Securing Mobile Unattended WSNs against a Mobile Adversary

Roberto Di Pietro § ‡, Gabriele Oligeri *, Claudio Sordini † ¶, Gene Tsudik †

* ISTI-CNR, Pisa Research Area, Pisa, Italy
† Computer Science Department, University of California, Irvine, USA
‡ Department of Mathematics, Università di Roma Tre, Rome, Italy
§ UNESCO Chair in Data Privacy, Universitat Rovira i Virgili, Tarragona, Spain
¶ DLSIS, Universidad Politécnica de Madrid, Spain

Abstract—One important factor complicating security in Wireless Sensor Networks (WSNs) is lack of inexpensive tamper-resistant hardware in commodity sensors. Once an adversary compromises a sensor, all memory and forms of storage become exposed, along with all secrets. Thereafter, any cryptographic remedy ceases to be effective. Regaining sensor security after compromise (i.e., intrusion-resilience) is a formidable challenge. Prior approaches rely on either (1) the presence of an on-line trusted third party (sink), or (2) the availability of a True Random Number Generator (TRNG) on each sensor. Neither assumption is realistic in large-scale Unattended Wireless Sensor Networks (UWSNs) composed of low-cost commodity sensors, periodic visits by the sink.) Previous work has demonstrated that sensor collaboration is an effective, yet expensive, means of attaining intrusion-resilience in UWSNs.

In this paper, we explore intrusion resilience in Mobile UWSNs in the presence of a powerful mobile adversary. We show how the choice of the sensor mobility model influences intrusion-resilience with respect to this adversary. We also explore self-healing protocols that require only local communication. Results indicate that sensor density and neighborhood variability are the two key parameters affecting intrusion resilience. Our findings are supported by extensive analyses and simulations.

I. INTRODUCTION

Wireless Sensor Networks (WSNs) enable environment monitoring and data collection in remote, inhospitable and even hostile environments. In a typical WSN encountered in the research literature, a powerful trusted entity (usually called a sink) manages a large number of inexpensive sensors that monitor various ambient phenomena, e.g., temperature, pollution, vibration or humidity. While many WSN types are composed of static sensors, it has been shown [1] that certain WSN functions benefit from sensor mobility. In particular, mobility can improve WSN coverage area, routing, self-deployment as well as load-balancing.

In a large-scale WSN composed of low-cost commodity sensors, security is exacerbated by some well-known limitations stemming from resource and cost constraints. In particular, lack of inexpensive tamper-resistant hardware makes sensor compromise a real threat. It has been convincingly demonstrated [2] that compromising commodity sensors is surprisingly easy. Moreover, remote attacks (i.e., those without direct physical access) can result in compromise of memory and other forms of storage [3]. Generally, once a sensor is compromised and its secrets become known to the adversary, any cryptographic protocol becomes ineffective. For example, as soon as the encryption key used by a sensor is exposed, any message encrypted earlier (or to be encrypted in the future) under the same key can be trivially decrypted.

In terms of security, the "lifetime" of a given sensor can be viewed as being composed of three epochs, based on the time of (potential) compromise: (1) before, (2) during, and (3) after, compromise. During epoch (2), the adversary controls the sensor and no security measure can help. However, effects of compromise can be limited to epoch (2) if the underlying cryptographic techniques are both forward- and backward-secure. Informally, a cryptographic protocol is forward secure if exposure of secret material (keys) at time $t$ does not lead to compromise of any secrets used prior to $t$. Whereas, backward security means that compromise of secrets used at time $t$ does not lead to compromise of any secrets that will be used in the future.

Forward security can be easily obtained via one-way evolution of secrets. Suppose that time is divided into rounds and let $K^0$ be the initial secret. The secret for round $r \geq 1$ is computed as: $K^r = H(K^{r-1})$, where $H(\cdot)$ is a suitable cryptographic one-way function (e.g., SHA-2). Although, after compromising a sensor the adversary learns that sensor’s current secret, it cannot compute any secrets used in prior rounds, because of the one-wayness property of $H(\cdot)$. Unfortunately, periodic evolution of secrets does not help with backward security, since knowledge of $K^r$ allows the adversary to compute secrets for all future rounds. In particular, even after the adversary releases or leaves the compromised sensor, it can still “reconstruct” the sequence of that sensor’s future secrets [4]. In other words, sensor behavior in future rounds is deterministic from the adversary’s point of view.

Backward security can be trivially attained using a True Random Number Generator (TRNG) as a source of round-specific sensor secrets. Since a TRNG outputs information-theoretically independent values, even if the adversary learns any number of TRNG outputs, it can compute neither past nor future TRNG outputs. (More precisely, it can not compute any missing outputs.) Unfortunately, TRNGs are not readily available on commodity sensors.

