Distributed Detection of Clone Attacks in Wireless Sensor Networks

Mauro Conti, *Member*, *IEEE*, Roberto Di Pietro,
Luigi Vincenzo Mancini, and Alessandro Mei, *Member*, *IEEE*

Abstract—Wireless Sensor Networks (WSNs) are often deployed in hostile environments where an adversary can physically capture some of the nodes, first can reprogram, and then, can replicate them in a large number of clones, easily taking control over the network. A few distributed solutions to address this fundamental problem have been recently proposed. However, these solutions are not satisfactory. First, they are energy and memory demanding: A serious drawback for any protocol to be used in the WSN-resource-constrained environment. Further, they are vulnerable to the specific adversary models introduced in this paper. The contributions of this work are threefold. First, we analyze the desirable properties of a distributed mechanism for the detection of node replication attacks. Second, we show that the known solutions for this problem do not completely meet our requirements. Third, we propose a new self-healing, Randomized, Efficient, and Distributed (RED) protocol for the detection of node replication attacks, and we show that it satisfies the introduced requirements. Finally, extensive simulations show that our protocol is highly efficient in communication, memory, and computation; is much more effective than competing solutions in the literature; and is resistant to the new kind of attacks introduced in this paper, while other solutions are not.

Index Terms—Wireless sensor networks security, node replication attack detection, distributed protocol, resilience, efficiency.

1 Introduction

A Wireless Sensor Network (WSN) is a collection of sensors with limited resources that collaborate to achieve a common goal. WSNs can be deployed in harsh environments to fulfil both military and civil applications [1]. Due to their operating nature, they are often unattended, hence prone to different kinds of novel attacks. For instance, an adversary could eavesdrop all network communications; further, an adversary could capture nodes acquiring all the information stored therein—sensors are commonly assumed to be not tamper-proof. Therefore, an adversary may replicate captured sensors and deploy them in the network to launch a variety of malicious activities. This attack is referred to as the *clone attack* [53], [11], [34]. Since a clone has legitimate information (code and cryptographic material), it may participate in the network operations in the same way as a noncompromised node; hence, cloned nodes can launch a variety of attacks. A few have been described in the literature [3], [7]. For instance, a clone could create a black hole, initiate a wormhole attack [37] with a collaborating adversary, or inject false data or aggregate data in such a way to bias the final result [50]. Further, clones can leak data.

The threat of a clone attack can be characterized by two main points:

- A clone is considered totally honest by its neighbors. In fact, without global countermeasures, honest nodes cannot be aware of the fact that they have a clone among their neighbors.
- To have a large amount of compromised nodes, the adversary does not need to compromise a high number of nodes. Indeed, once a single node has been captured and compromised, the main cost of the attack has been sustained. Making further clones of the same node can be considered cheap.

To the best of our knowledge, with the exception of the protocol proposed in [45] and reviewed in the following, only centralized or local protocols have been proposed so far to cope with the clone attack. While centralized protocols have a single point of failure and high communication cost, local protocols do not detect replicated nodes that are distributed in different areas of the network. In this work, we look for a network self-healing mechanism, where nodes autonomously identify the presence of clones and exclude them from any further network activity. In particular, this mechanism is designed to iterate as a "routine" event: It is designed for continuous iteration without significantly affecting the network performances, while achieving high clone detection rate.

In this paper, we analyze the desirable properties of distributed mechanisms for detection of node replication attack [17]. We also analyze the first protocol for distributed detection, proposed in [45], and show that this protocol is not completely satisfactory with respect to the above properties. Lastly, inspired by [45], we propose a new randomized, efficient, and distributed (RED) protocol for the detection of node replication attacks, and we prove that our protocol does meet all the above cited requirements. We further

- *M. Conti, L.V. Mancini, and A. Mei are with the Dipartimento di Informatica, University of Rome "La Sapienza," Via Salaria 113, 00198 Roma, Italy. E-mail: {conti, lv.mancini, mei}@di.uniroma1.it.*
- *R. Di Pietro is with the Dipartimento di Matematica, University of Rome "Tre," room DM 300, L.go S. Leonardo Murialdo 1, 00146 Roma, Italy. E-mail: dipietro@mat.uniroma3.it.*

Manuscript received 27 May 2009; revised 22 Dec. 2009; accepted 01 Apr. 2010; published online 30 July 2010.
Recommended for acceptance by V. Gligor.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-2009-05-0074.
Digital Object Identifier no. 10.1109/TDSC.2010.25.

provide analytical results when RED and its competitor face an adversary that selectively drops messages that could lead to clone detection. Finally, extensive simulations of RED show that it is highly efficient as for communications, memory, and computations required and shows improved attack detection probability (even when the adversary is allowed to selectively drop messages) when compared to other distributed protocols.

The remainder of this paper is organized as follows: Next section reviews related work; Section 3 shows the threat model assumed in this paper; Section 4 introduces the requirements a distributed protocol for the detection of the clone attack in wireless sensor networks should meet; Section 5 describes our randomized, efficient, and distributed solution; and Section 6 shows some experimental results on RED and compares them with the results obtained in [45] in terms of detection probability, memory overhead, and energy overhead. These results confirm that RED matches the requirements set in Section 4, is more energy, memory, and computationally efficient, and detects node replication attacks with higher probability. In Section 7, we analyze how malicious nodes can affect the detection protocol performances. Finally, Section 8 presents some concluding remarks.

2 Related Work

One of the first solutions for the detection of clone attacks relies on a centralized Base Station (BS) [33]. In this solution, each node sends a list of its neighbors and their locations (that is, the geographical coordinates of each node) to a BS. The same node ID in two lists with inconsistent locations will result in a clone detection. Then, the BS revokes the clones. This solution has several drawbacks, such as the presence of a single point of failure (the BS) and high communication cost due to the large number of messages. Further, nodes close to the BS will be required to route much more messages than other nodes, hence shortening their operational life.

Another centralized clone detection protocol has been recently proposed in [6]. This solution assumes that a random key predistribution security scheme is implemented in the sensor network. That is, each node is assigned a set of k symmetric keys, randomly selected from a larger pool of keys [33]. For the detection, each node constructs a counting Bloom filter from the keys it uses for communication. Then, each node sends its own filter to the BS. From all the reports, the BS counts the number of times each key is used in the network. The keys used too often (above a threshold) are considered cloned and a corresponding revocation procedure is raised.

Other solutions rely on local detection. For example, in [9], [29], [33], [43], a voting mechanism is used within a neighborhood to agree on the legitimacy of a given node. However, this kind of a method applied to the problem of replica detection fails to detect clones that are not within the same neighborhood. As described in [45], a naïve distributed solution for the detection of the node replication attack is Node-To-Network Broadcasting. In this solution, each node floods the network with a message containing its location information and compares the received location information with that of its neighbors. If a neighbor s_w of node s_a receives a location claim that the same node s_a is in a position not coherent with the originally detected position of s_a, this will result in a clone detection. However, this method is very energy-consuming since it requires n flooding per iteration, where n is the number of nodes in the WSN.

In the *sybil attack* [29], [43], a node claims multiple existing identities stolen from corrupted nodes. Note that both the sybil and the clone attacks are based on identity theft,; however the two attacks are independent. The sybil attack can be efficiently addressed with mechanism based on RSSI [21] or with authentication based on the knowledge of a fixed key set [9], [14], [15], [25], [27].

Recent research threads cope with the more general problem of node compromise [19], [51], [47]. However, detecting node "misbehavior" via an approach that is rooted on techniques taken from intrusion detection systems [24] seems to require a higher overhead compared to clone detection. Indeed, in current solutions, detecting a misbehaving node implies observing, storing, and processing a large amount of information. However, when mobility can be leveraged, security can improve incurring just limited overhead [13]. In particular, some preliminary solutions start appearing in the literature that allow to recover sensor secrecy after node compromising [16], [23], [28], but these solutions do not cope with replica attacks.

To the best of our knowledge, the first not naïve, globally aware, and distributed node replication detection solution appeared in [45]. In particular, two distributed detection protocols leveraging emergent properties [36] have been proposed. The first one, Randomized Multicast (RM), distributes node location information to randomly selected nodes. The second one, Line-Selected Multicast (LSM), uses the routing topology of the network to detect replicas. In RM, when a node announces (locally broadcasts) its location, each of its neighbors sends (with probability p) a digitally signed copy of the location claim to a set of randomly selected nodes. Assuming that there is a replicated node, if every neighbor randomly selects $O(\sqrt{n})$ destinations, with a not negligible probability, at least one node will receive a pair of not coherent location claims. We will call *witness* the node that detects the existence of a node in two different locations within the same protocol run. The RM protocol implies a high communication cost: Each neighbor has to send $O(\sqrt{n})$ messages. To solve this problem, the authors propose the LSM protocol.

The LSM protocol is similar to RM, but it introduces a remarkable improvement in terms of detection probability. In LSM, when a node announces its location, every neighbor first locally checks the signature of the claim, and then, with probability p, forwards it to $g \geq 1$ randomly selected destination nodes. As an example in Fig. 1, node a announces its location and one of its neighbors, node b, forwards the claim to node f. A location claim, when traveling from source to destination, has to pass through several intermediate nodes that form the so-called *claim message path*. Moreover, every node that routes this claim message has to check the signature, to store the message, and to check the coherence with the other location claims received within the same run of the detection protocol. Node replication is detected by the node (if present) on the intersection of two paths generated by two different node claims carrying the

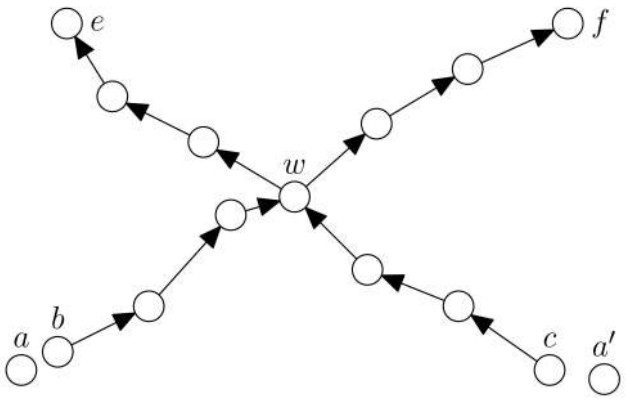

Fig. 1. Example of LSM protocol iteration.

same ID and coming from two different nodes. In the example shown in Fig. 1, node a' is a clone of node a (it has the same ID of node a). The claim of a' is forwarded by node c to node e. In the example, node w will then result in the intersection of two paths carrying the claim of ID a coming from different locations. Node w, the witness, detects the attack and triggers a revocation procedure.

