This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Intrusion-Resilience in Mobile Unattended WSNs

Roberto Di Pietro [§] [‡], Gabriele Oligeri [*] [†], Claudio Soriente [¶], Gene Tsudik [†]
[*] ISTI-CNR, Pisa Research Area, Pisa, Italy
[†] Computer Science Department, University of California, Irvine, USA
[§] UNESCO Chair in Data Privacy, Universitat Rovira i Virgili, Av. Països Catalans 26, E-43007 Tarragona, Spain
[¶] Universidad Politenica de Madrid, Spain
[‡] Department of Mathematics, Università di Roma Tre, Rome, Italy

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 and (even if the sensor is later released) it is very difficult for the sensor to regain security, i.e., to obtain intrusion-resilience. Existing solutions rely on the presence of an on-line trusted third party, such as a sink, or on the availability of secure hardware on sensors. Neither assumption is realistic in large-scale Unattended WSNs (UWSNs), characterized by long periods of disconnected operation and periodic visits by the sink. In such settings, a mobile adversary can gradually corrupt the entire network during the intervals between sink visits. As shown in some recent work, intrusion-resilience in UWSNs can be attained (to a degree) via cooperative self-healing techniques.**

In this paper, we focus on intrusion-resilience in Mobile Unattended Wireless Sensor Networks ($\mu UWSNs$) where sensors move according to some mobility model. We argue that sensor mobility motivates a specific type of adversary and defending against it requires new security techniques. Concretely, we 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. We provide a thorough analysis of the proposed protocol and support it by extensive simulation results.

I. 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, resource limitations is not the main challenge in designing security techniques for WSNs. It is lack of ubiquitous (inexpensive) tamper-resistant hardware that makes sensor compromise a real threat. Some recent results (e.g., [1], [2]) showed that commodity sensors can be easily corrupted. Once a sensor is corrupted and all of its secrets are exposed, any cryptographic protocol ceases to be effective.

Based on the time of corruption, we can view the security state of a given sensor as a sequence of 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 secrets 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, the secret for round $r \geq 1$ ($\mathcal{K}^r$) is computed as $H(\mathcal{K}^{r-1})$, where $H(\cdot)$ is a one-way function. Hence, if the adversary learns secret $\mathcal{K}^r$, it cannot compute any secrets used in prior rounds. However, backward security is much more challenging, since knowledge of $\mathcal{K}^r$ allows the adversary to compute secrets for future rounds (any $\mathcal{K}^{r'}$ for $r' > r$) by mimicking the secret evolution procedure. Note that this is possible even if the adversary is no longer in control of a given sensor in round r'. Backward secrecy would be trivial to obtain 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. 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; this is assumed in key-insulated schemes [3], [4]. In such schemes, forward and backward security is achieved by having end-devices evolve their secrets in cooperation with a trusted third party, 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, in Unattended WSNs (UWSNs), the sink visits the network infrequently, which rules out key-insulated schemes. This challenge forms the premise for our work in this paper.

Contributions: We investigate collaborative intrusion-resilience in $\mu UWSNs$ where sensors migrate within a fixed deployment area. Assuming a stationary adversary

978-1-4244-5837-0/10/$26.00 ©2010 IEEE

that controls a portion of the deployment area, sensors take advantage of mobility and cooperation with peers to regain security even after having been corrupted by inadvertently wandering into the area under adversarial control.

To analyze the proposed protocol, we use a spherical deployment area, which makes it easy to ensure uniform node distribution. However, our approach is generic and applicable to most geometric shapes and surfaces.

Using both analytical and simulation results, we show that the proposed protocol provides strong security (i.e., intrusion-resilience via secret state recovery) without any trusted third parties or secure hardware.

Organization. Next section surveys related work in the area and Section III introduces the mobility and the adversarial models. Section IV presents the new protocol to achieve intrusion-resilience. The analysis of the protocol, for each of the considered mobility models, is provided in Section V, where extensive simulations support the analytical findings. Discussion follows in Section VI. Finally, Section VII reports some concluding remarks.

II. Related Work

Some prior work has considered key exposure following sensor compromise. Dutta, et al. [5] 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 [6] 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 $K^r_{jq} = F\left(H(K^{r-1}_j), H(K^{r-1}_q)\right)$, where K^{r-1}_j and K^{r-1}_q 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.

In the last two 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 [8] to include the case where the adversary's goal is to indiscriminately erase all sensor data. Another recent result [9] introduced simple cryptographic techniques to prevent the adversary from recognizing data that it aims to erase. Sensor cooperation to achieve self-healing in stationary UWSNs is explored in [10] and [11].

