United We Stand: Intrusion-Resilience in Mobile Unattended WSNs

Roberto Di Pietro, Gabriele Oligeri,
Claudio Soriente, *Member, IEEE*,
and Gene Tsudik, *Member, IEEE*.

Abstract—Wireless Sensor Networks (WSNs) are susceptible to a wide range of attacks due to their distributed nature, limited sensor resources, and lack of tamper-resistance. Once a sensor is corrupted, the adversary learns all secrets. Thereafter, most security measures become ineffective. Recovering secrecy after compromise requires either help from a trusted third party or access to a source of high-quality cryptographic randomness. Neither is available in Unattended Wireless Sensor Networks (UWSNs) where the sink visits the network periodically. Prior results have shown that sensor collaboration is an effective but expensive means of obtaining probabilistic intrusion-resilience in static UWSNs.

In this paper, we focus on intrusion-resilience in Mobile Unattended Wireless Sensor Networks ($\mu UWSNs$) where sensors move according to some mobility models. Note that such a mobility feature could be independent from security (e.g. sensors move to improve area coverage). We define novel security metrics to evaluate intrusion resilience protocols for sensor networks. We also propose a cooperative protocol that – by leveraging sensor mobility – allows compromised sensors to recover secure state after compromise. This is obtained with very low overhead and in a fully distributed fashion. Thorough analysis and extensive simulations support our findings.

Index Terms—WSN security, intrusion resilience, distributed adversary, self-healing, mobility.

1 Introduction

Many current and envisaged applications for Wireless Sensor Networks (WSNs) involve data collection in remote, inaccessible or hostile environments, such as deserts, mountains, ocean floors and battlefields. A multitude of sensors might be deployed within a certain area and their activity is usually monitored and managed by a powerful trusted entity, commonly referred to as the *sink*.

Security in WSNs presents several well-known challenges stemming from all kinds of resource constraints of individual sensors. However, the main limitation that complicates sensor security techniques is lack of ubiquitous (inexpensive) tamper-resistant hardware. Lack of secure storage forces sensors to store cryptographic material, such as keys and seeds, in regular memory. Some recent work [1] showed that commodity sensors can be easily compromised, even without physical access [2]. With compromise, the adversary can read the sensor program memory and storage. As a result, no matter which security techniques are in use, sensor compromise reveals all of its secrets to an adversary. From that moment on, any cryptographic protocol ceases to be effective. For example, if the sensor routinely encrypts measurements using a secret key shared with the sink via a symmetric encryption algorithm (e.g., AES), the adversary that subverts the sensor learns the secret key and can decrypt any ciphertext produced by its victim. If the key is used for integrity purposes (e.g., via HMAC [3]) the adversary may fabricate arbitrary measurements. Based on the time of corruption, the security state of a given sensor can be partitioned in three epochs: (1) time before corruption; (2) time during corruption; and, (3) time following corruption. Nothing can be done about security in epoch 2 as the adversary controls the sensor, while enforcing security in epochs 1 and 3 requires forward and backward secrecy, respectively. Informally, a cryptographic protocol is *forward secure* if exposure of secret material at a given time does not lead to compromise of secrets for any time preceding compromise. Whereas, a cryptographic protocol is *backward secure* if compromise of secret material at a given time does not lead to compromise of any secret to be used in future.

It is well-known that forward secrecy can be easily obtained by periodically evolving a secret (e.g., a key), using a one-way function. If we assume that time is divided in rounds and let $\mathcal{K}^0$ be an initial secret shared with the sink, at round $r \geq 1$ a sensor computes the secret for the current round, $\mathcal{K}^r = H(\mathcal{K}^{r-1})$, where $H(\cdot)$ is a one-way function. As the sink knows the initial secret $\mathcal{K}^0$, it can mimic the secret evolution process and compute the sensor secret for any round. If the adversary learns secret $\mathcal{K}^r$, it can compute secrets for round $r' > r$, but it cannot compute any secrets used in prior rounds. Note that this is possible even if the adversary is no longer in control of the sensor in round r'.

R. Di Pietro is with Università degli Studi Roma Tre, Italy.
G. Oligeri is with University of Trento, Italy.
G. Tsudik is with Univeristy of California at Irvine, California, USA.
C. Soriente is with ETH Zurich, Switzerland.

Backward secrecy is much more challenging, since knowledge of $\mathcal{K}^r$ allows the adversary to compute secrets for future rounds. It would be trivial to obtain backward secrecy if each sensor had a True Random Number Generator (TRNG). Because a TRNG yields information-theoretically independent values, even if the adversary learns many (but not all) TRNG outputs, it cannot compute the missing values, whether they correspond to the past or to the future. In other words, when the adversary compromises the sensor, it cannot learn past secrets; once the adversary leaves the sensor it will not be able to compute future sensor secrets.

Unfortunately, TRNGs are not found on commodity sensors and not expected to be available for the near future. An alternative to per-sensor TRNGs is the presence of a Trusted Third Party (TTP); this is assumed in key-insulated schemes [4], [5]. In such schemes, forward and backward secrecy is achieved by having end-devices (e.g., the sensors) evolve their secrets in cooperation with a TTP, called a *base*. Unless both the end-device and the base are compromised at the same time, per-round keys are *insulated*. Key-insulated schemes are well-matched for WSNs with a constantly present sink, where the latter acts as a base.

However, there are settings where the constant presence of a sink is not viable. If the deployment area is large (e.g., a national border) or adverse (e.g., the ocean floor), an on-line sink might not be feasible. Moreover, if the area is hostile, a fixed sink could be a single point of failure: neutralizing the sink, the whole WSN becomes useless. In all these settings, the sink may be an itinerant entity that would rather visit the network at irregular intervals to collect sensor measurements and perform maintenance (e.g., software update).

This *modus operandi* has been introduced under the name Unattended Wireless Sensor Networks (UWSNs) [6]–[8]. In an UWSN, the sink is an itinerant entity that leaves the network unattended for possibly long periods of time. During sink absence, sensors have no access to a TTP and must cope with a broad range of attacks, while relying on their own meager resources. Since a UWSN cannot take advantage of any key-insulated protocol, backward secrecy can only be achieved via alternative techniques. In static UWSNs, sensor collaboration was shown to be one such effective (but expensive) technique [9], [10]. Sensors exchange pseudo-random contributions, i.e., values drawn from their pseudo-random number generator, and use received contributions – along with their current secrets – to compute future secrets. This way, if a previously-compromised sensor obtains at least one random contribution unknown to the adversary, it regains secrecy. However, protocols proposed in [9], [10] are quite energy-consuming due to sending/receiving of random contributions to/from peers. Moreover, prior collaborative solutions aimed at mitigating a mobile adversary in static UWSNs, suffer from the adversary's ability to eavesdrop on contributions exchanged by sensors. This is because communication is multi-hop and, if one sensor on the communication chain is under adversarial control, all exchanged messages are trivially observable (furthermore, using encryption is of no help, as argued in [9]).

Contributions: In this paper we investigate collaborative intrusion-resilience in Mobile UWSNs ($\mu UWSNs$) where unattended sensors migrate within a fixed deployment area and gather environmental data waiting for the sink to approach the network and to collect them. Our ultimate goal is to design techniques that enable sensors to recover secrecy of their cryptographic material (e.g., keys) after compromise. In particular, we study the impact on collaborative intrusion-resilience of sensor mobility models and number of regions controlled by the adversary.

To reach this goal, we first introduce general metrics to assess the effectiveness of intrusion-resilient protocols for $\mu UWSNs$ and later propose a collaborative distributed protocol that leverages sensor cooperation and locomotion to achieve probabilistic key-insulation. Sensors take advantage of mobility and collaboration with peers to regain secrecy after having been compromised by inadvertently wandering into the area under adversarial control. Using both analytical and simulation results, we show that the proposed protocol provides probabilistic key-insulation without any trusted third parties or secure hardware and with minimal overhead. Note that the assurance on the probability to regain key secrecy is a system parameter that can be expressed as a trade-off between security objective and sustained overhead.

Organization. Next section surveys related work in the area and Section 3 introduces the mobility and the adversarial models considered. Section 4 presents our framework and metrics to analyze intrusion-resilience in sensor networks. Section 5 introduces a collaborative intrusion-resilience protocol that is analyzed in Section 6. Analysis is refined in Section 7 and Section 8 for a centralized and a distributed adversary, respectively. Discussion follows in Section 9. Finally, Section 10 reports some concluding remarks.

2 Related Work

Some prior work has considered key exposure following sensor compromise. Dutta, et al. [11] proposed a constant storage self-healing protocol for WSNs. Sensor key update uses a polynomial-based secret sharing scheme, performed with the help of the sink. The sink periodically broadcasts information to allow non-revoked sensors to update their current session key. At any time, sensors can be revoked and prevented from learning keys of any sessions after revocation. Since this protocol relies on the constant presence of a sink, it is not applicable to UWSNs. WHISPER [12] provides both backward and forward secrecy for keys shared between any two sensors. Session keys are computed from two secrets, provided by each party, i.e., the key for session r between

s_j and s_q is computed as $K_{jq}^r = F\left(H(K_j^{r-1}), H(K_q^{r-1})\right)$, where K_j^{r-1} and K_q^{r-1} are s_j and s_q secrets for session $r-1$ and $F(\cdot)$ and $H(\cdot)$ are suitable hash functions. The scheme is secure as long as the adversary does not compromise both s_j and s_q. This assumption does not hold in UWSNs as their unattended nature allows the attacker to gradually compromise some (even all) sensors between successive sink visits.