In [52], the authors propose two different protocols with the aim of increasing the detection probability provided by LSM. The basic idea is to logically divide the network into cells and to consider all the nodes within a cell as possible witnesses. In the first proposed protocol, *Single Deterministic Cell*, each node ID is associated with a single cell within the network. When the protocol runs, the neighbors of a node a probabilistically send a's claim to the single predetermined witness cell for a. Once the first node within that cell receives the claim message, the message is flooded to all the other nodes within the cell. In the second proposal, *Parallel Multiple Probabilistic Cells*, the neighbors of a node a probabilistically send a's claim to a subset of the predefined witness cells for a. The proposed solutions show a higher detection probability compared to LSM. However, the same predictable mechanism used to increase the detection probability can be exploited by the adversary for an attack—compromising the witnesses in order to go undetected. In fact, this predictability restricts the number of nodes (and their geographic areas) that can act as witnesses.

Another interesting distributed protocol for replicated node detection that has recently been proposed is the SET protocol [11]. SET leverages the knowledge of a random value broadcast by a BS to perform a detection phase. In particular, the shared random value is first used to generate independent clusters and corresponding clusters' heads. The specific clustering protocol used assures that the clusters are, in fact, Exclusive Subset Maximal Independent Set (ESMIS)—cluster heads are called Subset Leaders (SLDRs). Further, within the same protocol iteration used to generate clusters and SLDRs, one or more trees are defined over the network graph. The nodes of the tree correspond to the SLDRs. Then, a bottom-up aggregation protocol is run to aggregate the list of nodes belonging to the ESMIS. If a node ID is present in two different independent subsets, then the node corresponding to that node ID has been cloned. The mechanism used by the protocol prevents a node to escape detection by claiming to be managed from a nonexisting SLDR—hence escaping the tree aggregation protocol. Note that defining such aggregating trees for each protocol iteration comes with a nonnegligible cost in terms of messages. However, the main problem of this protocol is that the detection protocol itself is flawed—it and can be maliciously exploited by the adversary to revoke honest nodes (that is, nodes that are not cloned). Indeed, a malicious node acting as an SLDR could declare in its ESMIS the presence of an honest node, say a, that eventually exists in some other part of the network (that is belonging to a different ESMIS). This malicious behavior will lead the network to the "detection" and possibly to the revocation of honest node a. Due to the possibility of this attack, in the following, we do not consider SET as a benchmark for our protocol.

In [17], the authors point out the desirable properties a clone detection protocol should meet. As shown in [17], the LSM protocol [45] does not meet these properties. In particular, in LSM, some nodes have a higher probability to act as witnesses, so weakening the detection itself. The attacker can take control of the node that will act as witness with highest probability. Furthermore, the protocol's overhead is not evenly distributed among the network nodes. In [18], a randomized, efficient, and distributed clone detection protocol has been proposed. The simulation results reported in [18] show that the proposed RED protocol meets the desirable properties presented in [17].

In this paper, we review the contribution of [18] and further thoroughly investigate the feasibility of the RED protocol. The analysis and the further set of simulations presented show that the RED protocol can be actually implemented in sensor network. Also, it can be continuously iterated over the same network, as a self-healing mechanism, without significantly affecting the network performance (nodes energy and memory) and the detection protocol itself. Furthermore, we investigate the influence of an attacker intervening on message routing both for RED and LSM.

3 The Threat Model

We define a simple yet powerful adversary: It can compromise a certain fixed amount of nodes and replicate one or more into multiple copies (the *clones*). In general, to cope with this threat, it could be possible to assume that nodes are tamper-proof. However, tamper-proof hardware is expensive and energy demanding [2], [1]. Therefore, consistently with a large part of the literature, we will assume that the nodes do not have tamper-proof components. The adversary goal is to prevent clones from being detected by the detection protocol used in the network. Hence, we assume that the adversary, to reach its goal, also tries to subvert the nodes that will possibly act as witnesses.

To formalize the adversary model, we introduce the following definition:

Definition 3.1. *Assume that the goal of the adversary is to subvert the distributed detection protocol by compromising a possibly small subset T of the nodes. The adversary has already compromised a set of nodes $\mathcal{W}$, while $\mathcal{N}$ is the initial set of*

TABLE 1
LSM Overheads ($n = 1{,}000, r = 0.1$, and $g = 1$)

	Memory Occupancy	Sent Messages	Received Messages	Signature Check
Asymptotic	$O(g \cdot p \cdot d \cdot \sqrt{n})$	$O(g \cdot p \cdot d \cdot \sqrt{n})$	$O(g \cdot p \cdot d \cdot \sqrt{n})$	$O(g \cdot p \cdot d \cdot \sqrt{n})$
Average ($p = 0.1$)	20.33	22.08	49.84	21.08
Max ($p = 0.1$)	197	216	252	223
Average ($p = 0.05$)	9.98	10.98	38.60	10.17
Max ($p = 0.05$)	59	56	92	60

nodes in the network. For every node s, the node appeal $P_{witness}(s)$ returns the probability that $s \in \mathcal{N} \backslash \mathcal{W}$ is a witness for the next run of the protocol.

We characterize the adversary by two different points of view: "where" and "how" it operates. As for "where," the adversary can be:

1. *Localized*: The adversary chooses a convex subarea of the network and compromises sensors from that area only.
2. *Ubiquitous*: The adversary compromises sensors choosing from the whole network.

Intuitively, the localized adversary describes an adversary that needs some time to move from one point to another of the network area, while the ubiquitous adversary, during the same time interval, can capture nodes regardless of their position.

As for the sequence of node capture (that is, "how"), the adversary can be:

1. *Oblivious*: At each step of the attack sequence, the next node to be tampered with is chosen randomly among the ones that are yet to be compromised.
2. *Smart*: At each step of the attack sequence, the next node to tamper with is node s, where s maximizes $P_{witness}(s)$, $s \in \mathcal{N} \backslash \mathcal{W}$.

Intuitively, the oblivious adversary does not take advantage of any information about the detection protocol implemented. Conversely, the smart adversary greedily chooses to compromise the node that maximizes its *appeal* in order to maximize the chance that its replicas go undetected.

4 Requirements for the Distributed Detection Protocol

In this section, we present and justify the requirements that a protocol for clone detection should meet.

4.1 Witness Distribution

A major issue in designing a protocol to detect clone attacks is the selection of the witnesses. If the adversary knows the future witnesses before the detection protocol executes, the adversary could subvert these nodes so that the attack goes undetected. The adversary can, in principle, use any information on the network to foresee probability $P_{witness}(s)$ for a generic node s. Here, we have identified two kinds of predictions:

- ID-based prediction and
- location-based prediction.

We say that a protocol for replica detection is *ID-oblivious* if the protocol does not provide any information on the ID of the sensors that will be the witnesses of the clone attack during the next protocol run. Similarly, a protocol is *area-oblivious* if probability $P_{witness}(s)$, for every $s \in \mathcal{N}$, does not depend on the geographical position of node s in the network. Clearly, when a protocol is neither ID-oblivious nor area-oblivious, then a smart adversary can have good chances of succeeding, since it is able to use this information to subvert the nodes that, most probably, will be the witnesses. Furthermore, when a protocol is not area-oblivious, then even a localized oblivious adversary (that, at a first glance, seems to be the weakest) can be effective if it concentrates node compromising activities in an area with a high density of witnesses.

4.2 Overhead

Designing protocols for wireless sensor networks is a challenging task due to the resource constraints typical of these networks. Any protocol is required to generate little overhead. However, this requirement alone is not enough. Indeed, even if a protocol shows a reasonably small overhead on average, it is still possible that a small subset of the nodes experiences a much higher overhead. This is bad—these nodes exhaust their batteries very quickly, with serious consequences on the network functionality. Moreover, the problem can be even more subtle when we consider memory. If a high memory overhead concentrates on a small number of nodes, then these nodes can overflow. During an overflow, the node could stop the protocol, or drop packets to free memory. It is very important to understand what kind of an impact this scenario can have on the detection capability of the protocol itself.

We can summarize the above considerations with the general requirement that the overhead generated by the protocol should be small, which is sustainable by the network as a whole, and (almost) evenly distributed among the nodes. To make a real example, in LSM, every node that relays a position claim must perform a signature verification and store the claim. As analyzed in [45], every line segment includes $O(\sqrt{n})$ nodes and every node stores $O(\sqrt{n})$ location claims. Note that this memory requirement could be impractical in real networks with thousands of nodes.

Table 1 shows—first row—the asymptotic overhead for one protocol run (also referred to as *round* in the following) of LSM. The second row reports, on average, overhead

generated by one round of LSM for a network of 1,000 nodes with 31 neighbors per node (on average). Finally, the third row shows the maximum overhead experienced by a node that turns out to be much higher than the average. Detailed discussion on the overhead of LSM and the protocol we propose and their compliance with the above described requirements are presented in Section 6.