Recently, mobile WSNs have began to attract attention because of the advantages that mobility brings to sensing applications [12]. If sensors move, the network can guarantee optimal area coverage, even if precise sensor deployment is unfeasible (e.g., because of hostile or inaccessible conditions of the deployment area) [13]. Also, mobility helps solve network connectivity problems caused by sensor failures and allows sensors to adapt their sampling power to respond to precise events [14]. Moreover, mobile sensors can extend sensor lifetimes bringing energy to sensors with depleted batteries [15]. Finally, mobility is currently being investigated as a means to detect sensor capture attacks [16], [17] and [18].

III. System Model

We now describe the network environment and the adversarial model.

A. Network Environment

We consider a network deployed over a spherical surface. The main reason for choosing it is that it is easy to achieve uniform coverage over a spherical area with random mobility models [19]. Another reason is that random mobility models over non-spherical surfaces exhibit high variability in the average number of neighbors [19]. The latter is a key parameter for *analyzing* the effectiveness of our approach and a sphere is likely the most suitable area for our purpose. However, we stress that the shape of the deployment area's surface is not the focus of our work. Our techniques can be applied to $\mu UWSN$ deployed on any fixed-area surface.

The envisioned $\mu UWSN$ includes N sensors uniformly distributed over a spherical region of radius ρ with surface area S. Let $\mathbf{N} = \{s_1, \ldots, s_N\}$ be the set of all sensors. At initial deployment, sensor s_j position is cp^0_j.

Time is divided in rounds and all sensors' clocks are loosely synchronized, e.g., via [20]. 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^r_j. Each sensor has enough storage to accommodate $O(v)$ data items.

Sensors have a common one-way hash function $H(\cdot)$ used as a pseudo-random number generator (PRNG). Each sensor has a unique random secret seed used to initialize the PRNG; this seed is chosen by the sink and loaded onto each sensor upon each sink visit. We use $\mathcal{K}^0_j$ to denote the initial seed of s_j. We use notation $X \xleftarrow[\$]{y} Z$ to denote a sensor using its PRNG to generate y distinct elements from Z and to assign them to X.

The sink is a trusted party that visits the $\mu UWSN$ with certain frequency which has an upper bound of v – the maximum number of rounds $\mu UWSN$ remains unattended. 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.

We assume that each collected data item is encrypted separately, using the sink's public key PK, known to each sensor. Although, in the past, public key encryption was shunned by the sensor security community because of its high cost, recent developments make public key encryption feasible on commodity sensors [21], [22]. It might not be obvious *why we* *are using public key encryption* when symmetric encryption is cheaper in all respects. The reason for using public key is that it allows the sink to seamlessly decrypt anything that sensors encrypt (for it) in any round. (As discussed below, the security is based on the use of secret padding or randomizers [23] and not on the mere use of public key encryption). In contrast, if we were to use symmetric encryption, it would be quite hard (indeed, sometimes even impossible) for the sink to decrypt data. This topic is discussed in detail in [9], [10] and [24].

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 is too long, 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, security is based on the secrecy of randomized padding[2], which, for convenience, we also refer to as K_j^r. Regardless of how it is used, K_j^r 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.

Sensors are free to move over the deployment area according to a network-wide mobility model. At round r, s_j moves to a new point cp_j^r on the sphere, obtains data d_j^r and encrypts it, as described above. We consider two mobility models:

- **Random Jump Model:** 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 Model:** all sensors move with the same constant speed and can cover a 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^1, it picks a new waypoint and starts moving towards it, etc..

It is worth noticing that increasing m the Random Waypoint tends to the Random Jump, but a detailed analysis on the influence of the step size m is not the goal of this paper. However, these two models provide uniform coverage of the sphere by randomly selecting the latitude θ and the longitude ϕ of waypoints, according to the "*trig*" method [25].

Algorithms 1 and 2 show the pseudocode 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 previous position cp, towards the waypoint wp.

Algorithm 1: RANDOM-JUMP()

```
/* Generate a new random waypoint and move to it
   */
$cp.\phi \overset{1}{\underset{\$}{\leftarrow}} [-\pi, \pi]$
$rnd \overset{1}{\underset{\$}{\leftarrow}} [-1, 1]$
$cp.\theta = \frac{1}{\pi} \arccos(rnd)$
return($cp$)
```

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

```
let $cp$ be the current position of the sensor
let $wp$ be the current waypoint of the sensor
let $m$ be the step length
/* If waypoint $wp$ is reached, generate new
   waypoint                                    */
if ($cp == wp$) then
    $wp.\phi \overset{1}{\underset{\$}{\leftarrow}} [-\pi, \pi]$
    $rnd \overset{1}{\underset{\$}{\leftarrow}} [-1, 1]$
    $wp.\theta = \frac{1}{\pi} \arccos(rnd)$
/* Move towards $wp$ with step of length $m$     */
$cp = \texttt{Move}(cp, wp, m)$
return([$cp, wp$])
```

Each s_j has a circular 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 mean number of neighbors of s_j at round r. Since sensors are always uniformly distributed on the sphere, this yields $B = B^r = N \cdot \frac{S_s}{S}, \ r > 0$.