Recently, mobile WSNs have began to attract attention because of the advantages that mobility brings to sensing applications [13]. If sensors move, the network can guarantee optimal area coverage, even if precise sensor deployment is infeasible (e.g., because of hostile or inaccessible conditions of the deployment area) [14]. Also, mobility helps to solve network connectivity problems caused by sensor failures and allows sensors to adapt their sampling power to respond to precise events [15]. Moreover, mobile sensors can extend sensor lifetimes bringing energy to sensors with depleted batteries [16]. Finally, mobility is currently being investigated as a means to detect sensor capture attacks [17], [18], and [19].

In the last few years, UWSNs have become subject of some attention. The initial work [7] introduced the UWSN scenario, defined the mobile adversary and investigated simple techniques to counter attacks focused on erasing specific data. This was later extended [6] to include the case where the adversary's goal is to indiscriminately erase all sensor data. Another recent result [20] introduced simple cryptographic techniques to prevent the adversary from recognizing data that it aims to erase. Sensor cooperation to achieve self-healing in static UWSNs is explored in [9] and [10]. Self-healing in our scenario has been studied in [21] and [22] in presence of a centralized, static adversary and a mobile adversary, respectively. This paper extends previous results assessing the impact of a distributed adversary on self-healing of mobile UWSNs.

3 System Assumptions

This section provides details of the network environment and the adversarial model.

3.1 Network Environment

The envisioned $\mu UWSN$ includes $\mathbf{N} = \{s_1, \ldots, s_N\}$ sensors.

Deployment area: We consider a network deployed over a sphere of radius ρ with surface area S. A spherical surface provides uniform coverage of the deployment area with random mobility models [23]. However, we stress that the shape of the deployment area is not the focus of our work. Our techniques can be applied to $\mu UWSN$ deployed on any fixed-area surface: uniform coverage only helps our analysis.

Time: Time is divided in rounds and all sensors' clocks are loosely synchronized, e.g., via [24]. Round length can be arbitrary; we assume that it reflects a single acquisition of data from the environment, i.e., sensors obtain measurements once per round, that is, at round r sensor s_j obtains data d_j^r.

Initialization: Before deployment, each s_j is initialized with: (1) the sink public key PK; (2) a common cryptographic hash function $H(\cdot)$ used as a pseudo-random number generator (PRNG); and, (3) a unique secret seed to bootstrap its PRNG. The PRNG is invoked for all random choices made by the sensor and its status is updated at each invocation — status at round r for sensor s_j is denoted with $\mathcal{K}_j^r$.

Sink Visits and Re-initialization: The sink is an itinerant trusted party that visits the network with a certain frequency. Upon each visit, the sink obtains collected measurements from every sensor, erases sensor memory, provides a fresh initial secret seed for the PRNG and resets the round counter to 1.

Security: Sensor secrets are fundamental to the provisioning of several security services, such as data confidentiality and authentication. The protocol introduced in this paper allows sensors to regain secret status after compromise and is not concerned with usage of sensor secrets. However, to ease exposition we will focus on a concrete example. That is, we assume that secrets are used to generate padding values to achieve public-key randomized encryption. Although in the recent past public key encryption was shunned by the sensor security community because of its high cost, novel developments make public key encryption feasible on commodity sensors [25], [26]. Further, the reason why *we are using public key encryption* when symmetric encryption is cheaper in all respects, is that using public key allows the sink to seamlessly decrypt anything that sensors encrypt (for it) in any round. Indeed, as discussed below, the security is based on the use of secret padding or randomizers [27] and not on the mere use of public key encryption. In contrast, if we were to use symmetric encryption, it would be quite hard (and in some specific cases even impossible) for the sink to decrypt data. This topic is discussed in detail in [9], [20], and [8].

In practice, details of data encryption depend on data size. If data (along with randomized padding) fits within a single public key encryption block (e.g., 160 bits for ECC or 1024 bits for RSA), then public key encryption suffices. However, if data exceed the block size, hybrid encryption becomes necessary. This entails encrypting data using a symmetric encryption algorithm (e.g., AES) with a one-time random key K_j^r (where j corresponds to s_j and r is the current round). Then, K_j^r is itself encrypted using the sink's public key PK.[1] If hybrid encryption is used, security is determined by the secrecy of K_j^r. Whereas, if pure public key encryption is used,

1. We emphasize that sensors do not have their own public/private keys and do not perform any public key decryption or any other public key operation, apart from encryption under PK. That is, there is a single public key in the entire network.

security is based on the secrecy of randomized padding,[2] (for convenience, we also refer to it as K_j^r). Regardless of how K_j^r is used, it is obtained from s_j's PRNG. To abstract away from the specifics, we use $E_{PK}(K_j^r, s_j, r, d_j^r)$ to denote ciphertext of d_j^r produced by s_j at round r using random padding K_j^r.

Mobility: Sensor s_j starts at position cp_j^0 and moves over the deployment area according to a network-wide mobility model. We consider two mobility models:

- **Random Jump Mobility Model (RJ):** each sensor sets its speed so it can reach any point of the sphere in one round. Starting with round $r = 1$ and initial position cp_j^0, s_j chooses a random point wp_j^r and moves there atomically.
- **Random Waypoint Mobility Model (RP):** all sensors move with the same constant speed and can cover at most distance m in a single round. At round 1, s_j at position cp_j^0 chooses a random point wp_j^1 and gradually moves there in $\left\lceil \frac{D^o(cp_j^0, wp_j^1)}{m} \right\rceil$ rounds, where $D^o(cp_j^0, wp_j^1)$ is the orthodromic distance between cp_j^0 and the waypoint wp_j^1. Once s_j reaches wp_j^1, it picks a new waypoint and starts moving towards it.

Both models are based on the random selection of the latitude θ and the longitude ϕ of waypoints, according to the "*trig*" method [28]. Algorithms 1 and 2 show the pseudo-code run by each sensor s_j, at any round r. In Algorithm 2, the function $\texttt{Move}(cp, wp, m)$ computes the next position of a sensor, at distance m from the current position, towards the current waypoint.

Algorithm 1: RANDOM-JUMP()

$cp.\phi \xleftarrow[\$]{1} [-\pi, \pi]$;
$rnd \xleftarrow[\$]{1} [-1, 1]$;
$cp.\theta = \frac{1}{\pi} \arccos(rnd)$;
`return`(cp);

We chose RJ and RP as mobility models since they have been extensively studied and are known to provide uniform coverage of the deployment area. Nevertheless, the proposed protocol may be applied with any mobility model.

Note that increasing m, RP tends to RJ, but a detailed analysis on the influence of the step size m is out of the scope of this paper. Further, we stress that sensor mobility is assumed to be an existing feature of the network motivated by reasons other than security, e.g., coverage, load balancing, or fault-tolerance. We simply take advantage of mobility to attain better security.

2. Note that we cannot rely on the secrecy of the collected data; it might be predictable or be drawn from just a small set of possible values.

Algorithm 2: RANDOM-WAYPOINT(cp,wp,m)

let cp be the current position
let wp be the current waypoint
let m be the step length
if $(cp == wp)$ **then**
  $wp.\phi \xleftarrow[\$]{1} [-\pi, \pi]$;
  $rnd \xleftarrow[\$]{1} [-1, 1]$;
  $wp.\theta = \frac{1}{\pi} \arccos(rnd)$
end
$cp = \texttt{Move}(cp, wp, m)$;
`return`$([cp, wp])$;

Communication: Each s_j has a spherical-cap shaped communication area S_s, with radius ρ_s. At round r, sensor s_j can communicate with s_p if $D^o(cp_j^r, cp_p^r) \leq \rho_s$, i.e., $s_p \in \mathcal{B}(s_j, r)$, where $\mathcal{B}(s_j, r)$ is the set of neighbors of s_j at round r. Let $B^r = E[\mathcal{B}(s_j, r)]$ be the average number of neighbors of s_j at round r. Since sensors are always uniformly distributed on the sphere, this yields $B = B^r = N \cdot \frac{S_s}{S},\ r > 0$.

3.2 Adversarial Model

The UWSN model considered in prior work assumes a mobile adversary that migrates among different subsets of compromised sensors. In our $\mu UWSN$ setting, sensors are mobile, while the adversary is static.

This latter operating hypothesis, other than being worth investigating on its own, is also motivated by the fact that the adversary might not have enough "resources" to move or there might just be no incentive for it to be mobile, i.e., it might as well be stationary and wait for sensors to move to its controlled area. Previous work [22] has shown that the adversarial mobility model has no or very little impact on the network performance in terms of resiliency, when sensor are mobile. Hence, in this paper we focus on the impact on self-healing of a distributed, static adversary.

Further, the envisioned adversary differs from other adversarial models considered in most prior WSN security literature. The latter is static in terms of the set of sensors it corrupts, i.e., it compromises k out of n sensor throughout the network lifetime. Our adversary (ADV) is stationary with respect to the portion of the deployment area it controls; but, the set of compromised sensors changes as nodes move in and out of the adversary-controlled area. Another unique feature characterizing our adversary is its *degree*, as explained in the following.