5 RED

In this section, we propose RED, a new protocol for the detection of clone attacks. RED is similar, in principle, to the Randomized Multicast protocol [45], but with witnesses chosen pseudorandomly based on a network-wide seed. In exchange for the assumption that we are able to efficiently distribute the seed, RED achieves a large improvement over RM in terms of communication and computation. When compared with LSM [45], a protocol that is more efficient than RM, RED proves to be again considerably more energy efficient. More than that, in the following sections, we will show that RED is also more robust against attacks that exploit the uneven distribution of witnesses of LSM.

As in LSM, we assume that the nodes in the network are relatively stationary; each node knows its own location (for instance, using a GPS or the protocols in [8], [44]); and all the nodes use an ID-based public key cryptosystem [12], [46]. We also assume that the network is loosely time synchronized. Observe that loose time synchronization can be achieved both in a centralized and in a distributed way [31], [32], [45].

RED executes routinely at fixed intervals of time. Every run of the protocol consists of two steps. In the first step, a random value, *rand*, is shared among all the nodes. This random value can be broadcasted with centralized mechanism (for example, from a satellite or a UAV [40], or other kinds of ground-based central stations), or with in-network distributed mechanisms. For instance, a secure, verifiable leader election mechanism [10], [48], [22] can be used to elect a leader among the nodes; the leader will later choose and broadcast the random value. In the rest of this paper, without loss of generality and to ease exposition, we will rely on a centralized solution for the broadcast of the random value. Furthermore, in this work, we assume that a different mechanism is used to enforce nodes not to lie about their physical location. As an example, neighbor nodes can physically check the coherence of the claimed location. Such a simple mechanism, also used in [45], has the following drawback: if all the neighbors of a cheating node c are corrupted, they will not identify c as a cheater. Hence, this is a drawback of both our protocol and LSM [45].

In the second step, each node digitally signs and locally broadcasts its claim—ID and geographic location (Procedure BROADCAST_CLAIM in Protocol 1). When the neighbors receive the local broadcast, they execute Procedure RECEIVE_MESSAGE. Each of the neighbors sends (with probability p) the claim to a set of $g \geq 1$ pseudorandomly selected network locations (rows 17-24 in Protocol 1). RED does not send the claim to a specific node ID because this kind of a solution does not scale well: A claim sent to a node ID that is no more present in the network would be lost; nodes deployed after the first network deployment could not be used as witnesses without updating all the nodes. However, RED can easily be adapted to work when a specific node is used as the message destination. Finally, in the following, we consider the same geographic routing protocol used in [45] for a fair comparison. Though, RED is actually independent of the routing protocol used in the network.

Algorithm 1. RED

```
 1: Procedure BROADCAST_CLAIM
 2:     claim ← ⟨ID_a, is_claim, location(), time()⟩
 3:     signed_claim ← ⟨claim, K_a^priv(claim)⟩
 4:     a → neighbors(): ⟨ID_a, neighbors(), signed_claim⟩
 5: end procedure
 6: procedure RECEIVE_MESSAGE (m)
 7:     if is_claim(m) then
 8:         ⟨−, −, signed_claim⟩ ← m
 9:         ⟨claim, signature⟩ ← signed_claim
10:         if bad_signature(claim, signature) then
11:             discard m
12:         else if incoherent_location(claim) then
13:             ⟨ID_x, −, −, −⟩ ← claim
14:             trigger revocation procedure for ID_x
15:             return
16:         end if
17:         do with probability p
18:             ⟨claim, signature⟩ ← signed_claim
19:             ⟨ID_x, −, loc_x, time_x⟩ ← claim
20:             locations ← pseudo_rand(rand, ID_x, g)
21:             for all l ∈ locations do
22:                 a → l: ⟨ID_a, l, is_fwd_claim, signed_claim⟩
23:             end forall
24:         end do
25:     else if is_fwd_claim(m) then
26:         ⟨−, −, −, signed_claim⟩ ← m
27:         ⟨claim, signature⟩ ← signed_claim
28:         if bad_sig(signed_claim) or replayed(claim) then
29:             discard m
30:         else
31:             ⟨ID_x, −, loc_x, time_x⟩ ← claim
32:             if detect_clone(memory, ⟨ID_x, loc_x, time_x⟩) then
33:                 trigger revocation procedure for ID_x
34:             else
35:                 store fwd_claim in memory
36:             end if
37:         end if
38:     end if
39: end procedure
```

We assume that the routing delivers a message sent to a network location to the node closest to this location [8], [39]; that the routing protocol does not fail (as done in [45]); and that message forwarding is not affected by dropping or wormhole attacks (for these kinds of attacks, a few solutions can be found in [20], [35], [38]). We also assume that the adversary is able to compute the set of witnesses. However, it cannot immediately compromise them since, by the time, it moves to reach those nodes, the protocol runs committed. Later, in Section 7, we will see how the protocol performs

when malicious nodes can drop packets. To test the protocols, we assume that the adversary has introduced two nodes with the same ID in the network. Clearly, if the adversary introduces more clones of the same node, then the task of detecting the attack is only easier. Within this ideal framework, the probability that the clone attack is detected is equal to the probability that at least one neighbor of each clone sends the claim to the same witnesses. Considering d neighbors, the probability that from a neighborhood, a claim message is sent is $1-(1-p)^d$; therefore, the detection probability is $(1-(1-p)^d)^2$. For example, with $p=0.1$ and $d=35$, we have a detection probability of 0.95 in a single run of the protocol. Detection probability will be further discussed in a more realistic framework in Section 6.

The set of witnesses is selected using a pseudorandom function (line 20 of Protocol 1). This function takes in input the ID of the node that is the first argument of the claim message, the current *rand* value, and the number g of locations that have to be generated. Using a pseudorandom function guarantees that, given a claim, the witnesses for this claim are unambiguously determined in the network, for a given protocol iteration, while time synchronization is used by the nodes to discern between different iterations. Furthermore, note that the detection probability is mainly influenced by both protocol parameters p, g and node density. Hence, the results hold even for network with a reasonably small number of nodes, and not only with high probability (whp).

Every node signs its claim message with its private key before broadcasting it (line 3 of Protocol 1). The nodes that forward the signed claim toward destination are not required to add any signature or to store any message. Further, relay nodes do not need to check the signature of the routed message: Signature check will be carried out by the destination only—saving $\Theta(\sqrt{n})$ signature checks per claim sent with respect to LSM. Indeed, while signature check at each node is necessary in LSM for the detection of the attack, it is not in RED. However, these checks can be useful in LSM to prevent the adversary from generating spurious messages. In RED, if the adversary sends out location claims with bogus signatures, the legitimate neighbors will then forward the message throughout the network, and the bad signature will not be detected until it reaches the recipient. This attack does not affect clone detection itself—its goal is to exhaust the nodes' battery. In this sense, this attack is out of the scope of this paper. However, in the following, we sketch a possible countermeasure: Relay nodes can store some bits regarding forwarded messages so that it is possible to trace back the message originator, in the case of failure of the signature check at the destination. Note that this countermeasure does not introduce a significant overhead.

For every received claim, the destination (possible witness of a clone attack):

- verifies the received signature (line 28); and
- checks for message freshness (line 28). This is important to prevent replay of old messages. This check is performed verifying the coherence between the time inserted in the message by the claiming node and the current time.

For every genuine message that passes the previous checks, the possible witness node extracts the information (ID and location). If this is the first claim carrying this ID, then the node simply stores the message (line 35). If another claim from the same ID has been received, the node checks if the new claim is coherent with the claim stored in memory for this ID (line 32). If it is not, the witness triggers a revocation procedure for the ID (line 33)—the two incoherent signed claims are the proof of cloning.

Here is an example of a run of the protocol. Assume that the adversary clones identity ID_a and assigns this identity to nodes a and a'. These two nodes are placed in two different network locations: l_1 and l_2, respectively. During an RED iteration, the nodes a and a' have to broadcast the same ID, but different location claims (l_1 and l_2). Indeed, if $l_1 \sim l_2$, then either the neighbors of a or the neighbors of a' will raise an exception (line 14 of Protocol 1).

Let b and c be neighbors of a and a', respectively. Using the pseudorandom function, both b and c will select the same set of witness nodes, containing at least a node w. In this way, w will receive two incoherent location claims for identity ID_a—l_1 and l_2. This results in clone detection. Hence, w can start a revocation procedure for node ID_a. Revocation can be performed by flooding the network with the two incoherent claims received by w. Remember that every claim message of a node is signed with the private key of the same node. Therefore, the two claims are a proof that ID_a has been cloned.

The protocol shows one caveat: After the *rand* value is shared, RED allows the adversary to know the witness set for any given ID. However, note that the witnesses of a node could be anywhere in the network and witnesses change at every protocol iteration in an unpredictable way. This means that the adversary, in order to prevent RED from detecting the replicas, is required to be extremely fast and to capture all the witnesses of the clones within a window period that can be at most comprised between the disclosure of *rand* and the end of the protocol round. Considering realistic network sizes and the possible adversary speed, there are few chances for the adversary to perform this attack.

6 Simulations

In this section, we show that RED meets the requirements described in Section 4: Area-obliviousness; ID-obliviousness; low overhead; overhead balancing; and high replica attacks detection probability. We further compare RED with LSM and show that RED outperforms LSM in several ways.