B. 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; thus, there is no incentive for the adversary to move, i.e., it might as well be stationary and wait for sensors to move to its controlled area. Albeit stationary, the envisioned adversary differs from other adversarial models considered in most prior WSN security literature. The latter is static in terms of the number of sensors it corrupts, i.e., it compromises k out of n sensor throughout the network lifetime. Our adversary, however, is stationary with respect to the portion of the deployment area it controls; but, the subset of compromised sensors changes as nodes move into and out of the adversary-controlled area.

ADV resides on a random point ap of the sphere and has a circular corruption area S_a of radius ρ_a. The set of compromised sensors at round r is $\{s_j | D^o(ap, cp_j^r) \leq \rho_a\}$. In other words, if s_j winds up at distance smaller or equal to ρ_a from ADV, we assume that it is corrupted, i.e., ADV reads all s_j's storage/memory and listens to all incoming

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

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

TABLE I
NOTATION SUMMARY.

S	spherical region/surface
ρ	radius of S
v	maximum number of rounds $\mu UWSN$ stays unattended
r, r'	round indices
$\mathbf{N} = \{s_1, \ldots, s_N\}$	set of sensors
N	size of $\mathbf{N}$
s_j, s_q	generic sensors
s_i	target sensor
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	key used 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 compromise 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

and outgoing communications. Even if a corrupted sensor moves away from the adversary-controlled area, ADV can still compute its future secrets[3].

We distinguish between a **focused** and a **non-focused** ADV. A non-focused ADV's goal is to learn as many sensor secrets as possible. In contrast, a focused ADV is interested in a specific (target) sensor s_i and aims to learn its secrets. Any sensor can be the target and its identity is only known to ADV.

We stress that ADV does not interfere with sensors' behavior, and can be described as a *read-only* adversary. This is in order to stay undetected for as long as possible. Actually, any modification to sensor code can be later discovered by the sink using techniques in [26], [27]. A stealthy adversary can benefit from repeated attacks.

Finally, we assume that ADV is aware of the network defence strategy while neither sensors nor the sink know ADV's location. In particular, there is no way to tell if a given sensor has ever been corrupted.

Table I summarizes the notation used throughout the paper.

IV. THE PROTOCOL

In our protocol, forward secrecy is (predictably) obtained with periodic secret evolution using a one-way function $H(\cdot)$. To obtain backward secrecy, the main idea is for sensors to serve as a source of randomness for their peers. A sensor that is not currently corrupted (i.e., resides outside the area controlled by ADV), but whose state is known to ADV, can regain security and move to a new secure state if it obtains at least one contribution of secure randomness from a peer sensor whose secret state is not compromised. Our protocol leverages mobility to bring computationally secure randomness to sensors whose state is compromised. Since ADV's location is secret and we cannot distinguish between compromised and non-compromised sensors, the protocol is proactively run by all.

At round r, each s_j runs Algorithm 3: it moves according to the adopted mobility model, and, after reaching its new position, broadcasts a random value based on its current PRNG state. Then, s_j obtains data from the environment and encrypts it, as described in Section III-A. Next, it receives random contributions from neighbors and uses those contributions, along with its current secret state, to compute the next round secret. Function `keyGeneration(·)` uses the sensor's current secret state to generate an encryption key.

Algorithm 3: intrusion-resilient Protocol

```
let $cp$ be the current position of the sensor
let $mov$ be the mobility model
/* Move according to a network-wide mobility
   model                                          */
switch $(mov)$ do
    case $(RANDOM\text{-}JUMP)$
        $cp = \mathtt{RANDOM\text{-}JUMP}(cp)$
    case $(RANDOM\text{-}WAYPOINT)$
        $[cp, wp] = \mathtt{RANDOM\text{-}WAYPOINT}(cp, wp, m)$
/* Pick a new secret                              */
$t \xleftarrow[\$]{1} \$$
/* Broadcast the new secret to neighbors/peers
   */
$\mathtt{broadcast}(t)$
/* Sense data                                     */
Obtain $d_j^r$
/* Generate a new key from secret state           */
$K_j^r = \mathtt{keyGeneration}(\mathcal{K}_j^r)$
/* Encrypt and store current data                 */
Store $E_{PK}(K_j^r, d_j^r, r, s_j)$
/* Initialize peer contributions vector           */
$R_j^r = [\emptyset]$
$c = 0$
/* Receive peer contributions                     */
while $(roundTimer)$ do
    Receive $t_p^r$ from $s_p$
    $R_j^r[c] = t_p^r$
    $c = c + 1$
/* Generate new secret state                      */
$\mathcal{K}_j^{r+1} = H(\mathcal{K}_j^r || R_j^r[0] || \ldots || R_j^r[c-1])$
Delete $K_j^r, \mathcal{K}_j^r$
```