Adversarial Degree: ADV is either centralized or distributed. In any case, it has an overall compromising area S_{ADV} that is partitioned in one or more equally-sized, non-overlapping *compromising regions*. ADV^A denotes an adversary with *degree* A, that is, distributed on A compromising regions. Each compromising region is a

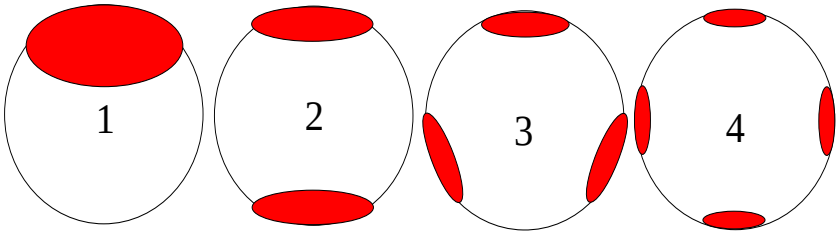


Fig. 1. Adversary layouts. Centralized adversary: ADV^1 (1), and Distributed adversaries: ADV^2 (2), ADV^3 (3), and ADV^4 (4).

spherical cap with center ap_a, surface S_a, and range ρ_a, for $1 \leq a \leq 4$.

Figure 1 shows four considered layouts.[3] The centralized adversary (ADV^1) is placed at the north pole of the sphere, whereas, in other cases ($A = 2, 3$, and 4), the adversary is distributed and placed according to Fig. 1. This paper does not investigate the impact of ADV's position: we placed the compromising regions such that the distance among them is maximal, in order to minimize their mutual influence.

Compromising Power: ADV^A compromises all sensors within its range, i.e., s_j is compromised at round r if $D^o(cp_j^r, ap_a) \leq \rho_a$, for any $a \leq A$. For each compromised s_j, the adversary reads all s_j's storage/memory and eavesdrops on all incoming and outgoing communications. A compromised sensor is released as soon as it moves away from all the compromising regions, i.e., $D^o(cp_j^r, ap_a) > \rho_a$, for all $a \leq A$, We assume that the adversary is not a global eavesdropper and can only eavesdrop on its compromising regions. We stress that ADV does not interfere with sensors' behavior, and can be described as a *read-only* adversary. A number of techniques allow to discover sensor compromise when the adversary modifies the sensor code [29]–[31]. Hence, if the adversary is limited to "read-only" attacks and keeps the sensor code unchanged, there is no way to tell whether that sensor has ever been compromised. This allows ADV to stay undetected and benefit from repeated attacks to the network. Finally, we assume that ADV is aware of the network defence strategy while neither the sensors nor the sink know ADV's location.

Table 1 summarizes the notation used throughout the paper.

TABLE 1
Notation Summary.

S	spherical region/surface
ρ	radius of S
r, r'	round indices
$\mathbf{N} = \{s_1, \ldots, s_N\}$	set of sensors
N	size of $\mathbf{N}$
d_j^r	data collected by s_j at round r
$\mathcal{K}_i^r$	s_j' secret state at round r
K_i^r	random padding by s_j at round r
cp_j^r	s_j's current position at round r
wp_j^r	s_j's current waypoint at round r
S_s	sensor communication area
ρ_s	sensor communication range
$D^o(a, b)$	orthodromic distance between points a and b
$\mathcal{B}(s_j, r)$	set of s_j's neighbors at round r
B	mean numbers of neighbors
ADV	adversary
ap	adversary position
ρ_a	adversary compromising range
$\mathcal{R}^r$	set of red sensors at round r
$\mathcal{Y}^r$	set of yellow sensors at round r
$\mathcal{G}^r$	set of green sensors at round r

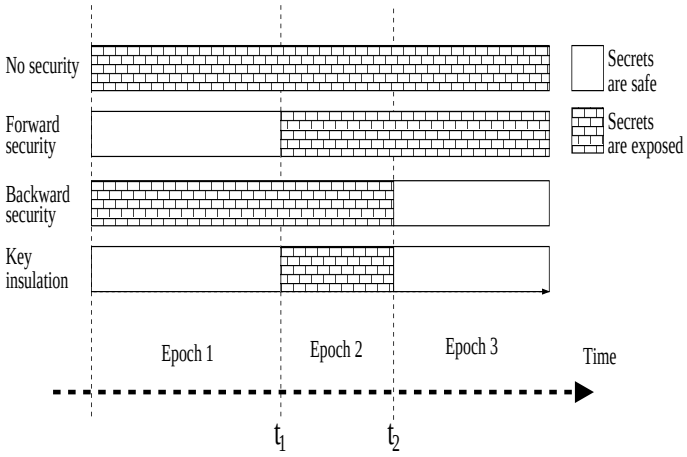


Fig. 2. Epochs and security properties.

4 Model and Metrics for Key-Insulation

Figure 2 summarizes security epochs and properties introduced in Section 1. The dashed arrow shows time and the moments when the adversary compromises (t_1) and releases (t_2) the victim sensor, respectively. Bricks represent time intervals when sensor secrets are not known to the adversary, while gray rectangles refer to time periods when sensor secrets are exposed. Let us assume the adversary learns the secrets during compromise (from t_1 to t_2). If the security protocol is neither forward nor backward secure, past (before t_1) and future (after t_2) secrets are exposed as well. If the security protocol is forward secure, the adversary also learns secrets after t_2, while if the protocol is backward secure, the adversary learns secrets used up to t_1. A key insulated protocol, i.e., our ultimate goal, prevents the adversary to compute secrets used before t_1 and after t_2. Based on sensor compromise and the adversary knowledge of its secrets,

3. We consider up to 4 compromising regions.

the set of sensors can be partitioned into three distinct groups at any round:

- **Red Sensors ($\mathcal{R}^r$)**: a sensor s_j is red if it is currently compromised (i.e., $cp_j^r \in S_a$) and its secrets are exposed to the adversary.
- **Yellow Sensors ($\mathcal{Y}^r$)**: a sensor s_j is yellow if it is not currently compromised (i.e., $cp_j^r \notin S_a$), but ADV still knows its secrets for the current round.
- **Green Sensors** $\mathcal{G}^r$: a sensor is green if its current secrets are unknown to ADV. This is because either it has never been compromised or because it has recovered secrecy via the key-insulated protocol.

When it becomes clear from the context, we will use the same notation to denote a set (i.e., $\mathcal{R}^r,\ \mathcal{Y}^r,\ \mathcal{G}^r$) and its size. In the following we refer to green sensors as *healthy* and to red or yellow sensors as *sick*. The knowledge of the sensor's secrets allows ADV to perform several attacks, ranging from sensor impersonation to compromising confidentiality of sensed data. Main goals of the adversary are: either to minimize the number of green sensors, or to keep a specific sensor compromised for as long as possible.

To assess the effectiveness of a generic key-insulated protocol we define two new metrics: *Health Ratio* ($\mathcal{H}_R$) and *Healthy Cycle* ($\mathcal{H}_C$). The former represents the network healthiness as the number of the green sensors, while the latter represents the number of rounds a sensor is green over its lifetime. The natural goal of any intrusion-resilient protocol is to have both $\mathcal{H}_R$ and $\mathcal{H}_C$ as close as possible to 1. In particular, $\mathcal{H}_R \approx 1$ means that secrets of almost all sensors are not exposed, while $\mathcal{H}_C \approx 1$ means that each sensor is green for most part of its lifetime.

5 The Protocol

In our protocol, forward secrecy is (predictably) obtained with periodic secret evolution using PRNG $H(\cdot)$. To obtain backward secrecy, the main idea is for sensors to serve as a source of randomness for their peers. A sensor that resides outside the area controlled by ADV, but whose secrets are exposed (that is, a yellow sensor), can regain security and move to a new secure state (i.e., become green) if it obtains at least one contribution of secure randomness from a peer sensor whose secret state is not exposed (green sensor). As the adversary eavesdrops on red sensors, their received contributions are observable, so they cannot regain secrecy. Our protocol leverages mobility to bring computationally secure randomness to yellow sensors. Since ADV's location is unknown and sensors cannot distinguish between compromised and non-compromised peers, the protocol is proactively run by all sensors.

At round r, each s_j runs Algorithm 3: it moves according to the adopted mobility model (`Move`()), and, after reaching its new position, senses data from the environment (`Read`()). The latter is encrypted under the sink public key and stored locally. Function `PadGen`$(\cdot)$ uses the sensor's current secret state to generate an encryption padding. At that time, s_j broadcasts a random value drawn from its secret state (`PadGen`$(\cdot)$) and collects randomness sent by its neighbors. Secret state is updated with all received random contributions before moving to the next round.