In the following simulations, we consider a unit square deployment area [4], [5], [26]. We fixed $n=1{,}000$ nodes in the network and $r=0.1$ communication radius. We also set $g=1$ and $p=0.1$ for both protocols. This means that the two protocols send the same number of location claims per node (on average). Further, we assume that the nodes are distributed in the network area uniformly at random. We simulate the same geographic routing protocol used in [45]—the relay node is the neighbor closest to destination. The routing stops when no node is closer to destination than the current node: This node will be a witness. Note that this simple version of geographic routing, especially when

TABLE 2
RED Overhead ($n = 1{,}000, r = 0.1$, and $g = 1$)

	Memory Occupancy	Sent Messages	Received Messages	Signature Check
Asymptotic	$O(g \cdot p \cdot d)$	$O(g \cdot p \cdot d \cdot \sqrt{n})$	$O(g \cdot p \cdot d \cdot \sqrt{n})$	$O(g \cdot p \cdot d)$
Average (p=0.1)	0.93	22.08	49.85	2.87
Max (p=0.1)	15	220	250	48
Average (p=0.05)	0.75	11.36	39.80	1.48
Max (p=0.05)	6	67	98	11

used in networks that are sparse or deployed in an area that is not convex, has the problem of "dead ends"—places where the message cannot proceed because there is no node closer to destination, while the destination is still far. There are a few solutions to this issue [30], [8] that can be used in both protocols to guarantee that the claim reaches the node closest to destination.

The resources required by RED are shown in Table 2 for the same parameters used for LSM in Table 1. The results in Table 2 do not consider the overhead due to the distribution of the random seed. The overhead of the distribution depends on the implementation: It is practically negligible (one receive and one signature check) compared to the cost of one iteration of RED if the seed is broadcast from either a satellite or a UAV; it is also very small (one broadcast operation) if the seed is distributed within the network from a trusted base station. A robust broadcast operation (e.g., [30]) can be easily implemented in such a way that every node is required to receive the message and send at most one local broadcasting message. If this is the case, the per-node additional overhead to RED is at most one send, one receive, and one signature check. Lastly, if a leader election algorithm has to be run, such as the ones in [10], [48], [22], the overhead of the leader election should be taken into account and also its vulnerabilities. A reasonable choice, however, would be to run the algorithm only once and follow a fixed schedule after the initial election. In this way, the overhead would be amortized through the many iterations of the RED protocol and the actual overhead would be very similar to a broadcast operation, though the adversary could predict the schedule and get some advantage from this information. To summarize, it seems reasonable that RED should be used when we can implement an efficient solution for the distribution of the seed, either a broadcast from a satellite, from a UAV, or by in-network distribution of a seed from a trusted base station. This is the assumption that we will be using in the rest of the paper.

6.1 Witness Distribution

Due to randomization, it is straightforward to verify that both LSM and RED are ID-oblivious. In both protocols, the IDs of the witnesses are randomly selected among all the nodes in the network. To assess area-obliviousness, we study the witness distribution as follows: We select increasing subareas of the network, and for each subarea, we count the number of witnesses present in the area after a run of the detection protocol. Each subarea from the center of the unit square toward the external border provides an increment of five percent of the total area. Hence, 20 subareas are considered.

In Fig. 2, we show an example of one iteration of LSM and RED. The black small filled squares indicate two clones (nodes with the same ID), the black filled small circles indicate nodes that route a claim from the clones, and finally, the larger empty circle indicates the witness. Lastly, the square at the center of the network indicates a central area whose size is 20 percent of the total area of the network. This example suggests that the witness nodes (large not filled circles) are located differently in LSM and RED. In the particular example, several claims are forwarded in LSM, while only one in RED (this is a real run of the two protocols with the same parameters, on average, the number of claims forwarded is the same for the two protocols if the parameters are the same). However, the figure is intended to show that the witness is near the center of the network area for LSM, while this is not true for RED, and to suggest that this is not a coincidence. In the following, we will see through extensive simulations that this is actually true—witnesses are mostly in the center of the network with LSM, while are uniformly distributed with RED—and this phenomenon affects the performances of the protocols.

Fig. 3 reports, for the two protocols, on the percentage of witnesses present in the incremental subareas. We simulate 10,000 different network deployments. For each deployment, we randomly select two nodes, assign to them the same ID, and execute a single LSM iteration and a single RED iteration. After each of these iterations, we localize the witness nodes for the two different protocols. Finally, for each of the 20 incremental subareas, we compute the percentage of witnesses with respect to the total number of witnesses. After collecting the outcome of 10,000 experiments, we plot the average. The x-axis in Fig. 3 indicates the percentage of the network area considered, while the y-axis the corresponding percentage of all the witnesses in that area.

In Fig. 3, we can see that the central area, corresponding to the 20 percent of all the network area, collects more than 50 percent of all the witnesses of LSM, while the most external area, corresponding to the 20 percent of the network area, contains only 1.75 percent of all the witnesses. Therefore, LSM is not area-oblivious, since $P_{witness}(s_i) > P_{witness}(s_j)$ for an s_i selected from the central area and an s_j selected from the most external area. This is a direct

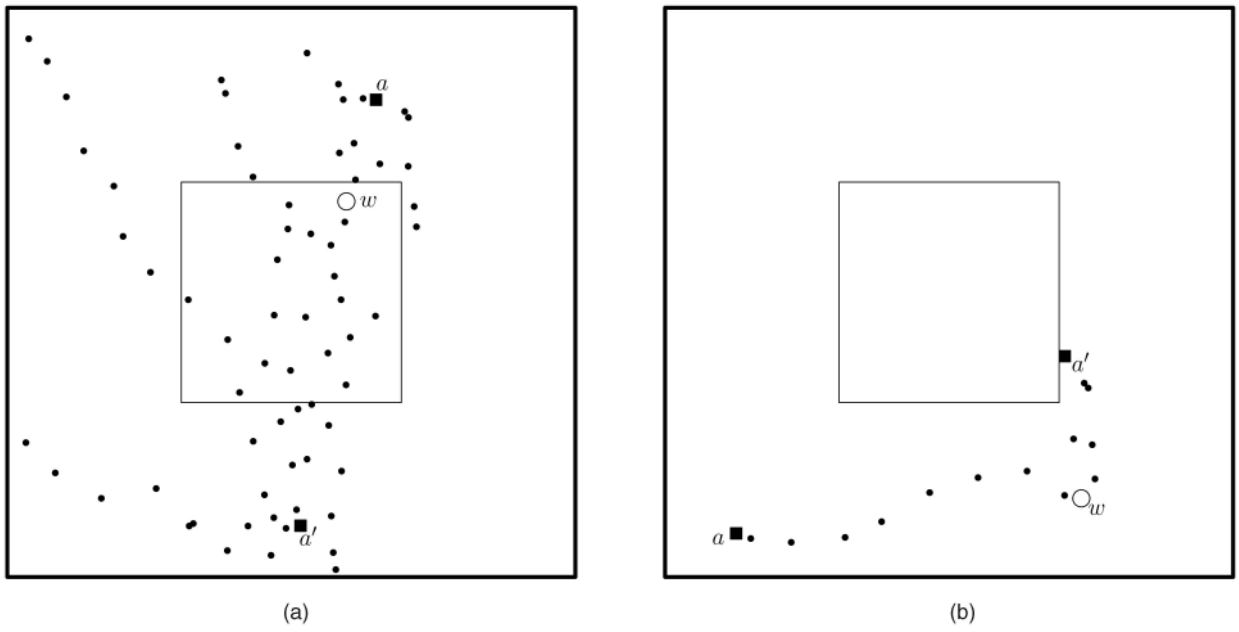

Fig. 2. Examples of protocols iteration: $n = 1{,}000$, $r = 0.1$, $g = 1$, and $p = 0.1$. (a) LSM protocol. (b) RED protocol.

consequence of the following fact: Paths are generated having random source and random destination; hence, two paths are more likely to intersect in the middle of the deployment area than close to the borders. This phenomenon is known in the literature [41], [42] and it is due to the fact that most of the shortest paths generated by a uniform traffic traverse the center of the network. Whereas it is straightforward to prove that the RED protocol, due to the pseudorandom choice of witness nodes, has a uniform witnesses distribution. In fact, Fig. 3 also shows how the behavior of RED corresponds to that of an ideal protocol: The witnesses are equally distributed in all the network areas. In other words, RED is area-oblivious.

6.2 Storage Overhead

Fig. 4 reports the number of messages that the nodes are required to store for LSM and RED. For a fixed x-value of messages in memory, we show the percentage of the nodes that needs to store that number of messages. The values were obtained averaging the result of 10,000 simulations.

Note that for LSM, some nodes could require to store as many as 200 messages. Our experiments show that LSM requires some 1.9 percent of the nodes to store more than 60 messages, some 7.6 percent of the nodes to store a number of messages between 40 and 59, and some 27.5 percent of the nodes to store a number of messages between 20 and 39. Just some 63 percent of the nodes are required to store less than 20 messages. As for RED, only a negligible percentage of the nodes (0.001 percent) requires to store more than 10 messages. Moreover, some 0.3 percent of the nodes need to store more than five messages and less than 10 percent of the nodes to store a number of messages between three and five. It is interesting to note that 47.7 percent of the nodes need to store only one or two messages, while 42.9 percent of the nodes do not require to store any message at all. Finally, observe that

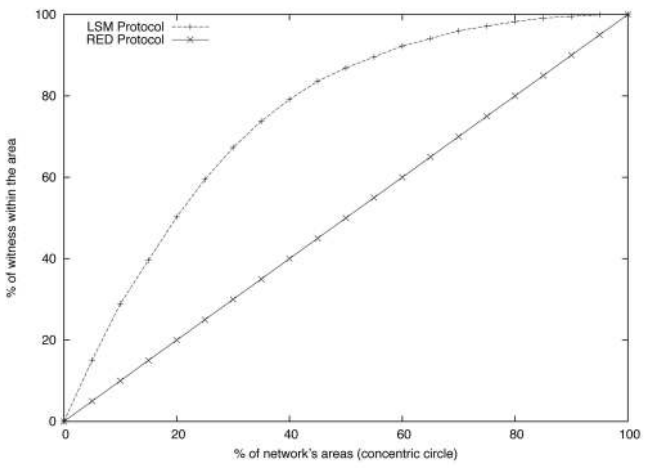

Fig. 3. Witness density. $n = 1{,}000$, $r = 0.1$, $g = 1$, and $p = 0.1$.