Alternatively, backward secrecy can be obtained with the help of an on-line trusted third party. This is the case in so-
called key-insulated schemes [5], [6], where per-round key evolution is performed by the owner in collaboration with a trusted third party, called a base. Past and future secrets can not be learned by the adversary as long as the owner and the base are not compromised simultaneously. Key-insulated schemes are suitable for WSNs where the sink acts as a natural base. However, if the WSN is unattended and intervals between successive sink visits are long, key-insulated schemes are not applicable and backward security requires a different approach.

Our Contributions: Sensor collaboration has been shown to be an effective – yet expensive – means of intrusion-resilience in Unattended Wireless Sensor Networks (UWSNs) with static sensors in the presence of a mobile adversary [7], [8]. A recent result [9] illustrated how to leverage sensor mobility in order to obtain intrusion-resilience with fairly low overhead. Specifically, intrusion-resilience through sensor collaboration in Mobile UWSNs depends on the average number of neighbors and neighborhood variability, i.e., the number of new neighbors a given sensor acquires over time. However, the adversary considered in [9] is static and “passive” as far as its choice of sensors to compromise. In particular, it is assumed to control a fixed portion of the network deployment area and compromises all sensors that move within it.

The goal of this paper is to investigate collaborative intrusion-resilience in Mobile UWSNs in the presence of a different adversary. The envisaged adversary roams the network and seeks to undermine its overall security. It is “active” in that it chooses the portion of the deployment area to compromise at each round and thus aims to compromise different sets of sensors per round. We explore and demonstrate the effects of adversarial mobility on the performance of the proposed collaborative intrusion-resilient protocol. We show that sensor collaboration provides a high degree of intrusion-resilience at very low cost. We also highlight the relationship between network performance and various sensor mobility models (that influence neighborhood variability). In particular, sensor mobility has different effects on network performance, depending on whether the adversary is mobile or static. Results indicate that best performance against a mobile adversary is achieved with sensors moving according to the Random Jump mobility model. Finally, we introduce two new metrics: Duty Cycle and Health Ratio that characterize Mobile UWSN resilience under the considered mobility models.

II. RELATED WORK

Mobility offers some clear advantages to WSNs. For example, mobile sensors can compensate for otherwise low sensor density and improve network coverage [10]. Mobility can also mitigate sensor failure and improve overall network capacity [11]. In terms of security, mobility can be leveraged as a means of detecting sensor capture attacks [12][13].

One approach to secret state recovery after compromise was proposed by Dutta, et al. [14]. However, it requires sink presence to update sensor keys and is thus unsuitable for UWSNs where the sink is mostly absent. A method for secret state recovery without the sink was proposed by Naik, et al. [15]: session keys between two sensors are computed from two secrets, one provided by each sensor. That is, the key for session $r$ between sensors $s_j$ and $s_q$ is computed as:

$$K_{r}^{q} = F(H(K_{r}^{q-1}), H(K_{r}^{q-1}))$$

where $K_{r}^{q-1}$ and $K_{r}^{q-1}$ are $s_j$’s and $s_q$’s secrets for session $r - 1$ and $F(\cdot)$ and $H(\cdot)$ are suitable hash functions. This scheme is secure as long as the adversary does not compromise both $s_j$ and $s_q$. The same assumptions do not hold in UWSNs as their unattended nature allows the adversary to gradually compromise some (even all) sensors between successive sink visits.

Static UWSNs and their unique security threats were introduced in [16] and investigated further in [17], [18]. The use of sensor collaboration to provide intrusion-resilience in static UWSNs was explored in [7] and [8]. However, both techniques impose very high communication overhead due to multi-hop message exchanges. Security in Mobile UWSNs was first studied in [9]. The scope of that work was limited to the static adversary model where the adversary was limited to compromising sensors that would wander into a fixed area controlled by the adversary. In contrast, this paper explores Mobile UWSNs facing a more powerful (and more realistic) mobile adversary that can choose the set of sensors to compromise at each round.

III. SYSTEM MODEL

This section describes our network environment and the adversarial model. Table I summarizes the notation.

A. Network Environment

Playground: We assume a network composed of $N$ sensors $\{s_1, \ldots, s_N\}$ uniformly distributed over a spherical region of radius $\rho$ with surface area $S$. While this choice of the network deployment area might appear unusual, using a sphere facilitates uniform coverage of the deployment area with random mobility models [19]. Moreover, UWSNs deployed over a sphere do not suffer from undesirable phenomena such as high variability in the average number of neighbors. Nevertheless, our study is general enough as to be applicable to any regular surface; the actual shape of the deployment area is not our primary focus.

Time: We assume that time is divided in equal rounds and sensors maintain loosely synchronized clocks. Round length corresponds to the periodicity of sensor data acquisition, i.e., at round $r$ sensor $s_j$ obtains data $d_j^r$.