Algorithm 3: Collaborative Intrusion-resilient Protocol

```
Move() ;
d_j^r = Read() ;
K_j^r = PadGen(K_j^r);
Store(E_PK(K_j^r, d_j^r, r, s_j)) ;
R_j^r = [∅] ;
c = 0;
t = RandGen(K_j^r) ;
Broadcast(t);
while (roundTimer) do
    Receive t_p^r from s_p ;
    R_j^r[c] = t_p^r ;
    c = c + 1;
end
K_j^{r+1} = H(K_j^r || R_j^r[0] || ... || R_j^r[c-1]) ;
Delete(K_j^r, K_j^r) ;
```

6 Analysis

To support our analytical findings with experimental results, we developed a software simulator [32] for the spherical deployment area. In all our simulations, the $\mu UWSN$ contains $N = 500$ sensors moving over a sphere of radius $\rho = 10^5$. Step size for RP is set to $m = 2,000$. Sensor transmission range ρ_s is chosen such that $\frac{S_s}{S}$ ranges in $[10^{-6}, \ldots, 10^{-2}]$. As the protocol of Section 5 is based on sensor cooperation, the number of neighbors is a key factor and sensor transmission range dramatically influences its performance. Neighborhood size can be tuned either via sensor density or via sensor communication range: we chose to fix the former and vary the latter. ADV is randomly placed on the sphere and the range of its compromise regions (ρ_a) is chosen such that its overall compromise area S_{ADV} is 0.05, 0.1 and 0.2, of the spherical surface, respectively.

Figure 3.a shows our scenario: (1) a green sensor remains green until it moves at distance less than or equal to ρ_a from ADV; (2) a red sensor cannot become green without becoming yellow first as ADV eavesdrops on red sensors; and, (3) a yellow sensor can become green only if it receives at least one contribution from a green sensor.

Figure 3.b depicts the state transitions diagram. The three probabilities that characterize a sensor state transitions and are relevant to our analysis are P_{RY}, P_{YR}, and P_{YG}; other probabilities can be computed from the previous ones. We consider two different adversarial models (see Fig. 4), i.e., centralized and distributed. For each of them, we consider two network mobility models, i.e. *Random Jump* (RJ) and *Random Waypoint* (RP).

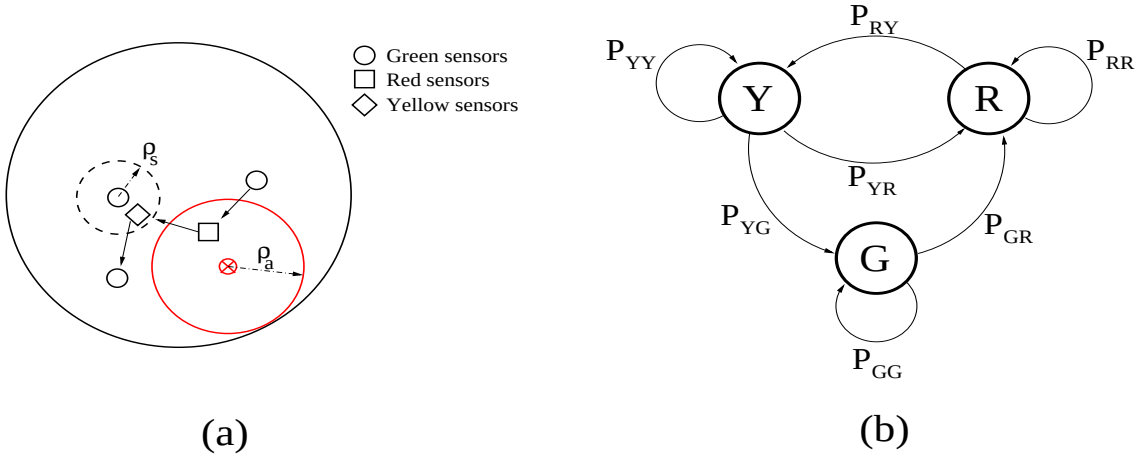


Fig. 3. Reference Scenario (a) and Transition Diagram (b).

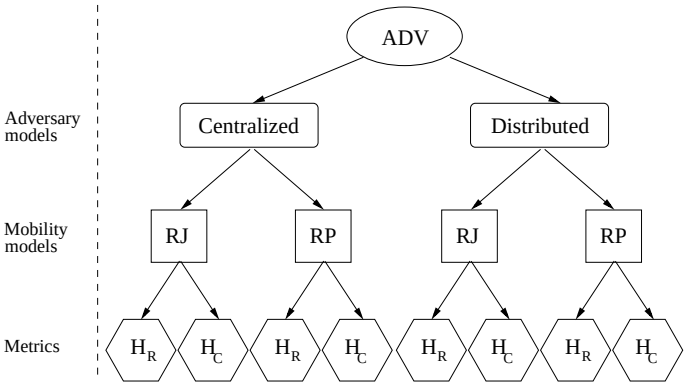


Fig. 4. Adversary models and metrics.

Hence, for each combination of adversarial model and sensor mobility model we study the metrics introduced in Section 4 and defined as:

$$\mathcal{H}_R = \frac{\mathcal{G}}{\mathcal{G}+\mathcal{Y}} \quad (1)$$

$$\mathcal{H}_C = \frac{TtC}{TtC+TtH} \quad (2)$$

Health ratio (Eq. (1)) is a network-wide status measure ($0 \leq \mathcal{H}_R < 1$) depending on the number of green sensors in the network. Equation (1) does not take into account red sensors as those are the sensors that are currently compromised and no security metric can prevent ADV from learning their secrets. Moreover, $\mathcal{H}_R$ cannot reach 1 as the number of yellow sensors will always be greater than 0 for real setting scenarios. In particular, Fig. 3.b shows that no direct transition from yellow to green is allowed. This is due to the fact that ADV eavesdrops on red sensors and learns the contributions they receive. Hence, for a sensor to be healed, it must first move out of ADV's compromise region(s) (i.e., become yellow) and later "meet" a green peer to receive a contribution that ADV cannot predict.
Hereafter, we assume the network to be at steady state. Hence, the size of each set $\mathcal{R}^r$, $\mathcal{Y}^r$, $\mathcal{G}^r$ is constant, i.e, $\mathcal{R} = \mathcal{R}^r$, $\mathcal{Y} = \mathcal{Y}^r$, $\mathcal{G} = \mathcal{G}^r$, for some $r > 0$. Since sensors are uniformly distributed on the sphere, the number of red sensors $\mathcal{R}$ is independent of the mobility model and can be computed as:

$$\mathcal{R} = N \cdot \frac{S_a}{S} \quad (3)$$

The number of yellow sensors at round r can be computed as:

$$\mathcal{Y}^r = \mathcal{Y}^{r-1} + \mathcal{R}^{r-1}P_{RY} - \mathcal{Y}^{r-1}P_{YR} - \mathcal{Y}^{r-1}P_{YG} \quad (4)$$

were P_{RY} is the transition probability from red to yellow, P_{YR} is the transition probability from yellow to red, and P_{YG} is the transition probability from yellow to green. Since the network is at steady state, Eq. (4) becomes:

$$\begin{aligned} &\mathcal{R} \cdot P_{RY} - \mathcal{Y} \cdot P_{YR} - \mathcal{Y} \cdot P_{YG} = 0 \iff \\ &\mathcal{Y} \cdot (P_{YR} + P_{YG}) = \mathcal{R} \cdot P_{RY} \iff \\ &\mathcal{Y} = \mathcal{R} \cdot \frac{P_{RY}}{P_{YR} + P_{YG}} \end{aligned} \quad (5)$$

where all the state transition probabilities appear. The number of green sensors can be estimated combining Eq. (3) and Eq. (5), yielding:

$$\begin{aligned} \mathcal{G} &= N - \mathcal{Y} - \mathcal{R} \\ &= N - \mathcal{R} \cdot \frac{P_{RY}}{P_{YR} + P_{YG}} - N \cdot \frac{S_a}{S} \end{aligned} \quad (6)$$

Finally, $\mathcal{H}_R$ can be computed combining Eq. (1), Eq. (5) and Eq. (6):

$$\mathcal{H}_R = 1 - \frac{\mathcal{R} \cdot \frac{P_{RY}}{P_{YR}+P_{YG}}}{N - \frac{S_{ADV}}{S}} \quad (7)$$

Healthy cycle (Eq. (2)) evaluates the status of a sensor over its lifetime ($0 \leq \mathcal{H}_C \leq 1$) and subsumes two other metrics:

- *Time to Compromise (TtC)*: Number of rounds it takes to a green sensor to be compromised. For instance, let $\{\ldots, R, Y, G, G, G, R, \ldots\}$ be an arbitrary sequence

of state transitions, it yields $TtC = 3$. The latter 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:

$$\begin{aligned} E\{T_c\} &= \sum_{k=0}^{\infty} k \cdot P(T_c = k) \\ &= \sum_{k=0}^{\infty} k \cdot P_{GR}(1 - P_{GR}^k) = \frac{1 - P_{GR}}{P_{GR}} \end{aligned} \quad (8)$$

In turn, P_{GR} can be computed from P_{YG} recalling that $P_{GR} = \frac{Y}{G} P_{YG}$ (see Fig. 3.b.