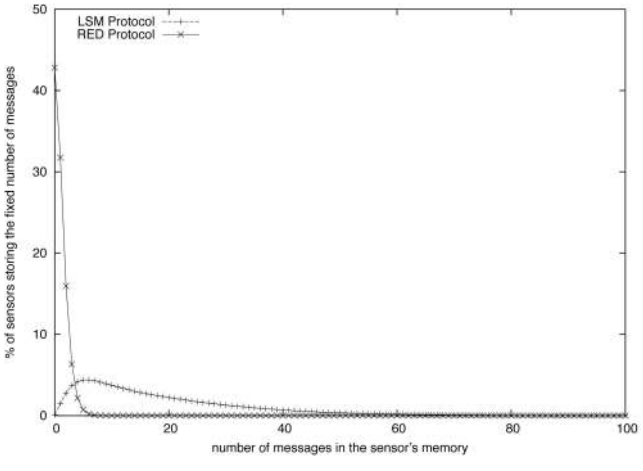

Fig. 4. Used memory for both RED and LSM. $n = 1{,}000$, $r = 0.1$, $g = 1$, and $p = 0.1$.

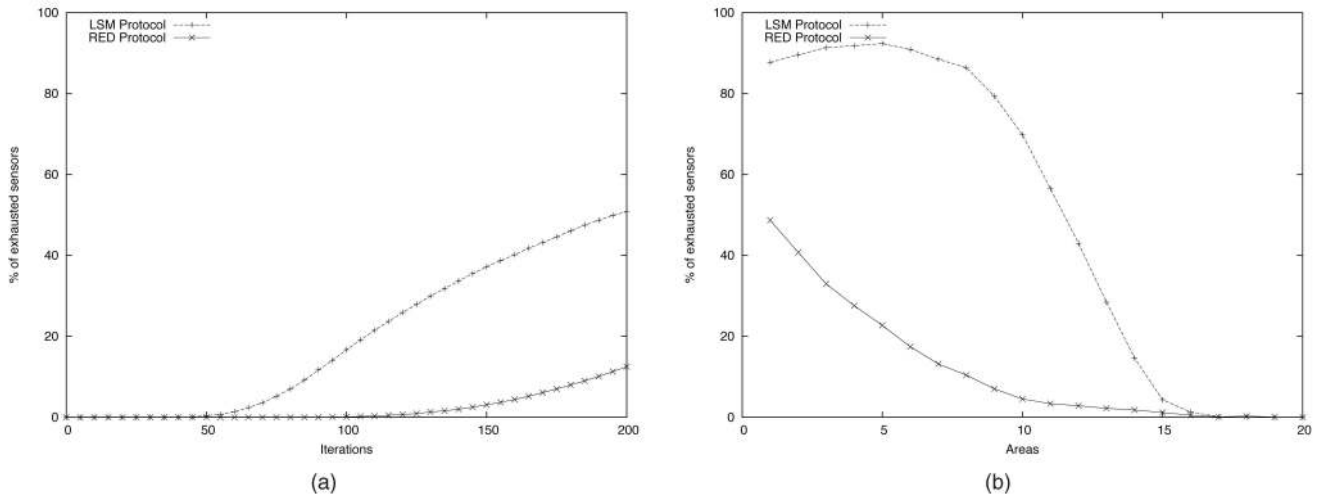

Fig. 5. Node exhaustion behavior: $n = 1{,}000$, $r = 0.1$, $g = 1$, and $p = 0.1$. (a) Exhausted node in different iterations. (b) Exhausted node distribution after 200 iterations.

for LSM, only 0.2 percent of the nodes do not require to store any message. LSM requires to store a higher number of messages compared to RED. This follows from the fact that, in LSM, every node in a claim path is a possible witness, and therefore, has to store every claim it relays. In RED, only destinations can be witnesses, and thus, only destinations are required to store the claims.

6.3 Energy Overhead

To assess the energy overhead of the two protocols, we consider both communication and computation-intensive operations (that is, public key cryptography: signature generation and signature verification). In particular, we use the energy model proposed in [49]: A node battery of 324,000 mJ; 15.104 mJ for sending a packet; 7.168 mJ for receiving a packet (assuming packet length of 32 byte, 0.059 mJ for bit sending, and 0.028 mJ for bit receiving); and 45.0 mJ for both signature generation and signature verification.

The operating life of a node depends on its battery. Different energy overheads for the two protocols will result in a different pattern of node exhaustion. Fig. 5a shows this phenomenon. After 100 protocol run executed with the same network topology, some 20 percent of the nodes are exhausted for LSM, while for RED, all the nodes are alive. After 150 iterations, LSM shows 40 percent of exhausted nodes, while RED only five percent. Finally, after 200 runs, LSM shows that half of the nodes of the network are exhausted (further, with such a number of exhausted nodes, the efficiency of LSM as for clone detection drops dramatically), while for RED, this percentage is less than 15 percent and the detection capabilities are still remarkable. It is also interesting to look at the different nodes exhaustion distribution in the network area. Fig. 5b shows the distribution after 200 protocol iterations. The x-axis indicates the network subareas (as plotted in Fig. 5a), numbered sequentially from the center (numbered as 1) to the external one (numbered as 20). The y-axis indicates the percentage of exhausted nodes in these areas. For both protocols, most of the exhausted nodes are in the center. As observed in Section 6.1, this is due to the phenomenon that nodes in the center of the network are more involved in routing messages in the presence of uniform traffic. In the case of LSM, almost all the nodes in the center are exhausted (except a few isolated ones), and the overhead is transferred to the semicentral areas, leading to the shape in Fig. 5b.

Different distribution of node exhaustion also implies different clone attack detection probability, as shown in Fig. 6. This figure shows the detection probability (y-axis) at different protocol iterations (x-axis). In particular, we plotted the detection probability for the first 200 runs. Plotted values were computed averaging the results obtained for 10,000 network deployments. Each single deployment was evaluated for both the LSM and the RED protocol. For all the considered iterations, the RED protocol shows a better detection probability compared to that of the LSM. From the 1st to the 50th iteration, LSM shows a probability detection of about 35 percent, while this probability is more than 80 percent for the RED protocol. However, it is important to note that LSM has been designed to use at least five witness nodes. If parameters

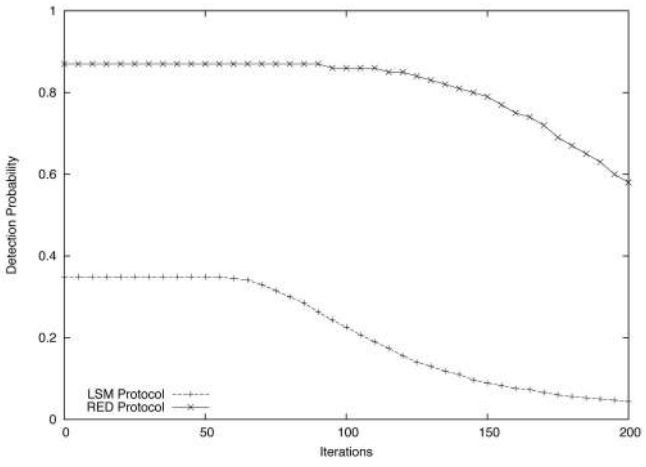

Fig. 6. Detection probability for both RED and LSM. $n = 1{,}000$, $r = 0.1$, $g = 1$, and $p = 0.1$.

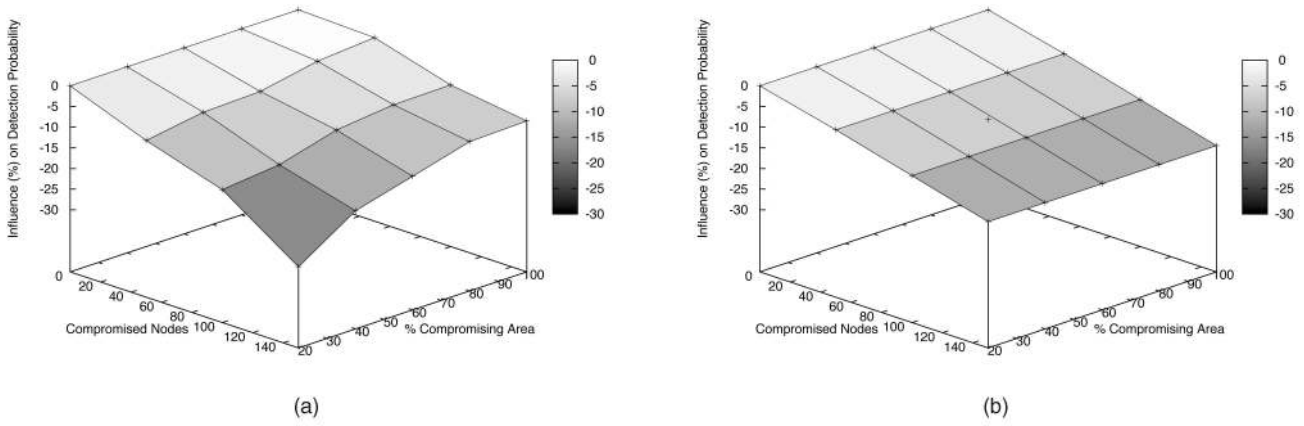

Fig. 7. Detection probability with compromised nodes. $n = 1{,}000$, $r = 0.1$, $g = 1$, and $p = 0.1$. (a) LSM protocol. (b) RED protocol.

are set in such a way to make LSM use at least five witnesses, the detection probability of LSM gets similar to the detection probability of RED, but with a much higher overhead. Our choice has been set to the parameters in such a way that the overhead of one iteration of RED and one iteration of LSM is similar, in order to make a fair comparison from a performance standpoint. It is interesting to note the tight relationship between the percentage of exhausted nodes (Fig. 5a) and the detection probability (Fig. 6). For LSM, nodes start exhausting after some 50 iterations; at the same iteration number, the detection probability starts decreasing. A similar behavior could be observed for RED as well. It is also possible to note that different slopes of the curves representing node exhaustion correspond to different slopes in curves representing detection probability.