V. ANALYSIS

To support analytical findings with experimental results, we developed a software simulator [28] for the spherical deployment area. In all our simulations, the $\mu UWSN$ contains $N = 500$ sensors moving over a sphere of radius $\rho = 10^5$. The

[3] Since secrets are computed using one-way functions, ADV can mimic the secret evolution step.

orthodromic distance covered with a step is $m = 2,000$ for the Random Waypoint Model. Sensor transmission range ρ_s is chosen such that $\frac{S_s}{S}$ ranges in $[10^{-6}, \ldots, 10^{-2}]$. In fact, as the number of neighbors is central to cooperative self-healing, sensor transmission range dramatically influences the performance of proposed protocol. 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 its corruption range ρ_a is chosen such that its compromise area S_a is 0.05, 0.1 and 0.2, of the spherical surface, respectively.
At any time during the protocol, the set of sensors can be partitioned into three distinct groups: red, yellow, or green, defined as follows.

- **Red** ($\mathcal{R}^r$): a sensor is red if it is currently within ADV's corruption area S_a
- **Yellow** ($\mathcal{Y}^r$): a sensor is yellow if it is not in S_a, but ADV still knows its state (i.e., keys and PRNG state).
- **Green** ($\mathcal{G}^r$): a sensor is green if its current secrets are unknown to ADV. This is because either it has never been within S_a or has been "healed" via the proposed protocol.

Figure 1 shows our scenario: (1) a green sensor remains green until it moves at distance less or equal than ρ_a from ADV; (2) a red sensor cannot become green without becoming yellow first; and (3) a yellow sensor can become green only if it receives at least one contribution from a green sensor. Figure 2 depicts the state transition diagram.

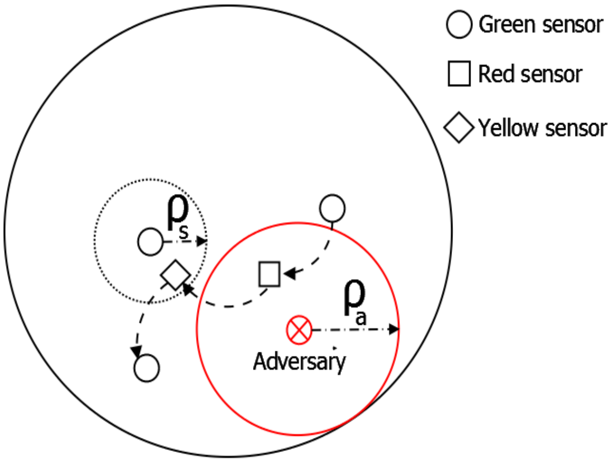

Fig. 1. Reference Scenario.

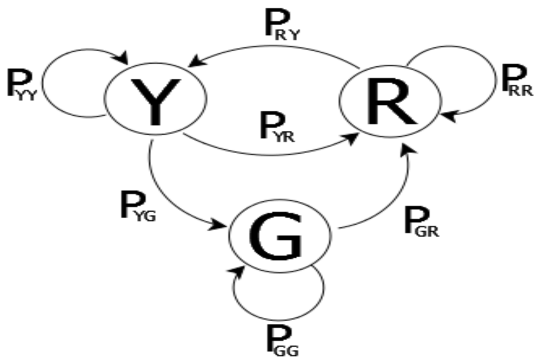

Fig. 2. State Transition Diagram.

Knowledge of sensor's secrets allows ADV to perform several attacks, ranging from sensor impersonation to compromising confidentiality of sensed data. The goal of a non-focused ADV is to maximize the set of red and yellow sensors. Whereas, our intrusion-resilient protocol is designed to maximize the number of green sensors.

A. Non-Focused ADV

To assess the effectiveness of the proposed intrusion-resilience protocol against a non-focused ADV, we analyze the number of red and yellow sensors at each round.
Let $\mathcal{R}$, $\mathcal{Y}$, and $\mathcal{G}$ be the mean number of red, yellow, and green sensors, respectively. Hereafter, we assume the network to be at steady state, that is $\mathcal{R}^r = \mathcal{R}$, $\mathcal{Y}^r = \mathcal{Y}$, $\mathcal{G}^r = \mathcal{G}$, for some $r > 0$. Since sensors are uniformly distributed on the surface, the number of red sensors $\mathcal{R}$ is independent of the mobility model and can be computed as:

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

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

$$\mathcal{Y}^r = \mathcal{Y}^{r-1} + \mathcal{R}^{r-1}P_{RY} - \mathcal{Y}^{r-1}P_{YR} - \mathcal{Y}^{r-1}P_{YG} \tag{1}$$

where P_{RY} is the transition probability from red to yellow, P_{YR} is the transition probability from yellow to red state, and P_{YG} is the transition probability from yellow to green.
Since the network is at steady state, Eq.(1) becomes:

$$\begin{aligned} &\mathcal{R} \cdot P_{RY} - \mathcal{Y} \cdot P_{YR} - \mathcal{Y} \cdot P_{YG} = 0 \iff \\ &\mathcal{Y} \cdot (P_{YR} + P_{YG}) = \mathcal{R} \cdot P_{RY} \iff \\ &\mathcal{Y} = \mathcal{R} \cdot \frac{P_{RY}}{P_{YR} + P_{YG}} \end{aligned} \tag{2}$$

1) Random Jump Model: Each sensor chooses a random point on the sphere and reaches it in one round. The probability to become yellow being red can be computed as:

$$P_{RY} = 1 - \frac{S_a}{S}$$

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

$$P_{YR} = \frac{S_a}{S}$$

The probability to become green can be evaluated as:

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

The probability that a yellow sensor s_j has no green sensors within its communication range can be approximated as:

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

The ratio $\frac{S_n}{S-S_a}$ can be rewritten as function of the mean number of neighbors B, yielding:

$$P_{YG} \approx 1 - \left(1 - \frac{B}{N}\right)^{N-\mathcal{R}-\mathcal{Y}} \tag{3}$$

Finally, Equation (2) can be rewritten as:

$$\mathcal{Y}\left(\frac{S_a}{S} + 1 - \left(1 - \frac{B}{N}\right)^{N-\mathcal{R}-\mathcal{Y}}\right) - \mathcal{R} \cdot P_{RY} \approx 0 \tag{4}$$

Figure 3 shows the simulation results and the theoretical analysis associated to the Random Jump Model. Errorbars show quantiles 5, 50 and 95 related to the sum of yellow and red (compromised) sensors, observed during the simulations. Solid lines show the numerical solutions of Eq. (4). When the number of neighbors is high ($B \geq 1$), the number of compromised sensors can be approximated as $2 \cdot \mathcal{R}$, that is, at each round there are $\mathcal{R}$ red sensors and $\mathcal{R}$ yellow sensors that were red at previous round and just came out from the adversarial region. Decreasing the mean number of neighbors, by decreasing the communication range, increases the number of compromised sensors, until no green sensors are left.

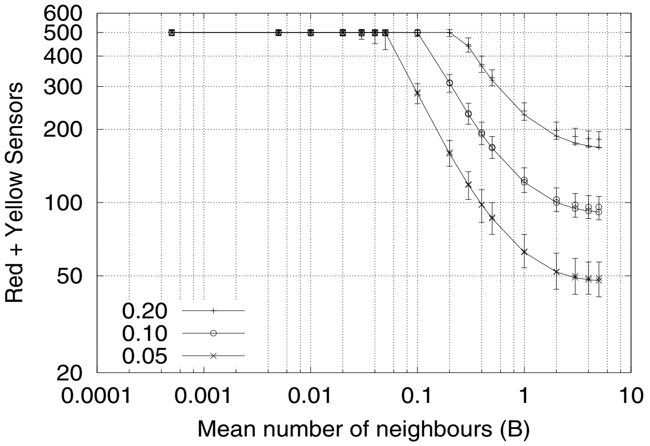

Fig. 3. Random Jump Model: simulation results and theoretical analysis.

2) Random Waypoint Model: In this model, P_{RY} can be computed if we approximate the corruption area of the adversary with a circle S_a of range ρ_a, as shown in Fig. 4. The error introduced by this approximation can be considered negligible as long as $\rho_a \ll \rho$. In the following, we denote with O and S_a the adversary position and the adversary area of radius ρ_a, respectively. Moreover, 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 $cp_j^{r+1} - cp_j^r = m$). P_{RY} can be computed as:

$$P_{RY} = P\{cp_j^{r+1} \in D_{ext} \wedge cp_j^r \in A_m\} + P\{cp_j^{r+1} \in D_{ext} \ \wedge cp_j^r \notin A_m\}$$

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

$$\begin{aligned} P_{RY} &= P\{cp_j^{r+1} \in D_{ext} \wedge cp_j^r \in A_m\} \\ &= P\{cp_j^{r+1} \in D_{ext} \;\Big|\; cp_j^r \in A_m\} \cdot P\{cp_j^r \in A_m\} \end{aligned} \tag{5}$$

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

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

The first term of Eq. (5) is:

$$P\{cp_j^{r+1} \in D_{ext} \;\Big|\; cp_j^r \in A_m\} = \frac{E\{D_{ext}\}}{2\pi m} \tag{6}$$