- *Time to Heal (TtH)*: Number of rounds it takes to a red sensor to be healed, that is, the sequence of rounds a sensor spends in the red and the yellow states. For instance, let $\{\ldots, G, R, Y, R, R, Y, Y, G, \ldots\}$ be an arbitrary sequence of state transitions, it yields $TtH = 6$. To evaluate TtH, we assume s_j is 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 [33]. In particular, the states of the chain are equivalent to those described in the above coloring scheme, where we consider the green state as an absorbing one—we are interested in counting rounds up to the time when s_j becomes green. The 3×3 matrix $\mathcal{M}$ associated with the absorbing Markov chain can be expressed as:

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

Further, $\mathcal{M}$ can be partitioned as:

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

where Q captures the transition probabilities between transient states; T 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:

$$E = (I - Q)^{-1} \quad (9)$$

Vector $D = [e_1, e_2]$ provides the average absorbing time 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), e_1 represents the TtH:

$$TtH = \frac{P_{RY} - P_{YY} + 1}{P_{RR}(P_{YY} - 1) - P_{RY}P_{YR} - P_{YY} + 1} \quad (10)$$

Finally, $\mathcal{H}_C$ can be computed combining Eq. (8) and Eq. (10).

The following two sections evaluate $\mathcal{H}_C$ and $\mathcal{H}_R$ for the two adversarial models of Fig. 4. Analysis is carried out studying state transition probabilities, i.e., P_{RY}, P_{YR} and P_{YG}, for each of the considered scenarios.

7 CENTRALIZED ADV

The centralized ADV is placed on the *north pole* of the sphere and has one compromise region of size S_{ADV}.

7.1 Random Jump Mobility Model

According to RJ, each sensor chooses a random point on the sphere and reaches it within one round. The probability to become yellow being red can be computed as:

$$P_{RY} = 1 - \frac{S_{ADV}}{S}$$

The probability to become red being yellow can be computed as:

$$P_{YR} = \frac{S_{ADV}}{S}$$

Hence, the probability to become green can be evaluated as:

$$P_{YG} = 1 - P\{\mathcal{B}(s_j, r) \cap \mathcal{G} = \emptyset\}$$

where $\mathcal{B}(s_j, r) \cap \mathcal{G}$ denotes the set of green neighbors of s_j at round r. The probability that a yellow s_j has no green peers within its communication range can be approximated as:

$$P\{\mathcal{B}(s_j, r) \cap \mathcal{G} = \emptyset\} \approx \left(1 - \frac{S_s}{S - S_{ADV}}\right)^{\mathcal{G}}$$

where S_s is the sensor communication area. Since sensors are uniformly distributed on the sphere, the ratio $\frac{S_s}{S - S_{ADV}}$ can be rewritten as function of the mean number of neighbors B, yielding:

$$P\{\mathcal{B}(s_j, r) \cap \mathcal{G} = \emptyset\} \approx \left(1 - \frac{B}{N - R}\right)^{\mathcal{G}}$$

Finally, we observe that P_{YG} can be computed solving Eq. (5), that can be rewritten as:

$$\mathcal{Y}\left(\frac{S_a}{S} + 1 - \left(1 - \frac{B}{N - R}\right)^{N - \mathcal{R} - \mathcal{Y}}\right) - \mathcal{R} \cdot P_{RY} \approx 0 \quad (11)$$

7.1.1 Health Ratio

$\mathcal{H}_R$ can be computed substituting P_{RY}, P_{YR} and P_{YG} computed above in Eq. (7). Figure 5 shows the relationship between $\mathcal{H}_R$ and the mean number of neighbors for a centralized ADV that occupies 0.20, 0.10, and 0.05 of the deployment area,respectively. Errorbars show quantiles 5, 50, and 95 observed during simulations. Solid lines show the numerical solutions of Eq. (7). To reach an arbitrary level of $\mathcal{H}_R$, the transmission range (that

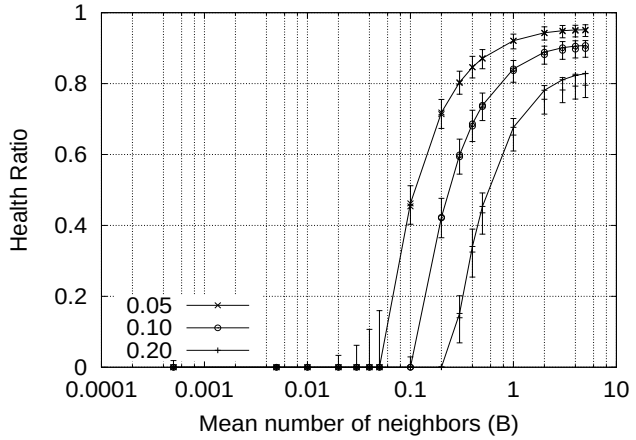


Fig. 5. $\mathcal{H}_R$ for a centralized adversary and sensor moving according to RJ: simulation results and theoretical analysis.

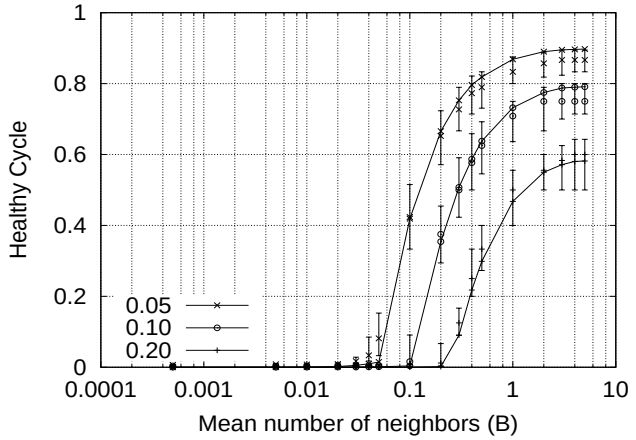


Fig. 6. $\mathcal{H}_C$ for a centralized adversary and sensor moving according to RJ: simulation results and theoretical analysis.

defines the neighborhood size) must be set proportionally to ADV's compromise power. When the number of neighbors is high ($B \geq 1$), the number of yellow sensors[4] is approximately $\mathcal{R}$, that is, at each round there are $\mathcal{R}$ yellow sensors that were red during the previous round and just came out from ADV's compromising region. For this particular setting, the effectiveness of the healing protocol is showed by the fact that all the yellow sensors are "healed" during the next round. Decreasing the mean number of neighbors, by decreasing the communication range, increases the number of yellow sensors that are not able to find a green peer to get healed, until no green sensors are left.

7.1.2 Healthy Cycle

$\mathcal{H}_C$ can be easily computed substituting Eq. (8) and Eq. (10) in Eq. (2) with P_{RY}, P_{YR}, and P_{YG} computed above.

Figure 6 shows the relationship between $\mathcal{H}_C$ and the mean number of neighbors. Errorbars show quantiles 5, 50, and 95 observed during simulations. Solid lines show the numerical solutions of Eq. (2). As sensors benefit from great mobility, they experience an appreciable number of compromises (wandering within ADV's compromise region) during their lifetime. Sensor transmission range determines its neighborhood and the number of round it will take to be healed after compromise. When the network is facing the most powerful adversary taken into account (i.e., $S_{ADV} = 0.2 \cdot S$), a sensor spends almost half of the rounds being "sick", no matter its neighborhood size.

7.2 Random Waypoint Mobility Model

According to the RP, each sensor chooses a random waypoint on the sphere and goes there in a sequence of steps of length m.

4. Recall that $\mathcal{H}_R$ does not consider red sensors.

In this model, P_{RY} can be computed if we approximate the adversary compromise area as a circle S_{ADV} of range ρ_a, as shown in Fig. 7. The error introduced by this approximation can be considered negligible as long as $\rho_a \ll \rho$. In the following, we denote with O the adversary position. Further, O' is the position of the sensor s_j at round r (i.e., $cp_j^r \equiv O'$) and D is the set of points at distance m from O', that is, the set of points the sensor could move to in the following round (i.e., $cp_j^{r+1} \in D$, with $cp_j^{r+1} - cp_j^r \leq m$). P_{RY} can be computed as:

$$\begin{aligned} P_{RY} &= P\{cp_j^{r+1} \in D_{ext} \wedge\ cp_j^r \in C\} + \\ &+ P\{cp_j^{r+1} \in D_{ext}\ \wedge\ cp_j^r \notin C\} \end{aligned}$$

but $P\{cp_j^{r+1} \in D_{ext} \wedge cp_j^r \notin C\} = 0$, that is, the sensor cannot exit from ADV because m is not sufficiently large. This yields:

$$\begin{aligned} P_{RY} &= P\{cp_j^{r+1} \in D_{ext}\ \wedge cp_j^r \in C\} \\ &= P\{cp_j^{r+1} \in D_{ext} \ \Big|\ cp_j^r \in C\} \cdot P\{cp_j^r \in C\} \end{aligned} \tag{12}$$

The probability $P\{cp_j^r \in C\}$ that s_j belongs to the circular ring C can be computed as:

$$P\{cp_j^r \in C\} = \frac{\pi\rho_a^2 - \pi(\rho_a - m)^2}{\pi\rho_a^2} \approx \frac{2m}{\rho_a}$$

The first term of Eq (12) is:

$$P\{cp_j^{r+1} \in D_{ext} \ \Big|\ cp_j^r \in C\} = \frac{E\{D_{ext}\}}{2\pi m} \tag{13}$$

where $E\{D_{ext}\}$ is the mean value of D_{ext} evaluated for $O' \in [\rho_a, \rho_a - m]$:

$$E\{D_{ext}\} = \frac{1}{m}\int_{\rho_a - m}^{\rho_a} 2 \cdot \beta'(x) \cdot m\ dx \tag{14}$$

where $\beta'(x)$ can be obtained by observing that $\rho_a \cos[\beta(x)] = x + m \cdot \cos[\beta'(x)]$, where $x = \overline{OO'}$. If $m \ll \rho_a$

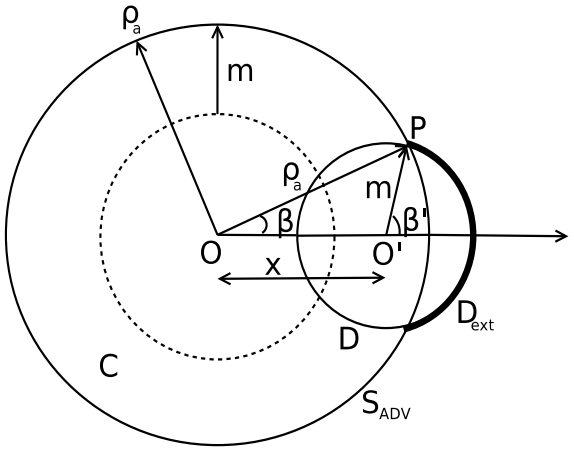


Fig. 7. Geometrical model for the evaluation of P_{RY}.