Finally, we simulated the protocol behavior under a coordinated attack: The adversary clones a node into two copies and in the same period of time, compromises a subset w of the other remaining nodes. In this setting, we assume that a compromised node forwards messages like an honest one: If not, this behavior could be detected, like in [20], [35], [38]. However, when a compromised node is a witness, we assume that it would not trigger any alarm, and the clones would go undetected for this specific protocol iteration. We investigated how the detection probability is affected under the above scenario, assuming that the adversary "smartly" compromises nodes from a so-called compromising area, which is a squared central area of the network of increasing size.

When no nodes are compromised, for the first 50 protocol runs, the detection probability is 87 percent for RED and 33.8 percent for LSM, as shown in Fig. 6. When taking into account node compromising, the results of our simulations for LSM and RED are shown in Figs. 7a and 7b, respectively. On the x-axis, we indicate the number of compromised nodes, while on the y-axis, the percentage of the total network, starting from the inner area. We can notice that for LSM, the detection probability is influenced by both the number of compromised nodes and the size of the compromising area (Fig. 7a). As for RED, the detection probability is influenced only by the number of compromised nodes (Fig. 7b). This is due to the following fact: As observed in Fig. 3, LSM shows an higher witness density in the most internal areas. For instance, capturing 150 nodes in the 20 percent central area implies a reduction of detection probability of 25.4 percent for LSM (from 33.8 to 25.2 percent), while the performance of RED is reduced by 14 percent only (from 87 to 74.8 percent). We can also note that when the same number of nodes are compromised in all the network areas, the relative resilience of LSM is a little bit higher than RED. For example, with 150 compromised nodes all over the network area, LSM decreases its detection probability by 8.5 percent only, while it is about 14 percent for RED. This is due to the particular behavior of LSM: More than one node can witness a clone attack; compromising a witness node does not imply that a clone attack will go undetected for the LSM, while this can be true for RED.

7 Detection Probability with Malicious Nodes

In this section, we investigate clone detection probability during a sequence of iterations. We assume that the adversary has cloned a node, it is also already controlling a subset of w randomly selected other nodes, and no mechanism for preventing packet dropping is implemented, so that malicious nodes can stop claim forwarding.

Further, we assume that a node (say a) is cloned and one of its clone (say a') is randomly deployed within the network area. Moreover, we assume no routing failure and from each neighborhood, exactly one claim message is sent (we do not explicitly consider d, p, and g values). We leverage the hypothesis that both claims are sent through path of length $\ell = c\sqrt{n}$ nodes (with constant network density, the average path length is $\Theta(\sqrt{n})$). The nodes on the two paths (the first one departing from the honest node a and the second one from the clone a') are those involved in the detection process by the two protocols.

In RED, if just one of these 2ℓ nodes in the two paths is malicious, detection can fail. In fact, note that the corrupted forwarding node can simply drop the received location claim. The probability that at least one malicious node is present in the two paths is

$$1 - \frac{\binom{n-w}{2\ell}}{\binom{n}{2\ell}}. \quad (1)$$

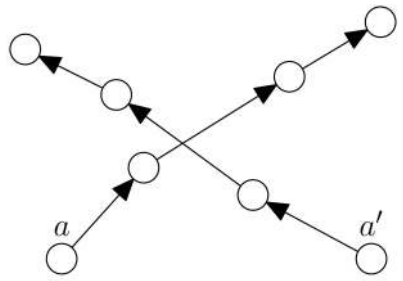

Fig. 8. Example of intersecting paths without intersection node in LSM.

The probability that the attack is not detected using the RED protocol, for a single protocol iteration, is exactly that of (1). To analyze a sequence of iterations, we assume that every iteration is probabilistically independent. Therefore, the probability that the attack is not detected after i RED protocol iterations is

$$\left(1-\frac{\binom{n-w}{2\ell}}{\binom{n}{2\ell}}\right)^{i}. \tag{2}$$

The analysis is different for LSM. In fact, even if all the nodes are honest, the attack is detected only with a given probability—the probability that two paths starting at a and a' intersect on a network node. Following the analysis proposed in [45], this probability is

$$\frac{1}{3}\left(1-\frac{35}{12\pi^2}\right). \tag{3}$$

However, note that the probability in (3) refers to geometric line intersection. Then, it is, in fact, an optimistic upper bound (also still assuming no failure in the routing). In fact, two intersecting paths (geometrically) do not necessarily have a node in common—an example of this case is shown in Fig. 8. Despite this fact, in the following, we optimistically consider (3) as the probability that the clone is detected when no malicious nodes are present.

Let U be the event that the attack is undetected for a single protocol iteration. For LSM, we have to consider the following two disjoint events. Here, the idea is that malicious nodes can prevent clone detection only if they are in the path *before* the witness. Let us define:

- Event E_h: All of the forwarding nodes before the (possibly present) witness are honest.
- Event E_m: There is at least one malicious forwarding node before the (possibly present) witness.

Note that E_h and E_m form a partition of the probability space; hence,

$$Pr[U] = Pr[U|E_h]Pr[E_h] + Pr[U|E_m]Pr[E_m], \tag{4}$$

where $Pr[U|E_h]$ is the probability that the attack is undetected when there are no malicious nodes in the paths. According to [45], this is equal to

$$1-\frac{1}{3}\left(1-\frac{35}{12\pi^2}\right) = \frac{1}{3}\left(2+\frac{35}{12\pi^2}\right). \tag{5}$$

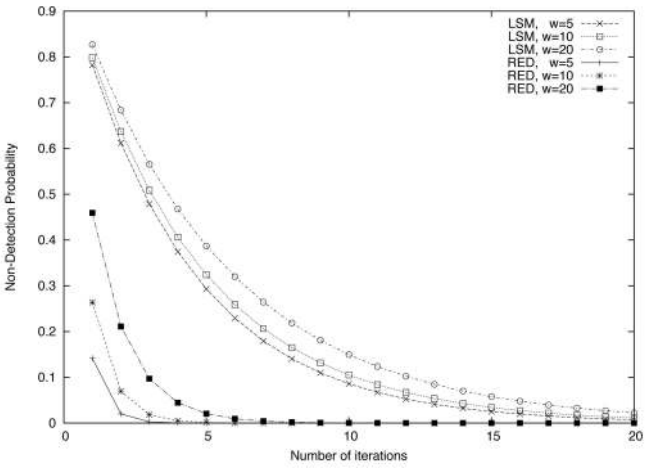

Fig. 9. Nondetection probability ($n = 1{,}000$ and $c = 0.5$).

We can assume that $Pr[U|E_m] = 1$, since the malicious node before the witness can discard the claim and stop the detection. $Pr[E_m] = 1 - Pr[E_h]$ is similar to (1), taking into account that the malicious nodes should appear before the witness. On average, the witness is in the middle of the paths; therefore, we can estimate this probability as follows:

$$Pr[E_m] = 1 - \frac{\binom{n-w}{\ell}}{\binom{n}{\ell}}.$$

Putting it altogether, we can compute $P(U)$ as follows:

$$Pr[U] = 1 + \frac{\binom{n-w}{\ell}}{\binom{n}{\ell}}\left(\frac{35}{36\pi^2}-\frac{1}{3}\right).$$

Therefore, the probability that the attack is not detected after i LSM protocol iterations is

$$\left[1 + \frac{\binom{n-w}{\ell}}{\binom{n}{\ell}}\left(\frac{35}{36\pi^2}-\frac{1}{3}\right)\right]^{i}. \tag{6}$$

Fig. 9 shows the analytical results for RED and LSM on nondetection probability. Remind that while the analysis for RED is essentially tight, the one for LSM is optimistic, since it depends on the assumption that paths that geometrically intersect have a node in common. This is not true, especially when the network is dense. The actual detection rate depends on several factors like node density, for example. Nonetheless, RED outperforms LSM even in the presence of malicious nodes that can stop the protocol. Fig. 10 shows the analytical results for several values of c (c controls the length of the average random path in the network, being $\ell = c\sqrt{n}$) of the nondetection probability. We considered subsequent protocol iterations (x-axis). We plotted the result for $c = 0.1, 0.2, \ldots, 1$.