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

$$E\{D_{ext}\} = \frac{1}{m}\int_{\rho_a - m}^{\rho_a} 2 \cdot \beta'(x) \cdot m \; dx \tag{7}$$

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

$$\begin{aligned} \rho_a &= x + m \cdot \cos[\beta'(x)] \\ \beta'(x) &= \arccos\left(\frac{\rho_a - x}{m}\right) \end{aligned} \tag{8}$$

Combining Eq. (7) with Eq. (8), yields $E\{D_{ext}\} = 2m$. Finally, Eq. (6) can be re-written as:

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

This yields:

$$P_{RY} = \frac{2m}{\pi\rho_a} \tag{9}$$

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 ($P_{YG} \simeq 1$). Therefore, we do not take into account those sensors that choose their waypoint one step out from the adversarial region and then come back into the adversary controlled area. 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 adversary region without having been healed, i.e., the probability for a

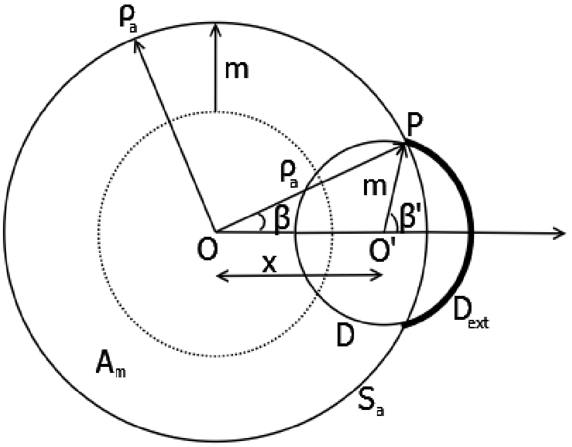

Fig. 4. Geometrical model for the evaluation of the probability P_{RY}.

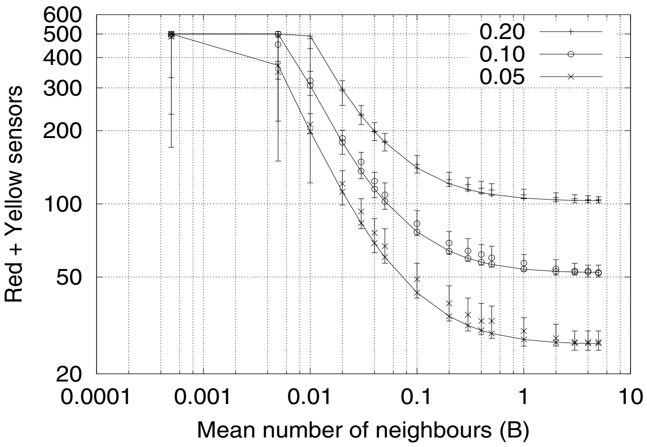

Fig. 5. Random Waypoint Model: simulation results and theoretical analysis.

sensor to move within a green sensor communication range is negligible. Considering Eq. (2), 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:

$$P_{YR} = \mathcal{R} \cdot \frac{P_{RY}}{N - \mathcal{R}} \tag{10}$$

The probability P_{YG} to become green from being yellow can be computed as in the Eq. (3). Also in this case, the latter is valid only if $m \geq 2 \cdot \rho_s$, otherwise it can be considered as an upper bound for the probability to be healed (P_{YG}). Combining Eq. (9), Eq.(10), Eq.(3) and Eq.(2), yields:

$$\mathcal{Y}\left(\frac{\mathcal{R} \cdot P_{RY}}{N - \mathcal{R}} + 1 - \left(1 - \frac{B}{N}\right)^{N-\mathcal{R}-\mathcal{Y}}\right) - \mathcal{R} \cdot P_{RY} = 0 \tag{11}$$

The three solid lines in Fig. 5 show the numerical solutions of Eq. (11) as a function of B (mean numbers of neighbors) and $\frac{S_a}{S}$ (the ratio between adversary controlled area and the whole sphere). Errorbars show quantile 5, 50, and 95 related to the sum of red and yellow sensors, experienced during the simulations. When the number of neighbors is high ($B \geq 1$), the healing protocol cuts down the number of compromised sensors to the red sensors only, that is, 25 (5% of 500), 50 (10% of 500) and 100 (20% of 500). Decreasing the number of neighbors, by reducing the communication range, increases the number of compromised sensors. In fact, the healing protocol performs worse due to the fact that less random contributions provided by green sensors are exchanged. When the number of neighbors is low ($B \leq 10^{-3}$), the healing protocol can be considered switched off (no contributions are exchanged), therefore, the adversary eventually controls all the sensors.