Initialization: Before deployment, each sensor is initialized with: (1) the sink’s public key $PK$; (2) a fixed one-way hash function $H(\cdot)$ used as a pseudo-random number generator (PRNG); and, (3) a unique secret seed used to bootstrap the PRNG. This PRNG outputs sensor secrets used for data encryption and authentication. $K_j^r$ denotes $s_j$’s PRNG output at round $r$ and it is used for deriving all randomness needed during that round. Note that sensors have no public or private keys of their own. The only public key in the entire system is that of the sink – $PK$.

Sink and Re-initialization: The sink is a trusted party that
visits the network with certain frequency. Upon each visit, for each sensor, the sink obtains all data collected since the last visit, erases sensor memory, resets the round counter to 1 and provides a new unique secret seed to initialize the PRNG.

**Encryption:** Particulars of encryption depend on data size. If data, along with random padding, fits into one public key encryption block, then randomized public key encryption suffices [20]. In this case, we use $K^r_j$ to denote random padding used by $s_j$ at round $r$. Whereas, if padded data is longer than one public key block, hybrid encryption is used. This entails encrypting data with a symmetric algorithm under a one-time random key and then encrypting the latter under the sink’s public key. In this case, $K^r_j$ denotes the one-time key used by $s_j$ at round $r$. Regardless of how it is used, $K^r_j$ is obtained from $s_j$’s PRNG, i.e., depends on $K^0_j$. To abstract from the specifics of encryption, we use $E_{PK}(K^r_j, d^r_j, r, s_j)$ to denote the ciphertext produced by $s_j$ at round $r$.

Security is based entirely on secrets derived from the PRNG. We say that $E_{PK}$ is used, one-time key used by $S$ if they wander into the adversary-controlled area. In this paper, we focus on a more powerful adversary that selects, at any round, the next network location to compromise. This mobile adversary moves within the deployment area, and, once it reaches its destination, subverts all sensors within its compromise area.

**Goal:** $µADV$’s goal is to learn secrets of as many sensors as possible. Although exactly how sensors use their secrets is not important in this context, for the ease of exposition, we assume that sensors use their secrets to encrypt collected data and $µADV$’s ultimate goal is to decrypt the corresponding ciphertexts.

Finally, we assume that $µADV$ knows the network defense mobility patterns, i.e., [21]. Another reason for $µADV$ to be “stealthy” is that its presence, there is no way to tell if a given sensor has ever been compromised. Whereas, any modification to sensor code can be later discovered by the sink using attestation techniques, e.g., [21]. Another reason for $µADV$ to be stealthy is that it can benefit from longer-lasting recurrent attacks.

We stress that sensor mobility is assumed to be a pre-existing feature, motivated by reasons other than security, e.g., the need to ensure uniform coverage, load balancing, or fault-tolerance. We merely take advantage of existing mobility as a means to improve security. As shown later in the paper, we also consider static sensors as a baseline case where the network exhibits the lowest performance.

1It is computationally infeasible for the adversary to do so.

2Nonetheless, our protocol can be used with other mobility models. Its behavior with other mobility models is currently being investigated.

**B. Adversarial Model**

The mobile adversary ($µADV$) considered in UWSN-related literature [16] can be viewed as a distributed entity that compromises a fixed fraction of sensors per round, regardless of their location. From round to round, $µADV$ migrates and compromises different sets of sensors. The adversary considered thus far in Mobile UWSNs is static and passive. It cannot influence the choice of sensors to compromise at each round: sensors are compromised only if they wander into the adversary-controlled area. In this paper, we focus on a more powerful adversary that selects, at any round, the next network location to compromise. This mobile adversary moves within the deployment area, and, once it reaches its destination, subverts all sensors within its compromise area.

**Compromise Power:** At round $r$, $µADV$ controls a circular compromise area $S_o$ of radius $\rho_o$, centered in point $ap^o$ on the sphere. Any sensor $s_j$ located at a distance at most $\rho_o$ from $ap^o$ is assumed to be compromised. $µADV$ reads all of $s_j$’s storage/memory and listens to all incoming and outgoing communication. The set of compromised sensors at round $r$ is $\{s_j : D^o(ap^o, c^o_j) \leq \rho_o\}$. We note that $µADV$ is not a global eavesdropper and can only overhear messages sent or received within $S_o$. If $s_j$ is compromised at round $r$ and its secrets are either fixed or evolved (using a one-way function), $µADV$ can compute $s_j$’s secrets for round $r'$, if $s_j$ is later released (i.e., $D^o(ap^o, c^o_j) > \rho_o$). In other words, $s_j$’s behavior during rounds $r' > r$ becomes deterministic and $µADV$ can decrypt any ciphertext produced by $s_j$ from round $r$ onwards.