It is interesting to note that how w and c influence the detection probability. Larger c means longer paths, and thus, higher probability that one of the malicious nodes can stop clone detection. Larger w means that the adversary can often thwart the protocols and influence detection probability considerably, especially when c is large. In all cases, it is clear that RED can converge to very high detection probability very quickly. Note that RED is more influenced

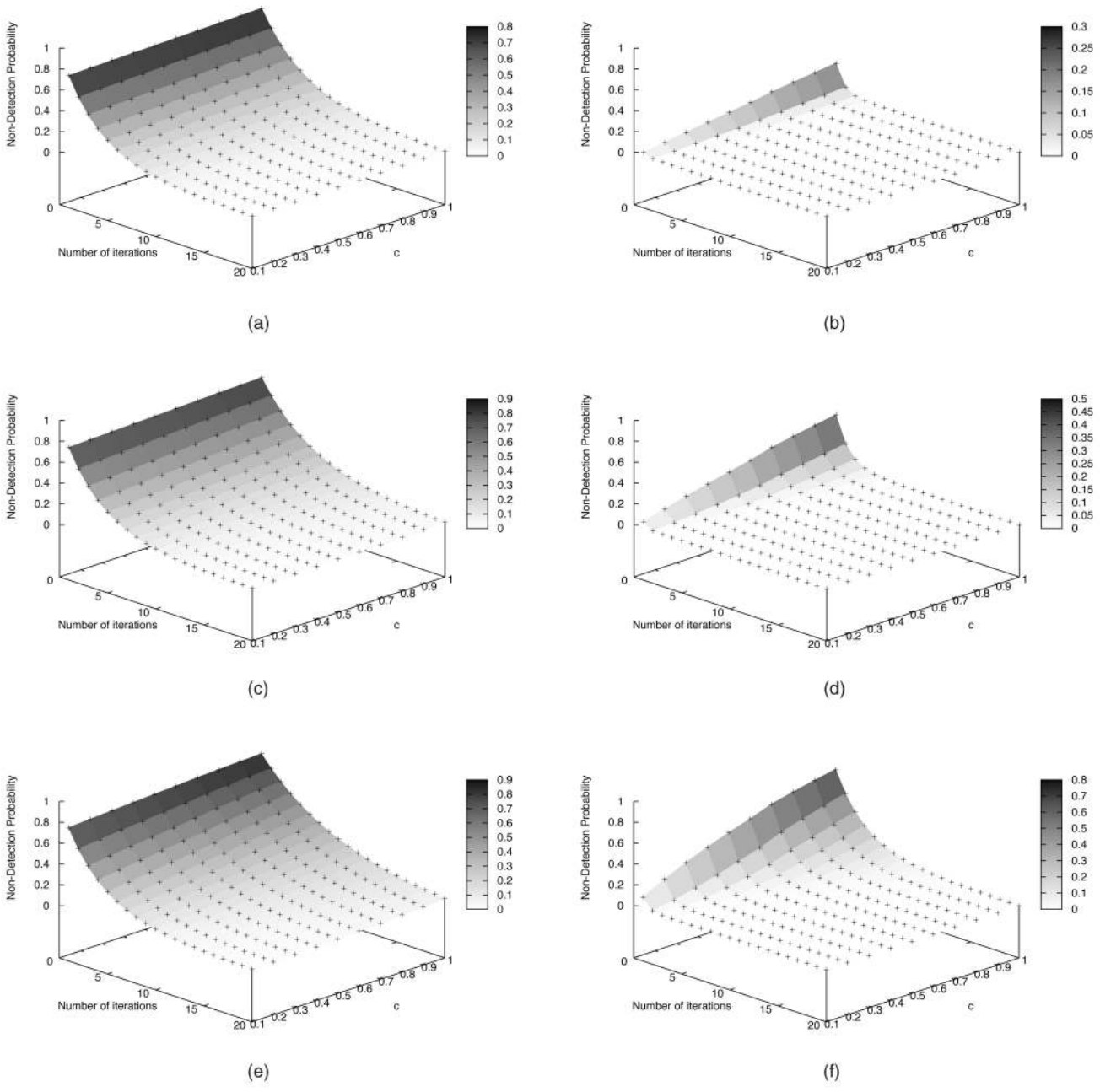

Fig. 10. Nondetection probability for both RED and LSM. $w = 5, 10, 20$. (a) LSM, $w = 5$. (b) RED, $w = 5$. (c) LSM, $= 10$. (d) RED, $w = 10$. (e) LSM, $w = 20$. (f) RED, $w = 20$.

than LSM by path lengths, since a malicious node can stop the protocol wherever it appears in the paths. However, experiments show that for a network of 1,000 nodes and communication range 0.1 in a network area of side one, c is about 0.35. Therefore, we can conclude that RED has better detection probability and converges faster than LSM for all practical values of the network parameters.

8 Concluding Remarks

In this paper, we presented and justified a few basic requirements an ideal protocol for distributed detection of node replicas should have. In particular, we have introduced the preliminary notion of *ID-obliviousness* and *area-obliviousness* that convey a measure of the quality of the node replicas detection protocol; that is, its resilience to a smart adversary. Moreover, we have indicated that the overhead of such a protocol should be not only small, but also evenly distributed among the nodes, both in computation and memory. Further, we have introduced new adversary threat models. However, a major contribution of this paper is the proposal of a self-healing, randomized, efficient, and distributed protocol to detect node replication attacks. We analytically compared RED with the state-of-the-art solution (LSM) and proved that the overhead introduced by RED is low and almost evenly balanced among the nodes; RED is both ID-oblivious and area-oblivious; furthermore, RED outperforms LSM in terms of

efficiency and effectiveness. Extensive simulations confirm these results. Lastly, also in the presence of compromised nodes, we can analytically show that RED is more resilient in its detection capabilities than LSM.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their comments that helped to improve the quality of this paper. The authors are solely responsible for the views expressed in this paper, which do not necessarily reflect the position of supporting organizations. This work was partly supported by:

1. the FP7 EU "EXTRABIRE" Project of the Prevention, Preparedness and Consequence Management of Terrorism and other Security-related Risks Programme -European Commission - Directorate General Home Affairs, www.extrabire.eu,
2. the FP7 EU Project "SENSEI", Integrating the Physical with the Digital World of the Network of the Future, Grant Agreement Number 215923, www.ict-sensei.org,
3. the Spanish Ministry of Education through projects TSI2007-65406-C03-01 E-AEGIS and CONSOLIDER INGENIO 2010 CSD2007-0004 "ARES",
4. the Government of Catalonia under grant 2005 SGR 00446.

REFERENCES

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Int'l J. Computer and Telecomm. Networking,* vol. 38, no. 4, pp. 393-422, 2002.

[2] R. Anderson and M.G. Kuhn, "Tamper Resistance—A Cautionary Note," *Proc. USENIX '96 Workshop,* pp. 1-11, 1996.

[3] A. Becher, Z. Benenson, and M. Dornseif, "Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks," *Proc. Int'l Conf. Security in Pervasive Computing (SPC '06),* pp. 104-118, 2006.

[4] C. Bettstetter, "On the Minimum Node Degree and Connectivity of a Wireless Multihop Network," *Proc. MobiHoc '02,* pp. 80-91, 2002.

[5] C. Bettstetter and C. Hartmann, "Connectivity of Wireless Multihop Networks in a Shadow Fading Environment," *Proc. Int'l Workshop Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM '03),* pp. 28-32, 2003.

[6] R. Brooks, P. Govindaraju, M. Pirretti, N. Vijaykrishnan, and M.T. Kandemir, "On the Detection of Clones in Sensor Networks Using Random Key Predistribution," *IEEE Trans. Systems, Man and Cybernetics, Part C: Applications and Rev.,* vol. 37, no. 6, pp. 1246-1258, Nov. 2007.

[7] S. Capkun and J.-P. Hubaux, "Secure Positioning of Wireless Devices with Application to Sensor Networks," *Proc. IEEE INFOCOM '05,* pp. 1917-1928, 2005.

[8] A. Caruso, A. Urpi, S. Chessa, and S. De, "Gps-Free Coordinate Assignment and Routing in Wireless Sensor Networks," *Proc. IEEE INFOCOM '05,* pp. 150-160, 2005.

[9] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," *Proc. Symp. Security and Privacy (S&P '03),* pp. 197-213, 2003.

[10] G. Chen, J.W. Branch, and B.K. Szymanski, "Local Leader Election, Signal Strength Aware Flooding, and Routeless Routing," *Proc. IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS '05),* p. 244.1, 2005.

[11] H. Choi, S. Zhu, and T.F. La Porta, "SET: Detecting Node Clones in Sensor Networks," *Proc. Int'l Conf. Security and Privacy in Comm. Networks and the Workshops (SecureComm '07),* pp. 341-350, 2007.

[12] C. Cocks, "An Identity Based Encryption Scheme Based on Quadratic Residues," *Proc. IMA Int'l Conf. '01,* pp. 360-363, 2001.

[13] 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," *Proc. Seventh Int'l Conf. Wired/Wireless Internet Comm. (WWIC '09),* pp. 85-96, 2009.

[14] M. Conti, R. Di Pietro, and L.V. Mancini, "Secure Cooperative Channel Establishment in Wireless Sensor Networks," *Proc. IEEE Pervasive Computing and Comm. (PERCOM '06) Workshop,* pp. 327-331, 2006.

[15] M. Conti, R. Di Pietro, and L.V. Mancini, "ECCE: Enhanced Cooperative Channel Establishment for Secure Pair-Wise Communication in Wireless Sensor Networks," *Ad Hoc Networks,* vol. 5, no. 1, pp. 49-62, 2007.

[16] M. Conti, R. Di Pietro, L.V. Mancini, and A. Mei, "Mobility and Cooperation to Thwart Node Capture Attacks in Manets," *J. Wireless Comm. and Networking.* Feb. 2009.

[17] M. Conti, R. Di Pietro, L.V. Mancini, and A. Mei, "Requirements and Open Issues in Distributed Detection of Node Identity Replicas in WSN," *Proc. IEEE Int'l Conf. Systems, Man and Cybernetics (SMC '06),* pp. 1468-1473, 2006.

[18] M. Conti, R. Di Pietro, L.V. Mancini, and A. Mei, "A Randomized, Efficient, and Distributed Protocol for the Detection of Node Replication Attacks in Wireless Sensor Networks," *Proc. MobiHoc '07,* pp. 80-89, 2007.

[19] M. Conti, R. Di Pietro, L.V. Mancini, and A. Mei, "Emergent Properties: Detection of the Node-Capture Attack in Mobile Wireless Sensor Networks," *Proc. ACM Conf. Wireless Netwok Security (WiSec '08),* pp. 214-219, 2008.

[20] B. Deb, S. Bhatnagar, and B. Nath, "ReInForM: Reliable Information Forwarding Using Multiple Paths in Sensor Networks," *Proc. IEEE Int'l Conf. Local Computer Networks (LCN '03),* pp. 406-415, 2003.

[21] M. Demirbas and Y. Song, "An RSSI-Based Scheme for Sybil Attack Detection in Wireless Sensor Networks," *Proc. Int'l Symp. World of Wireless, Mobile and Multimedia Networks (WOWMOM '06),* pp. 564-570, 2006.