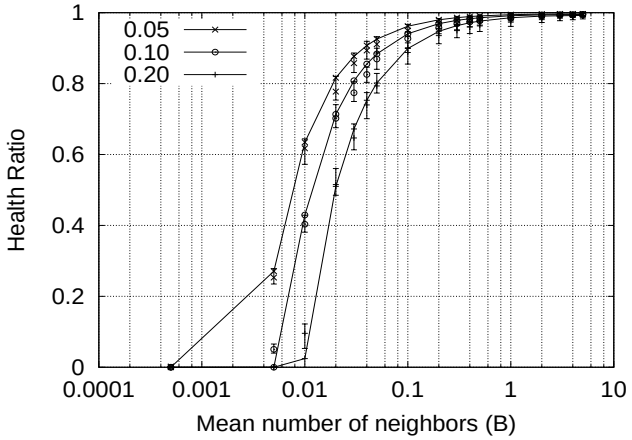


Fig. 8. $\mathcal{H}_R$ for a centralized adversary and sensor moving according to RP: simulation results and theoretical analysis.

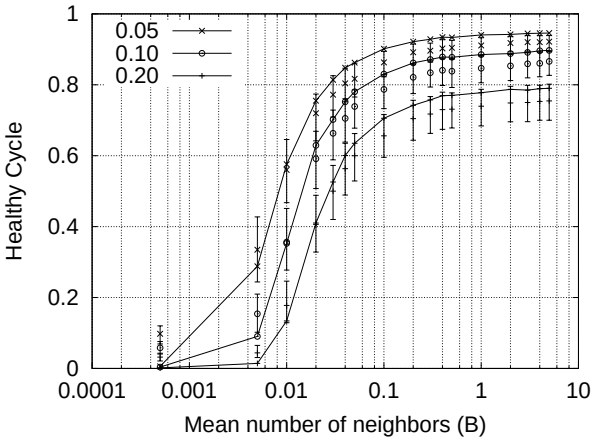


Fig. 9. $\mathcal{H}_C$ for a centralized adversary and sensor moving according to RP: simulation results and theoretical analysis.

then $\beta(x) \simeq 0$ and $\cos[\beta(x)] \simeq 1$, yielding:

$$\begin{aligned} \rho_a &= x + m \cdot \cos[\beta'(x)] \\ \beta'(x) &= \arccos\left(\frac{\rho_a - x}{m}\right) \end{aligned} \quad (15)$$

Combining Eq. (14) with Eq. (15), yields $E\{D_{ext}\} = 2m$. Finally, Eq. (13) can be re-written as:

$$P\{cp_j^{r+1} \in D_{ext} \;\Big|\; cp_j^r \in C\} = \frac{1}{\pi}$$

This yields:

$$P_{RY} = \frac{2m}{\pi \rho_a} \quad (16)$$

We split the evaluation of P_{YR} in two distinct cases: (i) when the number of neighbors is negligible ($B \ll 1$); and, (ii) when the number of neighbors is high ($B \geq 1$). In the latter case, we assume P_{YR} as negligible since all sensors are healed just after they come out from the adversarial region ($P_{YG} \simeq 1$). In the first case ($B \ll 1$), we assume the number of green sensors negligible. In fact, $P_{YR} > 0$ means that a sensor can reach its current waypoint and come back to the compromise region without having been healed, i.e., the probability for a sensor to move within a green sensor communication range is negligible. Considering Eq. 5, assuming $\mathcal{G} \simeq 0$ and consequently $P_{YG} \simeq 0$, yields:

$$P_{YR} = \mathcal{R}\frac{P_{RY}}{\mathcal{Y}}$$

Since $N = \mathcal{G}+\mathcal{Y}+\mathcal{R}$, the above equation can be rewritten as:

$$P_{YR} = \mathcal{R} \cdot \frac{P_{RY}}{N - \mathcal{R}} \quad (17)$$

Finally, we observe that P_{YG} can be computed solving Eq. (5), that in turn can be rewritten as:

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

7.2.0.1 Health Ratio: Figure 8 shows the relation between $\mathcal{H}_R$ and the mean number of neighbors. Errorbars show quantiles 5, 50, and 95 observed during simulations. Solid lines show the numerical solutions of Eq. (7). Surprisingly, limited mobility of RP lead to better performance in terms of $\mathcal{H}_R$. Given an arbitrary level of $\mathcal{H}_R$, the average neighborhood size to reach it with RP is smaller than the one required with RJ. Moreover, RP reaches values for $\mathcal{H}_R$ that are not achievable with RJ. This is due to the number of new yellow sensors at each round (Recall that $\mathcal{H}_R = \frac{\mathcal{G}}{\mathcal{G}+\mathcal{Y}}$). With RJ, sensors benefit from great mobility and any red sensor during current round will become yellow during next round with high probability. With the limited mobility range of RP, only sensors that are close the the border of ADV's compromise region (C of Figure 7) have the chance to become yellow during the next round.

7.2.0.2 Healthy Cycle: Benefits from reduced mobility are also shown in Fig. 9 where simulation and theoretical results of the $\mathcal{H}_C$ are shown. Once again,

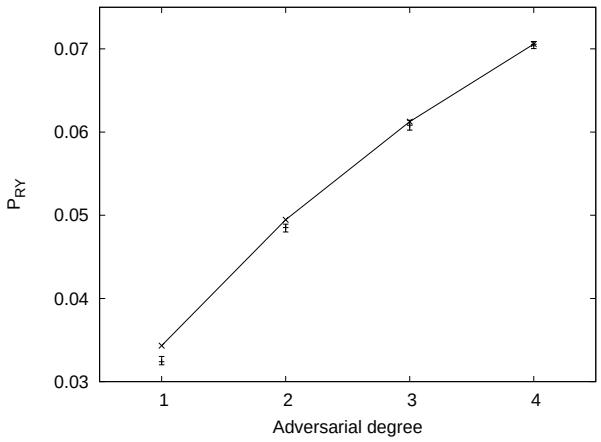


Fig. 10. P_{RY} as function of the adversarial degree: simulation results and theoretical analysis.

limited mobility allows sensors to lower their chances to be compromised. Sensors experience less compromise during their lifetime, hence leading to higher values of $\mathcal{H}_C$ while having smaller transmission ranges.

8 DISTRIBUTED ADV

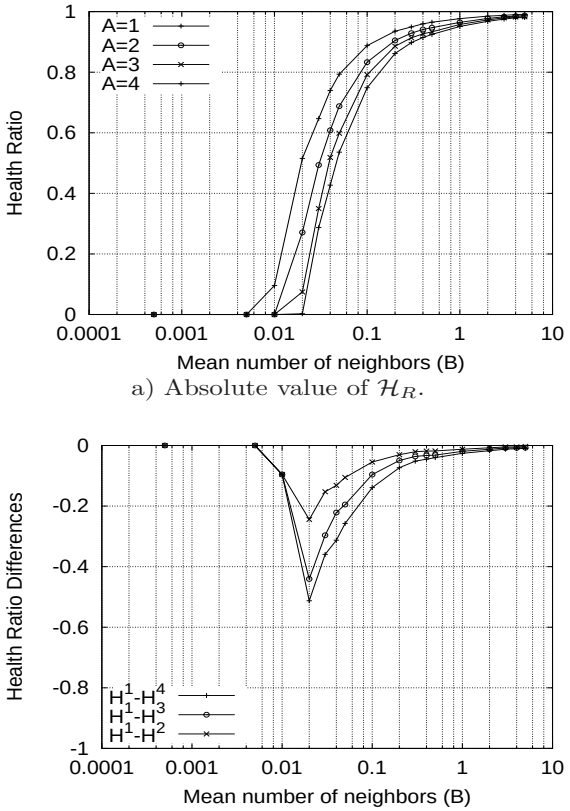
When ADV is distributed, it benefits from two or more disjoint compromise regions. In this section we fix the overall area that ADV controls to $0.2 \cdot S$ and focus on the effects of the adversarial degree on $\mathcal{H}_R$ and $\mathcal{H}_C$.