$µADV$ does not interfere with sensor behavior. While compromising sensors $µADV$ does not corrupt their state; its only goal is to learn their secrets. Since $µADV$ leaves no trace of its presence, there is no way to tell if a given sensor has ever been compromised. Whereas, any modification to sensor code can be later discovered by the sink using attestation techniques, e.g., [21]. Another reason for $µADV$ to be stealthy is that it can benefit from longer-lasting recurrent attacks.

Finally, we assume that $µADV$ knows the network defense strategy, while sensors are unaware of $µADV$ mobility patterns and history.

**Mobility:** $µADV$ moves according to the Random Jump mobility model detailed in Section III-A. That is, at round $r$ it picks a new position $ap^r$ and moves there from $ap^{r-1}$. Reached $ap^r$, it compromises all sensors within radius $\rho_o$ — sensors no longer in $S_o$ are out of $µADV$ control; however, unless healed, their state can be mimicked by $µADV$. 

---

Footnotes:

1. It is computationally infeasible for the adversary to do so.
2. Nonetheless, our protocol can be used with other mobility models. Its behavior with other mobility models is currently being investigated.
Recall that our main goal is to provide intrusion-resilience without the aid of any trusted third parties or secure hardware. The main idea is for sensors themselves to serve as a source of secure randomness for their peers. If secrets evolve in a pseudo-random fashion, $\mu_{ADV}$ can monitor future secrets of previously compromised sensors, because it learns the PRNG status of those sensors upon compromise. However, if a previously compromised sensor obtains some randomness that is unknown to $\mu_{ADV}$, that sensor can regain security and its new secrets are unknown to $\mu_{ADV}$. Thus, any sensor that either (1) has never been compromised, or (2) has regained security after compromise, can act as a source of secure randomness to its peers. An infusion of randomness from a non-compromised sensor, allows its previously compromised peers to move to a new secure state. The protocol takes advantage of sensor mobility to diffuse randomness throughout the network. Since a sensor is unaware of $\mu_{ADV}$’s location and can not determine whether it is (or has ever been) compromised, each sensor assumes the worst and proactively runs the healing protocol at every round.

Specifically, at round $r$, each $s_j$ moves according to the underlying mobility model. After reaching its new position, $s_j$ obtains data from the environment and encrypts it, as described in Section III-A. Then, $s_j$ broadcasts a random value based on its current PRNG status to immediate neighbors. Next, it uses all random values received from its neighbors, along with its current secret, to compute the next round secret. For example, $s_j$ computes its secret at round $r$ as: $K_j^r = H(K_j^{r-1}, C_j^{r-1})$, where $C_j^{r-1}$ is the set of contributions received by $s_j$ during round $r-1$ from $B(s_j, r-1)$. Clearly, if $\mu_{ADV}$ does not know $K_j^{r-1}$ and all values in $C_j^{r-1}$, it cannot compute the new secret $K_j^r$. In every round, each sensor sends one random value and receives (on average) $B$ random contributions.

## IV. THE PROTOCOL

Before analyzing the above protocol, we introduce our system abstraction. At any round $r$, all sensors can be partitioned into three disjoint sets:

- **Green** $G'$. Sensors with state unknown to $\mu_{ADV}$ (i.e., $\mu_{ADV}$ can not compute their secret in the current round). A green sensor either: (1) has never been compromised, or (2) was compromised and has since then regained security through our protocol. Any green sensor can help its immediate neighbors regain security.

- **Red** $R'$. Currently compromised sensors that are located within $S_a$, i.e., $R' = \{s_j|D^o(a, b) \leq \rho_a\}$. A red sensor cannot help its peers since its current state is known to $\mu_{ADV}$. Moreover, contributions from green peers is of no help to a red sensor.

- **Yellow** $Y'$. Sensors that have been compromised in the past and have since moved outside $S_a$, but have not yet regained security. A yellow sensor behaves deterministically from the $\mu_{ADV}$’s point of view, i.e., $\mu_{ADV}$ can compute its current secret. A yellow sensor can become green if its receives at least one contribution from a green peer. Any contribution by a yellow sensor is useless (in terms of regaining security) to its peers.

\begin{table}[h]
\centering
\caption{Notation Summary.}
\begin{tabular}{l|l}
\hline
$S$ & (spherical) deployment region/surface \\
$\rho$ & radius of $S$ \\
$r, r'$ & round indices \\
$N$ & number of sensors \\
$s_j$ & generic sensors \\
$d_{j, r}$ & data collected by $s_j$ at round $r$ \\
$K_j^r$ & $s_j$’s secret state at round $r$ \\
$K_j^p$ & padding used by $s_j$ at round $r$ \\
$c_j^s$ & $s_j$’s current position at round $r$ \\
$S_a$ & sensor communication area \\
$\rho_a$ & sensor communication range \\
$D^o(a, b)$ & orthodromic distance between points $a$ and $b$ \\
$B(s_j, r)$ & set of $s_j$’s neighbors at round $r$ \\
$B$ & mean numbers of neighbors \\
$\mu_{ADV}$ & mobile adversary \\
$\rho$ & adversary compromise range \\
$R^r$ & set of red sensors at round $r$ \\
$Y^r$ & set of yellow sensors at round $r$ \\
$G^r$ & set of green sensors at round $r$ \\
\hline
\end{tabular}
\end{table}