3) Mobility models comparison: Table II summarizes the probabilities for sensors to change their state. When the number of neighbors is high ($B \geq 1$), then $P_{YR} < P_{YG} \simeq 1$ in both mobility models, therefore almost all yellow sensors are healed in one round. However, note that the number of compromised sensors is highly dependant on the number of sensors that leave the adversarial region in one round. In the Random Jump Model, just a small fraction of the sensors that are red at round r do not leave the adversarial area (that is, do not become yellow) during next round ($P_{RY} \geq 0.8$). In the Random Waypoint Model, the step is smaller and therefore only few sensors exit the adversarial region ($P_{RY} \leq 0.05$). As a result, when $B \geq 1$, the number of compromised sensors in the Random Jump Model is twice the number of compromised sensors in the Random Waypoint Model. When $B < 1$, the Random Waypoint Model still guarantees better performances in terms of number of green sensors. In fact, while the adversary controls almost all the network in the Random Jump Model when $B \simeq 0.1$, using the Random Waypoint Model, the number of compromised sensors is just half the network.

TABLE II
PROBABILITIES AND MOBILITY MODELS

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

B. Focused ADV

Assume that ADV picks one of the sensors in the adversary controlled area, namely s_i, and decides to monitor its secret state even when s_i moves away from S_a. ADV will be able to do so as long as s_i remains yellow. Hence, we are interested in the time required to sensor s_i, once it has been corrupted, to be healed by a green peer. In the following we analyze this event, referred to as *time to heal*.

We assume s_i eventually moves to the adversary controlled area and from that round on, we investigate the number of rounds s_i remains compromised, i.e., red or yellow. To provide a theoretical analysis of the time to heal, we will use absorbing Markov chains [29]. In particular, the states of the chain are equivalent to the ones detailed in the above coloring scheme, i.e., green, yellow, and red. We consider the green state of the model depicted in Fig. 2 as the absorbing state of the Markov chain, that is $P_{GG} = 1$ and $P_{GX} = 0$ for $X \neq G$ and $X \in \{Y, R\}$. The 3×3 matrix $\mathcal{M}$ associated to the absorbing Markov chain can be generated as:

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

where P_{RY}, P_{YR}, and P_{YG} can be computed as presented in the previous sections, while P_{RR} and P_{YY} can be computed considering the fact that the sum of all the transition probabilities related to each node in Fig. 2 must be equal to 1. Hence: $P_{RR} = 1 - P_{RY}$, $P_{YY} = 1 - P_{YR} - P_{YG}$.
The matrix $\mathcal{M}$ can be partitioned as:

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

where Q gives the probabilities of transitions between transient states; A gives the probability of transition from transient state to absorbing state; and, finally, I and 0 are the identity and the null matrices, respectively. The expected number of times that the process is in a compromised state can be computed as:

$$D = (I - Q)^{-1} \qquad (12)$$

Vector $D = [d_1, d_2]$ provides average absorbing times when the chain starts from the red and the yellow state. Since we are interested in the time between corruption and healing, (i.e., the chain always starts from the red state), d_1 represents the time to heal. Figure 6 shows the time to heal experimented during the simulations and its theoretical evaluation considering Eq. (12). For both mobility models, the theoretical analysis fits simulation results. It is worth noticing that the Random Jump Model experiments shorter time to heal in respect to the Random Waypoint Model. In particular, considering a high number of neighbors ($B \geq 1$), we observe that, according to the Random Jump Model, a sensor needs at least 2 rounds to move back into the green state: the first one is needed to leave the adversarial region and to become yellow, while the second one is required to be healed by a green neighbor. On the contrary, using the Random Waypoint Model, the minimum time to heal is almost 20 rounds. In fact, the $\frac{\rho_a}{m}$ ratio is almost 13. Therefore, the sensor must spend at least 13 rounds inside the adversarial region before coming out of it. Moreover, note that the sensor also needs few rounds out of the adversary region before meeting a green peer. Indeed, the proximity of the adversarial area is populated by yellow sensors mainly.

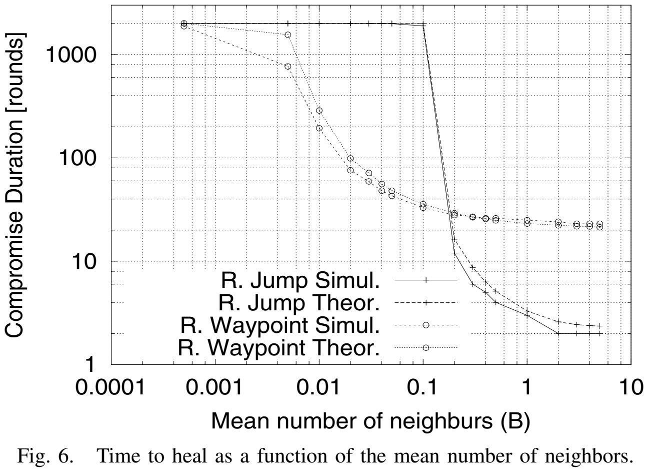

Fig. 6. Time to heal as a function of the mean number of neighbors.