8.1 Random Jump Model

A distributed adversary is usually considered more powerful than its centralized counterpart. However, the great mobility introduced by the RJ, allows the network to experience the same performance, regardless of the number of compromise regions that ADV controls. In fact, sensors can reach any point of the sphere in one round. Hence, probabilities P_{RY}, P_{YR}, and P_{YG} can be computed as in Section 7.1. In other words, $\mathcal{H}_R$ and $\mathcal{H}_C$ are independent of the adversarial degree and ADV has no specific incentive in being distributed.

8.2 Random Waypoint Model

RP exhibits lower mobility and its performance are affected by the adversarial degree. In particular, the analysis of Section 7.2 can be also used to evaluate our protocol when the network is facing a distributed adversary. The only parameter influenced by the adversarial degree is P_{RY} that, in turn, affects the number of sensors that become yellow at any round. The solid line in Fig. 10 shows the trend of Eq. (16) as a function of the adversarial degree (A), when ADV occupies 0.2 of the deployment area. Errorbars show quantiles 5, 50, and 95 obtained via simulations. As we fix the overall area controlled by the adversary, lower values of A correspond to larger ρ_a. Hence, the ratio $\frac{m}{\rho_a}$ decreases and red sensors are less likely to exit the compromise region where they reside. As Fig. 10 shows, ADV^4 generates twice as many yellow sensors as those generated by its centralized counterpart.



a) Absolute value of $\mathcal{H}_R$.

b) $\mathcal{H}_R$ performance loss due to ADV's distribution.

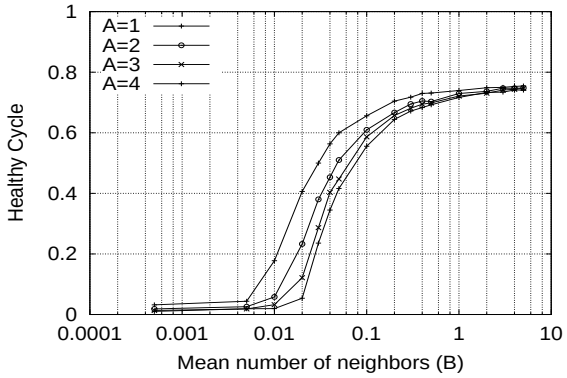
Fig. 11. $\mathcal{H}_R$ for a distributed adversary and sensor moving with RP.

8.2.1 Health Ratio

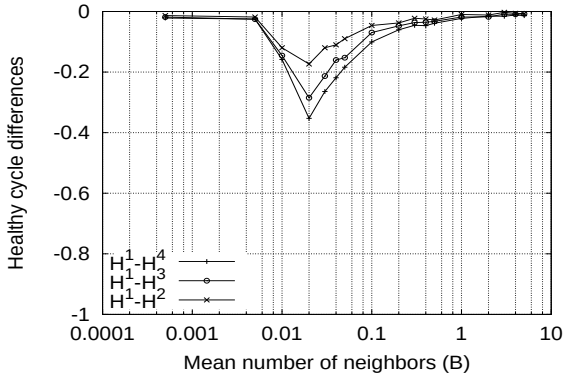
Figure 11(a) shows the relation between $\mathcal{H}_R$ and the average neighborhood size for an adversary that controls up to 4 compromise regions. Figure 11(b) shows the $\mathcal{H}_R$ loss due to an adversary of degree 2, 3, and 4, with respect to a centralized adversary. The adversarial degree is irrelevant for scenarios where sensors benefit from a large number of neighbors ($B > 1$) or when connectivity is scarce ($B < 0.01$). However, if $0.01 < B < 1$, the effect of the adversarial degree are appreciable. For example, when $B = 0.02$, the $\mathcal{H}_R$ of the network facing an adversary of degree 4, is half of the $\mathcal{H}_R$ when ADV is centralized. That is, the adversary can double the number of compromised sensors while being distributed.

8.2.2 Healthy Cycle

Also $\mathcal{H}_C$ is affected by the adversarial degree, as shown in Fig. 12(a) and Fig. 12(b). Once again, if $0.01 < B < 1$, the effect of the adversarial degree are appreciable and, when $B = 0.02$, the $\mathcal{H}_C$ of a sensor against an adversary of degree 4 is half of its $\mathcal{H}_C$ when ADV is centralized. In other words, a distributed adversary (of degree 4) doubles the number of rounds a sensor is compromised over its lifetime.



a) Absolute value of $\mathcal{H}_C$.



b) $\mathcal{H}_C$ performance loss due to ADV's distribution.

Fig. 12. $\mathcal{H}_C$ for a distributed adversary and sensor moving with RP.

9 Discussion

Table 2 summarizes, for all the mobility models considered, the key probabilities for a sensor to change its state. Performances of our protocols combined with RJ are not affected by the adversarial degree, i.e., the network exhibits the same values of $\mathcal{H}_R$ and $\mathcal{H}_C$ despite ADV is either centralized or distributed. Differently, if sensors move according to RP network performances are negatively affected by larger adversarial degrees.

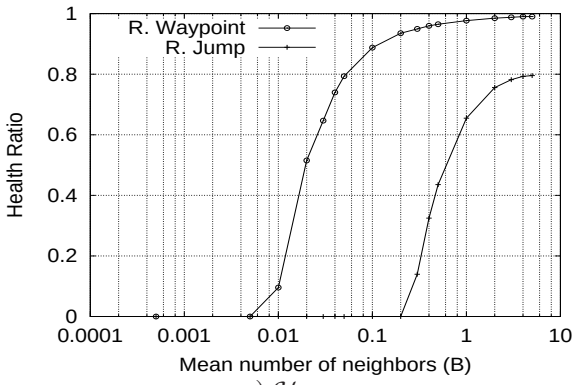
Figure 13 shows the impact of the sensor mobility model on $\mathcal{H}_R$ and $\mathcal{H}_C$ when the adversary is centralized and occupies the 20% of the deployment area. Despite its limited mobility, RP shows better performance of $\mathcal{H}_R$ and $\mathcal{H}_C$ for any average neighbor size. As an example, with RP, an average neighbor size $B \approx 0.05$ suffice to have 64% of green sensors[5]. To obtain the same performance with RP, an average neighborhood size greater than 5 is required. Both analysis and simulation results show that our collaborative protocol is effective in providing intrusion-resilience in $\mu UWSNs$. For small neighborhood sizes (i.e. $B \simeq 2$), the network exhibits a self-healing property that, with high probability, allows sensors to regain secret state as soon as they move away from the adversary-controlled regions. As an example, in a network of 500 sensors moving according to RJ, where the adversary controls 20% of the deployment area, our protocol reaches above 300 green nodes[6] with an average neighborhood size $B \approx 2$. If sensors move according to RP, the number of green almost reaches 400.

Our protocol is based on cooperation among sensors. The more sensors exchange random contribution, the better the resiliency performance. For this reason, at a first glance, large values of ρ_s might seem as an effective way for a sensor to reach more peers and improve randomness exchange. Nevertheless, as the adversary eavesdrops on its compromise area(s), sensors have an incentive in keeping a limited communication range (i.e., $\rho_s \ll \rho$). As an example, if $\rho_s \approx \rho$ then each

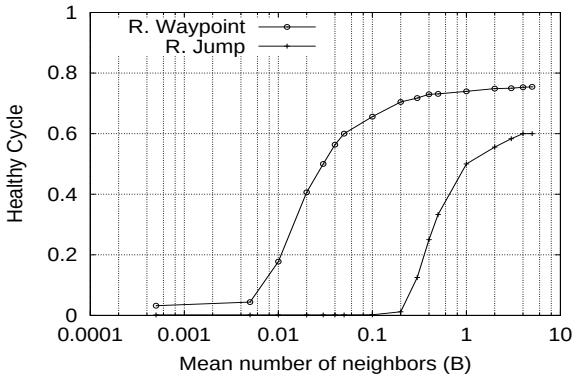
5. Recall that 20% of the sensor are inevitably red and are not considered to compute $\mathcal{H}_R$, hence 0.8 in Fig.13(a) amounts to the eighty percent of the sensors that can be possibly healed, that is 64% of the total number of sensors.

TABLE 2
Probabilities and Mobility Models

Mobility model	P_{RY}	P_{YR}	P_{YG}
RJ	$1 - \frac{S_{ADV}}{S}$	$\frac{S_a}{S}$	$\leq 1 - \left(1 - \frac{B}{N}\right)^G$
RP	$\simeq \frac{2m}{\pi \rho_a}$	$\leq \frac{R \cdot P_{RY}}{N - R}$	$\leq 1 - \left(1 - \frac{B}{N}\right)^G$



a) $\mathcal{H}_R$.



b) $\mathcal{H}_C$.

Fig. 13. Comparison of the effects of sensor mobility models on $\mathcal{H}_R$ and $\mathcal{H}_C$. $S_{ADV} = 0.2 \cdot S$ and the adversary is centralized.

6. Recall that 100 nodes are always "red".

sensor would reach all peers but, at the same time, the adversary would eavesdrop on each contribution exchange. A further assessment of this property is left open for further investigation.

To spread "healing randomness" around the network, our protocol leverages sensor mobility rather than transmission range.