At any round $r$, $N = G^r + R^r + Y^r$. In our terminology, a sensor is called sick if it is either yellow or red. A green sensor is said to be healthy and a sensor that recovers secure state after compromise (becomes green) is said to be healed. Figure 1 shows how sensors change color and Figure 2 illustrates the state transition diagram. The meaning of dashed lines will become clear throughout the rest of the paper.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{reference_model.png}
\caption{Reference model.}
\end{figure}

To assess the performance of our protocol, we define two new metrics: Health Ratio ($\mathcal{H}_R$) and Duty Cycle ($D_C$). The former represents the fraction of healthy sensors at any round. It provides a snapshot of current network status and, as discussed below, it is round-independent. While $\mathcal{H}_R$ does not yet yield information about the evolution of a single sensor over time, $D_C$ captures the fraction of time a sensor is healthy. In fact, $D_C$ subsumes two other metrics: Time to Compromise ($T(tC)$) and the Time to Heal ($T(tH)$). $T(tC)$ is the number of rounds a sensor is green; in these rounds, $\mu_{ADV}$ cannot monitor sensor secret evolution, and hence cannot decrypt its messages. $T(tH)$ is the number of rounds it takes for a red
sensor to become green. We define $D_C = \frac{P_{IC}}{\rho - T_{HI}}$. Note that the natural main goal of any UWSN intrusion-resilience protocol is to have both $H_R$ and $D_C$ as close as possible to 1. In particular, $H_R \approx 1$ means that $\mu ADV$ does not learn secrets of most sensors, while $D_C \approx 1$ means that sensors are green in almost all rounds.

We now evaluate the proposed protocol with respect to $H_R$ and $D_C$. We show that network performance is related to both the average sensor neighborhood size and its variability (i.e., the number of new neighbors at any round). The latter is, in turn, influenced by the sensor mobility model and sensor density. The following analysis is based on simulated data [22]. However, due to space constraints, we only provide analytical details for the mobility models at the extremes of our mobility spectrum, i.e., Random Jump and the scenario where sensors are static. We consider a network of $N = 500$ sensors deployed over a sphere of radius $\rho = 10^3$. $\mu ADV$ moves according to RJ and has a compromise area ($S_a$) that constitutes 5%, 10% and 20% of the overall network deployment area, respectively.

VI. HEALTH RATIO

Based on the above abstraction, $\mu ADV$ aims to maximize the number of yellow sensors. Because of uniform sensor density, the number of red sensors can be approximated as a constant. For example, if $\mu ADV$ covers 20% of the sphere, around 20% of sensors, on average, are red. The network’s goal is to maximize the number of green sensors. Let $R^r$, $Y^r$, and $G^r$ be the mean number of red, yellow, and green sensors at round $r$, respectively. The Health Ratio at round $r$ $(H_R)$ becomes:

$$H_R = \frac{G^r}{G^r + Y^r}$$

In this equation, the number of red sensors is not considered, as it is strictly related to $S_a$ and independent of the security protocol.

The number of yellow sensors at round $r$ is:

$$Y^r = Y^{r-1} + R^{r-1}P_{RY} - Y^{r-1}P_{YG}$$

where $P_{RY}$, $P_{RY}$, and $P_{YG}$ are transition probabilities: red $\rightarrow$ yellow, yellow $\rightarrow$ red, and yellow $\rightarrow$ green, respectively. These probabilities are represented by solid lines in Figure 2.

From here on, we assume that the network is in a steady state, that is $R^r = R$, $Y^r = Y$, $G^r = G$, for any $r > 0$, and, consequently $H_R = H_R$, for any $r > 0$. As discussed above, the number of red sensors only depends on $S_a$ and can be expressed as:

$$R = N \cdot \frac{S_a}{S}$$

Equation (2) can be re-written as:

$$R \cdot P_{RY} - Y \cdot P_{RY} - Y \cdot P_{YG} = 0 \iff Y \cdot (P_{RY} + P_{YG}) = R \cdot P_{RY} \iff Y = R \cdot \frac{P_{RY}}{P_{RY} + P_{YG}}$$

Combining Eq. (1) with Eq. (3) yields:

$$H_R = \frac{G}{G + R \cdot P}$$

where $P = \frac{P_{RY}}{P_{RY} + P_{YG}}$.

