic integrity assurance. In *Proceedings of the 28th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'09)*. 954–962.

YANG, Y., WANG, X., ZHU, S., AND CAO, G. 2007. Distributed software-based attestation for node compromise detection in sensor networks. In *Proceedings of the 26th IEEE Symposium on Reliable Distributed Systems (SRDS'07)*. 219–230.

YAVUZ, A. AND NING, P. 2009. Hash-based sequential aggregate and forward secure signature for unattended wireless sensor networks. In *Proceedings of the 6th Annual International Conference on Mobile and Ubiquitous Systems (MobiQuitous'09)*. 1–10.

YU, C., CHEN, C., LU, C., KUO, S., AND CHAO, H. 2010. Acquiring authentic data in unattended wireless sensor networks. *Sensors 10,* 4, 2770–2792.