[22] A. Derhab and N. Badache, "A Self-Stabilizing Leader Election Algorithm in Highly Dynamic Ad Hoc Mobile Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 19, no. 7, pp. 926-939, July 2008.

[23] R. Di Pietro, D. Ma, C. Soriente, and G. Tsudik, "Posh: Proactive Co-Operative Self-Healing in Unattended Wireless Sensor Networks," *Proc. IEEE Symp. Reliable Distributed Systems (SRDS),* pp. 185-194, 2008.

[24] R. Di Pietro and L.V. Mancini, "Intrusion Detection Systems," *Advances in Information Security,* vol. 38, Springer, 2008.

[25] R. Di Pietro, L.V. Mancini, and A. Mei, "Energy Efficient Node-to-Node Authentication and Communication Confidentiality in Wireless Sensor Networks," *Wireless Networks,* vol. 12, no. 6, pp. 709-721, 2006.

[26] R. Di Pietro, L.V. Mancini, A. Mei, A. Panconesi, and J. Radhakrishnan, "Connectivity Properties of Secure Wireless Sensor Networks," *Proc. Workshop Security of Ad Hoc and Sensor Networks (SASN '04),* pp. 53-58, 2004.

[27] R. Di Pietro, L.V. Mancini, A. Mei, A. Panconesi, and J. Radhakrishnan, "Sensor Networks That Are Provably Resilient," *Proc. Int'l Conf. Security and Privacy in Comm. Networks and the Workshops (SecureComm '06),* pp. 1-10, 2006.

[28] 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.

[29] J.R. Douceur, "The Sybil Attack," *Proc. Int'l Workshop Peer-to-Peer Systems (IPTPS '01),* pp. 251-260, 2002.

[30] D. Dubhashi, O. Häggström, L. Orecchia, A. Panconesi, C. Petrioli, and A. Vitaletti, "Localized Techniques for Broadcasting in Wireless Sensor Networks," *Algorithmica,* vol. 49, no. 4, pp. 412-446, 2007.

[31] J. Elson and D. Estrin, "Time Synchronization for Wireless Sensor Networks," *Proc. Int'l Parallel and Distributed Processing Symp. (IPDPS '01),* pp. 1965-1970, 2001.

[32] J. Elson, L. Girod, and D. Estrin, "Fine-Grained Network Time Synchronization Using Reference Broadcasts," *SIGOPS Operating Systems Rev.,* vol. 36, pp. 147-163, 2002.

[33] L. Eschenauer and V.D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," *Proc. Conf. Computer and Comm. Security (CCS '02),* pp. 41-47, 2002.

[34] F. Fu, J. Liu, and X. Yin, "Space-Time Related Pairwise Key Predistribution Scheme for Wireless Sensor Networks," *Proc. Int'l Conf. Wireless Comm., Networking and Mobile Computing (WiCom '07),* pp. 2692-2696, 2007.
[35] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks," *SIGMOBILE Mobile Computing and Comm. Rev.*, vol. 5, no. 4, pp. 11-25, 2001.
[36] V.D. Gligor, "Emergent Properties in Ad-Hoc Networks: A Security Perspective," *Proc. ACM Symp. Information, Computer and Comm. Security (ASIACCS '06),* p. 1, 2006.
[37] Y.C. Hu, A. Perrig, and D.B. Johnson, "Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks," *Proc. IEEE INFOCOM '03,* pp. 1976-1986, 2003.
[38] C. Karlof and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures," *Ad Hoc Networks,* vol. 1, nos. 2/3, pp. 293-315, 2003.
[39] B. Karp and H.T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," *Proc. ACM MobiCom '00,* pp. 243-254, 2000.
[40] J. Kong, H. Luo, K. Xu, D.L. Gu, M. Gerla, and S. Lu, "Adaptive Security for Multi-Layer Ad-Hoc Networks," *Wireless Communication and Mobile Computing,* vol. 2, no. 5, pp. 533-547, Wiley Interscience Press, 2002.
[41] S. Kwon and N.B. Shroff, "Paradox of Shortest Path Routing for Large Multi-Hop Wireless Networks," *Proc. IEEE INFOCOM '07,* pp. 1001-1009, 2007.
[42] A. Mei and J. Stefa, "Routing in Outer Space: Fair Traffic Load in Multi-Hop Wireless Networks," *Proc. MobiHoc '08,* pp. 23-32, 2008.
[43] J. Newsome, E. Shi, D. Song, and A. Perrig, "The Sybil Attack in Sensor Networks: Analysis & Defenses," *Proc. Int'l Symp. Information Processing in Sensor Networks (IPSN '04),* pp. 259-268, 2004.
[44] J. Newsome and D.X. Song, "Gem: Graph Embedding for Routing and Data-Centric Storage in Sensor Networks without Geographic Information," *Proc. Conf. Embedded Networked Sensor Systems (SenSys '03),* pp. 76-88, 2003.
[45] B. Parno, A. Perrig, and V.D. Gligor, "Distributed Detection of Node Replication Attacks in Sensor Networks," *Proc. IEEE Symp. Security and Privacy (S&P '05),* pp. 49-63, 2005.
[46] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes," *Proc. Advances in Cryptology (CRYPTO '84),* pp. 47-53, 1985.
[47] H. Song, L. Xie, S. Zhu, and G. Cao, "Sensor Node Compromise Detection: The Location Perspective," *Proc. Int'l Conf. Wireless Comm. and Mobile Computing (IWCMC '07),* pp. 242-247, 2007.
[48] S. Vasudevan, B. DeCleene, N. Immerman, J. Kurose, and D. Towsley, "Leader Election Algorithms for Wireless Ad Hoc Networks," *Proc. DARPA Information Survivability Conf. and Exposition (DISCEX '03),* pp. 261-272, 2003.
[49] A. Wander, N. Gura, H. Eberle, V. Gupta, and S.C. Shantz, "Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Pervasive Computing and Comm. (PERCOM '05),* pp. 324-328, 2005.
[50] Y. Yang, X. Wang, S. Zhu, and G. Cao, "SDAP: A Secure Hop-by-Hop Data Aggregation Protocol for Sensor Networks," *Proc. MobiHoc '06,* pp. 356-367, 2006.
[51] Q. Zhang, T. Yu, and P. Ning, "A Framework for Identifying Compromised Nodes in Wireless Sensor Networks," *ACM Trans. Information and System Security,* vol. 11, no. 3, pp. 1-37, 2008.
[52] B. Zhu, V.G.K. Addada, S. Setia, S. Jajodia, and S. Roy, "Efficient Distributed Detection of Node Replication Attacks in Sensor Networks," *Proc. Ann. Computer Security Applications Conf. (ACSAC '07),* pp. 257-266, 2007.
[53] S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks," *Proc. Conf. Computer and Comm. Security (CCS '03),* pp. 62-72, 2003.

Mauro Conti received the laurea degree (equivalent to MSc) and the PhD degree both in computer science from the University of Rome "La Sapienza," Italy, in 2005 and 2009, respectively. Since 2008, he has been a visiting scholar at the Center for Secure Information Systems (CSIS), George Mason University, Fairfax, Virginia. Since 2009, he has been a postdoctoral researcher at the Vrije Universiteit Amsterdam, The Netherlands. His current research interest is on security and privacy for wireless-resource-constrained mobile devices (sensors, RFIDs, and mobile phones). He is a member of the ACM and the IEEE.

Roberto Di Pietro received the MSc degree in computer science from the University of Pisa, Italy, in 1994, and the PhD degree in computer science in 2004 from the Universit di Roma "La Sapienza," Italy, where he received the specialization diploma in operating research and strategic decisions from the Department of Statistics in 2004. He is currently an assistant professor in the Department of Mathematics, Universit di Roma Tre—Roma, Italy. He has been a visiting scholar at the UNESCO Chair in Data Privacy (URV), Institut EURECOM, and George Mason University-CSIS. His main research interests include security for mobile, ad hoc, and underwater wireless networks; security for distributed systems; access control (role mining); virtualization and cloud security; computer forensics; and applied cryptography.

Luigi Vincenzo Mancini received the laurea degree in computer science from the University of Pisa, Italy, in 1983, and the PhD degree in computer science from the University of Newcastle upon Tyne, United Kingdom, in 1989. Since 2000, he has been a full professor of computer science in the Dipartimento di Informatica, University of Rome "La Sapienza." His current research interests include computer network and information security, secure multicast communication, public key infrastructure, authentication protocols, system survivability, computer privacy, wireless network security, fault-tolerant distributed systems, and large-scale peer-to-peer systems. He has published more than 80 scientific papers in international conferences and journals such as *ACM TISSEC, IEEE TC, IEEE TKDE, IEEE TPDS,* and *IEEE TSE*. He is the founder of Information and Communication Security (ICSecurity) Laboratory, and currently the director of the master degree programs in information and network security at the University of Rome "La Sapienza."

Alessandro Mei received the laurea degree in computer science with highest honors from the University of Pisa, Italy, in 1994, and the PhD degree in mathematics from the University of Trento, Italy, in 1999. He was a visiting scholar in the Department of EE-Systems at the University of Southern California during 1998 and part of 1999. After holding a postdoctoral position at the same university for one year, he joined the faculty of the Computer Science Department at Sapienza University of Rome, Italy. His main research interests include computer system security and parallel, distributed, and networked systems. He was the recipient of the Best Paper Award of the 16th IEEE International Parallel and Distributed Processing Symposium in 2002, the EE-Systems Outstanding Research Paper Award of the University of Southern California in 2000, and the Outstanding Paper Award of the Fifth IEEE/ACM International Conference on High Performance Computing in 1998. He is a past associate editor of the *IEEE Transactions on Computers* (2005-2009), and the general chair of IEEE IPDPS 2009, Rome, Italy. He is a member of the ACM and the IEEE.