When $H_R \approx 1$, i.e., sensors are either green or red, $P \approx 0$. Indeed, $P_{YG} \approx 1$ and sensors are healed immediately after exiting $S_a (Y \approx 0)$. When $H_R \approx 0$, the number of green sensors is nearly 0, and $P_{YG} \approx 0$.

A. Random Jump

Recall that, in RJ, a sensor can reach any location on the sphere in one round. Transition probabilities in Eq. (3) become:

$$P_{RY} = 1 - \frac{B}{S}; \quad P_{YG} = 1 - \frac{\{B(s_j,r) \cap G = \emptyset\}}{S}$$

where: (i) $B(s_j,r)$ is the set of $s_j$' neighbors at round $r$; (ii) $P_{RY}$ is the probability that a red sensor picks its next destination outside of $\mu ADV$'s compromise area ($S_a$); and, (iii) $P_{YG}$ is the probability that a yellow sensor picks its next destination within $S_a$.

$P_{YG}$ is the probability of a yellow sensor to be healed, computed as:

$$P_{YG} = 1 - \frac{\{B(s_j,r) \cap G = \emptyset\}}{S} = 1 - \left(1 - \frac{S_a}{S - S_a}\right)^G$$

In other words, $P_{YG}$ is the probability that, at round $r$, no green sensor winds up within the neighborhood $B(s_j,r)$ of the yellow sensor $s_j$. Combining Eq. (3) with Eq. (6), and considering that $\frac{S_a}{S - S_a} = \frac{S}{\mu ADV}$, yields:

$$Y \cdot \left(\frac{S_a}{S} + 1 - \left(1 - \frac{B}{N - R}\right)^Y\right) - R \cdot P_{RY} = 0$$

Equation 7 can be solved to express $Y$ as a function of $B$. Finally, observing that $G = N - R - Y$ and combining Eq. (4) with solutions of Eq. (7), we can compute $H_R$ as a function of $B$.

![Fig. 2. State transition diagram.](image-url)
When sensors move using RJ, the UWSN achieves a good healing rate, since sensors change their neighborhood, with high probability, at every round. In other words, a yellow sensor has a high chance of meeting a green peer and becoming healed. Any other model that reduces mobility and exhibits less variability in sensor neighborhood is expected to yield lower $H_R$. In the Random Waypoint (RP) mobility model, sensors move towards their current waypoint with constant speed. Once the current waypoint is reached, a new one is randomly chosen and the sensor starts moving again. As sensors take more rounds to reach their waypoint, they experience lower neighborhood variability and hence lower $H_R$. Figure 5 shows $H_R$ for different average neighborhood sizes, with the same adversarial configuration of Figure 3. A quick comparison of these two figures clearly shows that RP exhibits worse performance than RJ. For example, to get $H_R \approx 0.8$ when $\muADV$ controls 10% of the sphere, it takes $B = 4$, whereas, with RJ, $B = 1$ suffices. Also, the optimal healing rate is achieved only with a very large number of neighbors ($B \approx 30$), while, with RJ, $B \approx 5$ is enough.

Figure 6 shows the difference between the $H_R$ of the considered network scenario and the $H_R$ with a static adversary. The RP mobility model does not allow for great variability in sensor neighborhood. Thus, for some network configurations, the adversary benefits from mobility. Small differences for $B < 0.01$ are due to the fact that, with a very small neighborhood, both types of adversaries succeed in compromising the whole network. Similarly, small differences for $B > 10$ are due to the healing power of the intrusion-resilience protocol that can reach optimal $H_R$, no matter if $\muADV$ is static or mobile.
C. Static Sensors

Static sensors are expected to provide the worst healing ratio. Even if sensors are static, probability of transitions red ⇐ yellow are the same as the ones in Section VI-A, i.e., \( P_{RY} = 1 - \frac{S_e}{S_a} \) and \( P_{RY} = \frac{S_e}{S_a} \).

To compute \( P_{YG} = 1 - P(B(s_j, r) \cap G = 0) \) we refer to Figure 7. Assume \( \mu \) ADV is in position 1 at round \( r - 1 \) and jumps to position 2 at round \( r \). Let \( S^e_a \) be the circle of radius \( \rho_a - \rho_s \) centered at position 1. Also, let \( S^e_a \) represent the “donut” with outer and inner radii of \( \rho_a \) and \( \rho_s \), respectively, centered at position 1. All sensors that were red at round \( r - 1 \) become yellow at round \( r \) and have a chance to be healed. Yellow sensor \( s_j \) can heal if \( s_j \in S^e_a \) and it has at least one green neighbor (\( |B(s_j, r) \cap G| \geq 1 \)). Whereas, sensors within \( S^e_a \) cannot be healed because their current neighborhood is composed only of yellow peers.