VI. Discussion

The above analysis shows that the proposed protocol is effective to guarantee intrusion-resilience in $\mu UWSN$s.
In particular, the two analyzed mobility models provides complementary results. The Random Waypoint Model performs better against a non-focused adversary, i.e., there are less compromised sensors at any given round. The Random Jump Model performs better against a focused adversary as compromised sensors experience shorter time to heal.
An interesting feature of our protocol is that if the average number of neighbors (tuned either via sensor density or sensor transmission range) is high enough ($B \geq 1$), the network exhibits a self-healing property that allows sensors to regain secret state as soon as they move away from the adversary-controlled area. In this case, the surface controlled by ADV plays a very small role – if we exclude red sensors, that are proportional to S_a. This is because sensors cooperate; hence, a key parameter in the performance of the proposed protocol is the mean number of neighbors. In our simulations, we changed B tuning sensor transmission ranges. In real deployment, similar results can be obtained using shorter transmission ranges. (That would save sensor energy, hence allowing the $\mu UWSN$ to operate longer.)

Intrusion-resilience is provided with minimal computational overhead. At each round, a sensor broadcasts one message and receives B messages, on average. It also runs two one-way function operations and performs one public-key encryption operation. In contrast with [10] and [11], the proposed protocol involves lower communication overhead, since it does not rely on routing, i.e., a sensor only communicate with its neighbors. As stated earlier, our protocol leverages sensor mobility to spread "healing randomness" around the network. Since sensor mobility is a built-in feature of $\mu UWSNs$, intrusion-resilience comes at virtually no cost.
The protocol is also robust with respect to message loss or sensor failure. As noted in [10], cooperative self-healing with symmetric-key cryptography is not feasible if sensors fail or message delivery is not guaranteed. Indeed, decryption of sensors data would require the sink to synchronize with sensors decryption keys. If sensors compute keys exchanging secrets, the sink must be aware of each exchange of secret that has occurred during its absence, which is not feasible in

case of message loss or sensor failures. The use of public key encryption (or hybrid encryption) allows the sink to decrypt any ciphertext, no matter which messages were not correctly received or which sensors failed during the sink absence.

VII. Conclusions

We proposed a distributed self-healing protocol that, based on sensor mobility and one-hop communication, offers intrusion-resilience in $\mu UWSNs$. The protocol has been tested under two mobility models, assuming a spherical deployment surface. Analytical results indicate that, under the realistic adversarial model, the proposed protocol is both highly effective and efficient for a wide range of network parameters. For example, when the number of neighbors is at least (roughly) one, the protocol minimizes the number of nodes not under the control of the adversary. Our proposed approach is evaluated via thorough analysis and extensive simulations.

Acknowledgements

R. Di Pietro's research was supported in part by: The Spanish Ministry of Science and Education through projects TSI2007-65406-C03-01 'E-AEGIS' and CONSOLIDER CSD2007-00004 'ARES', and by the Government of Catalonia under grant 2005 SGR 00446.
G. Oligeri's research was supported in part by the EU 6th framework program, contract no. 38419 (Intermedia NoE), and by the University of California, Irvine.
G. Tsudik's research was supported in part by the US Army Research Office (ARO) under contract W911NF-04-1-0280.
This work was partially done while C. Soriente was at University of California, Irvine.

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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] R. Di Pietro, L. V. Mancini, C. Soriente, A. Spognardi, and G. Tsudik, "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] R. D. 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.

[9] ——, "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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] G. Wang, G. Cao, T. F. L. 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.

[15] M. H. R. andHardik 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.

[16] 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.

[17] 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.

[18] M. Conti, R. Di Pietro, A. Mei, and L. V. Mancini, "Mobility and cooperation to thwart node capture attacks in manets," *Journal on Wireless Communications and Networking (EURASIP)*.

[19] 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.

[20] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: a survey," *IEEE Network*, vol. 18, no. 4, pp. 45–50, 2004.

[21] 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.

[22] X. L. Ronghua Wang, Wenliang Du and P. Ning, "Shortpk: A short-term public key scheme for broadcast authentication in sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, to appear.

[23] V. Shoup, "OAEP reconsidered," in *21st Annual International Cryptology Conference (CRYPTO'01)*, 2001, pp. 239–259.

[24] 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.

[25] D. Frenkel and B. Smit, Eds., *Understanding Molecular Simulation: From Algorithms to Applications*. Orlando, FL, USA: Academic Press, Inc., 1996.

[26] 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.

[27] 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.

[28] G. Oligeri, "Mobile unattended sensor networks @ sphere," http://muwsns.sourceforge.net/, July 2009.

[29] I. Marius, *Finite Markov Processes and their applications*, Wiley, Ed., New York, NY, USA, 2007.