Since sensor mobility is a built-in feature of $\mu UWSNs$, intrusion-resilience comes at virtually no cost. At each round, a sensor broadcasts one message and receives B messages, on average. Differently from [9], [10], no message forwarding is required as all communication is one-hop. Secret update requires only hash function computations.

The protocol is also robust with respect to message loss or sensor failure. As noted in [9], cooperative self-healing with symmetric-key cryptography is not feasible if sensors fail or message delivery is not guaranteed. The use of public key encryption (or hybrid encryption) allows the sink to decrypt any ciphertext, no matter which messages were not correctly exchanged or which sensors failed during the sink absence.

10 Conclusions

In this paper we have provided several contributions to the UWSN field. First, we have introduced a new adversary model that spreads over different areas of the deployment field. Second, we have introduced two novel metrics that, other than being interesting on their own, are of general help when assessing self-healing protocols in autonomous, distributed systems. Third, we have studied, for a wide range of system parameters, how the degree distribution of the adversary affects our self-healing protocol. In particular, the latter shows a great capability to recover from compromising for several deployment settings while incurring a negligible overhead—only local communications are required. Finally, thorough analysis and extensive simulation do support our findings.

Acknowledgements

Roberto Di Pietro has been partially supported by a Chair of Excellence granted by University Carlos III, Madrid.

G. Oligeri's research was supported by the project Autonomous security, sponsored by the Italian Ministry of Research under the PRIN 2008 Programme.

Claudio Soriente has been partially funded by the Spanish Research Agency MICINN under project CloudStorm (TIN2010-19077) and the Juan de la Cierva Fellowship ref. JCI-2010-06161, by the Madrid Research Foundation (CAM) under project S2009/TIC-1692 (cofunded by ERDF and ESF), and the European Commission under project MASSIF (FP7-257475).

References

[1] C. Hartung, J. Balasalle, and R. Han, "Node compromise in sensor networks: The need for secure systems," University of Colorado at Boulder, Technical Report TR-CU-CS-990-05, 2005.

[2] A. Francillon and C. Castelluccia, "Code injection attacks on harvard-architecture devices," in *15th ACM Conference on Computer and Communications Security (CCS'08)*, 2008, pp. 15–26.

[3] N. I. of Standards and Technology, "Fips pub 198: The keyed-hash message authentication code," http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf, 2002.

[4] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Key-insulated public key cryptosystems," in *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'02)*, 2002, pp. 65–82.

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

[6] R. Di Pietro, L. V. Mancini, C. Soriente, A. Spognardi, and G. Tsudik, "Data security in unattended wireless sensor networks," *IEEE Trans. Computers*, vol. 58, no. 11, pp. 1500–1511, 2009.

[7] ——, "Catch me (if you can): Data survival in unattended sensor networks," in *6th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'08)*, 2008, pp. 185–194.

[8] D. Ma, C. Soriente, and G. Tsudik, "New adversary and new threats: security in unattended sensor networks," *IEEE Network*, vol. 23, no. 2, pp. 43–48, 2009.

[9] R. Di Pietro, D. Ma, C. Soriente, and G. Tsudik, "POSH: Proactive co-operative self-healing in unattended wireless sensor networks," in *27th IEEE Symposium on Reliable Distributed Systems (SRDS'08)*, 2008, pp. 185–194.

[10] D. Ma and G. Tsudik, "Dish: Distributed self-healing," in *10th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS'08)*, 2008, pp. 47–62.

[11] R. Dutta, Y. D. Wu, and S. Mukhopadhyay, "Constant storage self-healing key distribution with revocation in wireless sensor network," in *IEEE International Conference on Communications (ICC'07)*, 2007, pp. 1323–1328.

[12] V. Naik, A. Arora, S. Bapat, and M. G. Gouda, "Whisper: Local secret maintenance in sensor networks," *IEEE Distributed Systems Online*, vol. 4, no. 9, 2003.

[13] K. Dantu, M. H. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. S. Sukhatme, "Robomote: enabling mobility in sensor networks," in *4th International Symposium on Information Processing in Sensor Networks (IPSN'05)*, 2005, pp. 404–409.

[14] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," in *IEEE International Conference on Robotics and Automation (ICRA'02)*, 2002, pp. 1327–1332.

[15] G. Wang, G. Cao, T. F. La Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*, 2005, pp. 2302–2312.

[16] M. H. Rahimi, H. Shah, G. S. Sukhatme, J. S. Heidemann, and D. Estrin, "Studying the feasibility of energy harvesting in a mobile sensor network," in *IEEE International Conference on Robotics and Automation (ICRA'03)*, 2003, pp. 19–24.

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

[18] M. Conti, R. Di Pietro, A. Gabrielli, L. V. Mancini, and A. Mei, "The quest for mobility models to analyse security in mobile ad hoc networks," in *7th International Conference on Wired/Wireless Internet Communications (WWIC'09)*, 2009, pp. 85–96.

[19] M. Conti, R. Di Pietro, L. V. Mancini, and A. Mei, "Mobility and cooperation to thwart node capture attacks in manets," *EURASIP J. Wireless Comm. and Networking*, vol. 2009, 2009.

[20] R. Di Pietro, L. V. Mancini, C. Soriente, A. Spognardi, and G. Tsudik, "Playing hide-and-seek with a focused mobile adversary in unattended wireless sensor networks," *Ad Hoc Networks*, vol. 7, no. 8, pp. 1463–1475, 2009.

[21] R. Di Pietro, G. Oligeri, C. Soriente, and G. Tsudik, "Intrusion-resilience in mobile unattended wsns," in *29th IEEE International Conference on Computer Communications (INFOCOM'10)*, 2010, pp. 2303–2311.

[22] ——, "Securing mobile unattended wsns against a mobile adversary," in *29th IEEE Symposium on Reliable Distributed Systems (SRDS'10)*, 2010, pp. 11–20.
[23] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, pp. 483–502, 2002.
[24] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: a survey," *IEEE Network*, vol. 18, no. 4, pp. 45–50, 2004.
[25] E. P. G.D. Murphy and W. Marnane, "Area-efficient processor for public-key cryptography in wireless sensor networks," in *2nd International Conference on Sensor Technologies and Applications (SENSORCOMM'08)*, 2008, pp. 667–672.
[26] R. Wang, W. Du, X. Liu, and P. Ning, "ShortPK: A short-term public key scheme for broadcast authentication in sensor networks," *ACM Trans. Sen. Netw.*, vol. 6, pp. 9:1–9:29, January 2010.
[27] V. Shoup, "OAEP reconsidered," in *21st Annual International Cryptology Conference (CRYPTO'01)*, 2001, pp. 239–259.
[28] D. Frenkel and B. Smit, Eds., *Understanding Molecular Simulation: From Algorithms to Applications*. Orlando, FL, USA: Academic Press, Inc., 1996.
[29] T. Park and K. G. Shin, "Soft tamper-proofing via program integrity verification in wireless sensor networks," *IEEE Trans. Mob. Comput.*, vol. 4, no. 3, pp. 297–309, 2005.
[30] A. Seshadri, A. Perrig, L. van Doorn, and P. K. Khosla, "Swatt: Software-based attestation for embedded devices," in *2004 IEEE Symposium on Security and Privacy (SP'04)*, 2004, pp. 272–282.
[31] Y. Yang, X. Wang, S. Zhu, and G. Cao, "Distributed software-based attestation for node compromise detection in sensor networks," in *26th IEEE Symposium on Reliable Distributed Systems (SRDS'07)*, 2007, pp. 219–230.
[32] G. Oligeri, "Mobile unattended sensor networks @ sphere," http://muwsns.sourceforge.net/, 2009.
[33] I. Marius, *Finite Markov Processes and their applications*, Wiley, Ed., New York, NY, USA, 2007.

Roberto Di Pietro is Assistant Professor of Computer Science at the Department of Mathematics of Università di Roma Tre - Roma, Italy; a research associate at National Research Council - Security Group - Pisa; and, the PI of the: Security and PRivacy INnovation GRoup (SPRINGER) at Roma Tre. Since 1995 to 2006 he served as an Officer for the technical branch of the Italian Ministry of Defence. His main research interests include: security and privacy for wireless systems; cloud and virtualization security; security and privacy for distributed systems; applied cryptography; computer forensics, and role mining for access control systems (RBAC).

Gabriele Oligeri is a post-doctoral researcher at the University of Trento. He received his Ph.D. in Information Engineering from the Engineering Ph.D. school "Leonardo da Vinci" of the University of Pisa. His research interests include security and privacy in distributed systems.

Claudio Soriente is a post-doctoral researcher in the Institute of Information Security at ETH Zürich. He received his Ph.D. in Networked Systems from the University of California, Irvine. His research interests include distributed systems, security and privacy.

Gene Tsudik is a "Lois and Peter Griffin" Professor of Computer Science at the University of California, Irvine (UCI). He obtained his PhD in Computer Science from USC in 1991. Before coming to UCI in 2000, he was at IBM Zurich Research Laboratory (1991-1996) and USC/ISI (1996-2000). Over the years, his research interests included many topics in security and applied cryptography. He currently serves as Director of Secure Computing and Networking Center (SCONCE) and Director of the Networked Systems (NetSys) Graduate Program at UCI. Since 2009, he is the Editor-in-Chief of ACM Transactions on Information and Systems Security (TISSEC).