The probability \( P_{YG} \) becomes:

\[
P_{YG} = 1 - \left[ P(s_j \in S^e_a) + P(s_j \in S^e_a')P(s_j \in S^e_a', B(s_j, r) \cap G = 0) \right]
\]

This yields:

\[
P_{YG} = 1 - \left[ \frac{S^e_a}{S_a} + \frac{S^e_a'}{S_a} \left( 1 - \frac{E(S^e_a)}{S - S_a} \right) \right]  
\]

where \( E(S^e_a) \) is the average area \( S^e_a \) of a yellow sensor in \( S^e_a' \). Figure 8 shows how to compute \( S^e_a \), assuming \( \rho_a \gg \rho_s \). That is, the adversarial boundary can be approximated by a straight line.

\[
S^e_a = \frac{\alpha}{2\pi} S_a - \rho_s (\rho_s - x) \sin \left( \frac{\alpha}{2} \right) = \alpha \rho_s^2 - \rho_s (\rho_s - x) \sin \left( \frac{\alpha}{2} \right)
\]

but \( \alpha = 2 \arccos \left( 1 - \frac{x}{\rho_s} \right) \), yielding:

\[
S^e_a = \rho_s^2 \arccos \left( 1 - \frac{x}{\rho_s} \right) - \rho_s (\rho_s - x) \sin \left( \arccos \left( 1 - \frac{x}{\rho_s} \right) \right)
\]

Then, \( E\{S^e_a\} \) can be evaluated as:

\[
E\{S^e_a\} = \frac{1}{\rho_s} \int_0^{\rho_s} S^e_a \, dx
\]

Finally, Equation (3) can be rewritten as:

\[
\mathcal{Y} \cdot \left( \frac{S_a}{S} + P_{YG} \right) - \mathcal{R} \cdot P_{RY} = 0
\]

Equation (9) can be solved to express \( \mathcal{Y} \) as a function of \( B \). Finally, observing that \( \mathcal{G} = N - \mathcal{R} - \mathcal{Y} \) and combining Eq. (4) with solutions of Eq. (9), we can compute \( H_R \) as function of \( B \).

\footnote{This assumption holds when \( \mu \) ADV’s compromise area is large with respect to sensors transmission range.}
Figure 9 shows the simulation results and our analysis for the static sensors scenario. Error-bars show quantiles 5, 50, and 95 of simulated data. Solid lines show the solutions of Eq. 9. The healing protocol performs worse compared to scenarios where sensors move and the $H_R$ reaches optimal values for $B \approx 40$. We do not compare results between $\mu_{ADV}$ and a static adversary as the latter is no match for a network of static sensors.

VII. DUTY CYCLE

In this section we focus on the color transitions of a sensor over time. The Duty Cycle is a random variable that captures the fraction of rounds a sensor is green, over the total number of rounds it takes to that sensor to perform a loop in Figure 2. $D_C$ consists of the Time to Heal $TtH$ and the Time to Compromise $TtC$:

$$D_C = \frac{TtC}{TtC + TtH} \quad (10)$$

If $D_C \approx 1$, a sensor is green for most rounds; if $D_C \approx 0$, a sensor is sick (i.e., red or yellow) most of the time.

A. Time To Heal

$TtH$ is a random variable that expresses the number of rounds for a red sensor to heal. For example, let $\{..., R, Y, G, G, G, R, \ldots\}$ be an arbitrary sequence of $s_j$ state transitions for $s_j$, then $TtH = 6$ for that sequence. Note that, if the healing protocol can not cope with $\mu_{ADV}$’s compromise power, a sensor can transition several times between red and yellow states, before it gets healed (if it meets a green peer).

To evaluate $TtH$, we assume $s_j$ gets compromised and, from that round on, we count the number of rounds it remains red or yellow. The system can be modeled with an absorbing Markov chain [23]. In particular, the states of the chain are equivalent to those described in the above coloring scheme, where we consider the green state as the absorbing one. The $3 \times 3$ matrix $M$ associated with the absorbing Markov chain can be expressed as:

$$M = \begin{bmatrix} P_{RR} & P_{RY} & 0 \\ P_{YR} & P_{YY} & P_{YG} \\ 0 & 0 & 1 \end{bmatrix}$$

Probabilities $P_{YR}, P_{RY}$, and $P_{YG}$ were shown in Section VI. From Figure 2, we have $P_{RR} = 1 - P_{RY}$ and $P_{YY} = 1 - P_{YG} - P_{YR}$. Further, $M$ can be partitioned as:

$$M = \begin{bmatrix} Q & A \\ 0 & I \end{bmatrix}$$

where $Q$ captures the transition probabilities between transient states; $A$ is the probability of transition from a transient state to an absorbing state, and, finally, $I$ and 0 are the identity and the null matrices, respectively. The expected number of rounds to reach the absorbing state can be computed as:

$$D = (I - Q)^{-1} \quad (11)$$

Vector $D = [d_1, d_2]$ provides average absorbing times when the chain starts from the red or yellow state, respectively. Since we are interested in the time between compromise and healing, (i.e., the chain always starts from red), $d_1$ represents the $TtH$.

B. Time To Compromise

$TtC$ is a random variable that provides the number of rounds it takes to a green sensor to be compromised. For example, let $\{..., R, Y, G, G, G, R, \ldots\}$ be an arbitrary sequence of $s_j$ state transitions; $TtC = 3$ because it takes 3 rounds for $s_j$ to be compromised since the last time it was healed. Though $TtC$ is independent of the healing protocol, it depends on both the sensors and the adversary mobility models. Let $T_c$ be the random variable associated to the sequence of $k$ consecutive rounds for which a sensor is green ($TtC$). Thus:

$$P(T_c = k) = P_{GG}^k P_{GR}$$

Recalling that $P_{GG} + P_{GR} = 1$, yields: $P(T_c = k) = P_{GR}(1 - P_{GR})^k$. The mean value of the random variable $T_c$ can be expressed as:

$$E\{T_c\} = \sum_{k=0}^{\infty} k \cdot P(T_c = k) = \sum_{k=0}^{\infty} k \cdot P_{GR}(1 - P_{GR}^k) = 1 - \frac{P_{GR}}{P_{GR}}$$

Since $P_{GR} = \frac{S}{S_a}$ for both static and mobile scenario, we have:

$$E\{T_c\} = \frac{S}{S_a} - 1 \quad (12)$$

Figures 10, 11, and 12 show the Duty Cycle of a sensor as it moves according to RJ mobility model, RP mobility model, or when it is static, respectively. Once again, RJ represents the best alternative for collaborative intrusion-resilience. Comparing results of $\mu_{ADV}$ with the ones for a static adversary, we do not observe any appreciable difference when sensors move according to RJ. Differences between the two adversarial
scenarios when sensors move according to RP are shown in Figure 13. It clearly demonstrates the benefit of mobility from \( \muADV \)’s point of view. Once again we do not look at the scenario where both sensors and the adversary are static.

**VIII. DISCUSSION**

The intrusion-resilient collaborative self-healing protocol presented in this paper depends on the following three parameters: (i) portion of the deployment surface controlled by \( \muADV \); (ii) sensor mobility model; and, (iii) mean number of neighbors \(- B \). We varied \( B \) by changing the sensor transmission range. Both analyses and simulations showed that the mobile adversary can be mitigated by adopting a sensor mobility model that provides high variability in sensor neighborhoods. Indeed, if sensors change their neighborhood at every round, cooperation is a very effective means of intrusion-resilience: this is why RJ provides better performance than RP.

Table II compares static and mobile adversaries when \( S_a = 0.1 \). We only present three sample values of average neighborhood size, i.e. \( B = \{0.1, 1, 10\} \). Performance is determined by the metric introduced earlier in the paper: \( \mathcal{H}_R \) and \( D_C \). The average neighborhood size \( B \) turns out to be a critical parameter for the proposed protocol, i.e., in all scenarios, it is possible to achieve better performance in terms of \( \mathcal{H}_R \) and \( D_C \) by increasing the sensor transmission range.

Table II also shows that static and mobile adversaries behave in a completely different fashion: decreasing mobility when the adversary is static increases healing. Indeed, low network mobility does not allow the adversary to compromise new sensors. Therefore, \( \mathcal{H}_R \) and \( D_C \) performance tends to be better. Whereas, with \( \muADV \), lower sensor mobility reduces \( \mathcal{H}_R \) and \( D_C \) values. Intuition suggests that, if the adversary moves, sensors must move as well to avoid compromise. However, this turns out not to hold: increasing mobility increases the healing property of the network only because higher numbers of mobile compromised (yellow) sensors imply higher likelihood of them being healed by green peers.

**IX. CONCLUSION**

In this paper, we continued exploring intrusion-resilience in recently introduced Mobile UWSNs. We introduced a new adversarial model, and investigated how sensor mobility affects self-healing. We also defined two new metrics (Health Ratio and Duty Cycle) that characterize the overall security and behavior of the network. Both analytical and simulation.
results show that sensor mobility is an effective and efficient means of self-healing in the presence of a mobile adversary.

ACKNOWLEDGEMENTS

Claudio Soriante’s research was partially supported by the Spanish National Science Foundation (MICINN) under grant TIN 2010-19077, the Madrid Regional Research Council (CAM) under the CLOUDS project (S2009/TIC-1692) and EU structural funds, and the European Commission under the MASSIF project (FP7-257475).

G. Oligeri’s research was partially supported by the EU 7th framework program, contract no. 247950 (UNIVERsal open platform and reference Specification for Ambient Assisted Living).

REFERENCES


