International Journal of Modern Electronics and Communication Engineering (IJMECE) ISSN: 2321-2152
Volume No.-6, Issue No.-6, November, 2018

Hybrid-RAndomWaLk(H-RAWL) to Detect Clone Attacks on Wireless Sensor Networks

M.Thirunavukkarasan*[1]
[1]Research Scholar
Dept. of Computer Science and Engineering,
ManonmaniamSundaranar University, Tamil Nadu, India[1],
E-mail: thirujpccse@gmail.com

Dr. S.A. Sahaaya Arul Mary[2]
[2]Professor & Head,
Dept. of Computer Science and Engineering,
Saranathan College of Engineering, Tamil Nadu, India
E-mail: mary-cse@saranathan.ac.in

Abstract: Wireless sensor networks are helpless against the hub clone, and a few circulated conventions have been proposed to recognize this assault. Be that as it may, they require excessively solid suspicions, making it impossible to be handy for expansive scale, arbitrarily conveyed sensor systems. In this paper, we propose two novel hub clone identification conventions with various tradeoffs on organize conditions and execution. We demonstrate the dependability and versatility of Hybrid arbitrary stroll to Detecting Clone Attacks on the Node Clone Detection by investigating the likelihood that a foe may successfully discourage the set activities. Proposed in the writing for the location of these clones from which witness hub based circulated arrangements have indicated palatable outcomes. Cross breed Random Walk (H-RAWL) is one of the witness hub based disseminated systems in which witness hubs are arbitrarily chosen by starting a few irregular strolls all through the system. In spite of the fact that H-RAWL has accomplished high security of witness hubs however in achieving high identification likelihood H-RAWL experiences high correspondence and memory overhead. In this paper we propose a novel improvement in H-RAWL convention meaning to diminish the correspondence and memory costs while keeping the location likelihood high. Our reproduction comes about demonstrate that this change in H-RAWL diminishes the correspondence and memory costs as well as guarantees high security of witness hub. Execution investigation and reenactments additionally exhibit that the proposed plot is more effective than existing plans from both correspondence and memory cost outlooks.

Keywords: Wireless Sensor Networks; Security; Clone or Replica Detection; Random Walks, Constrained Random Walk, Hybrid Random Walk.

I. INTRODUCTION

Remote Sensor Network (WSN) is a gathering of sensor hubs with detecting abilities, restricted assets and propelled arrange designs with a wide assortment of uses [1-2]. WSNs are regularly sent in brutal, antagonistic and unattended conditions. These sensor hubs need alter obstruction equipment and they are inclined to numerous assaults. Here, we especially center on more destructive assault which is known as clone join or hub replication assault. In this assault an enemy physically catches at least one sensor hubs and trade off the entirety of its mystery certifications. He/she at that point makes imitations of the bargained hubs and secretly sends them at vital places of the system. A foe can use these clones to dispatch numerous insider assaults and pernicious exercises like he can dispatch a dark opening, wormhole assault or lunch particular sending assault and D7oS assault, infuse false information, screen and catch huge bit of movement, slander and outrage other end true blue hubs [3-4]. Outfitting the sensor hubs with an alter safe equipment is a basic answer for manage clone hub assaults, however this arrangement isn't engaging due to two principle reasons; in the first place, it is the cost, as it exceptionally costly to shield each sensor hub in the system with a carefully designed equipment, and second, a specialist assailant can at present sidestep alter obstruction. In this manner, there is a need to create programming based countermeasures for the recognition of clone hubs as all at present accessible conventions for verification and secure correspondence enable them to be a piece of system [5-8]. In the writing, two kinds of programming based arrangements have been proposed for the location of hub replication assault in static WSNs in particular brought together and appropriated.

In brought together arrangements the recognition procedure depends on a base station [9-10] or helped focal expert (i.e. base station, group head and so forth) [14-15]. In dispersed arrangements the location procedure is done by all sensor hubs in the system without the association of any focal expert. Some circulated approaches proposed to identify clone assaults [12-13, 16] have utilized claimer-correspondent witness structure (likewise called witness hub-based procedures) in which the claimer hub locally communicates its area claim to its neighbors and each neighbor fills in as a journalist hub whose obligation is to outline claimer id to at least one witness hubs. The strategy of witness hub based procedures is appeared in Fig. 1. The unified arrangements have accomplished high clone identification rates yet they all experience the ill effects of single purpose of disappointment and high correspondence costs. Because of these weaknesses the consideration of analysts is redirected towards conveyed arrangements. The principle issue with the current witness hub based methodologies is the determination and circulation of witness hubs i.e. either the witness hub choice is deterministic or the appropriation of witness hubs over the system is non-uniform (for every emphasis of the convention).

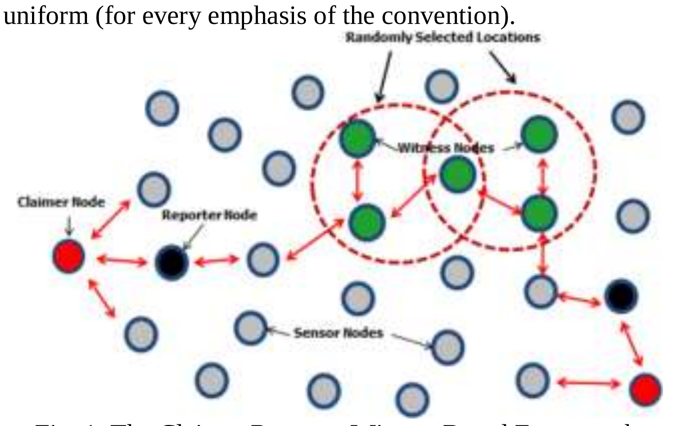

Fig. 1. The Claimer Reporter Witness Based Framework.

Our second protocol, named randomly directed exploration, is intended to provide highly efficient communication performance with adequate detection probability for dense sensor networks. In the protocol, initially nodes send claiming messages containing a neighbor-list along with a maximum hop limit to randomly selected neighbors; then, the subsequent message transmission is regulated by a probabilistic directed technique to approximately maintain a line property through the network as well as to incur sufficient randomness for better performance on communication and resilience against adversary. In addition, border determination mechanisms employed to further reduce communication payload. During forwarding, intermediate nodes explore claiming messages for node clone detection. By design, this protocol consumes almost minimal memory, and the simulations show that it out performs all other detection protocols in terms of communication cost, while the detection probability is satisfactory. Whatever is left of the paper is composed as takes after. To begin with, the past countermeasures are talked about in Section II. At that point, we display framework starters in Section III. A short time later, we expound the DHT-based recognition convention, break down its execution, and give the recreation brings about Sections IV– VI, separately. The arbitrarily coordinated investigation convention is point by point in Section VII, and its strong reproduction comes about are tended to in Section VIII. At long last, we finish up our work in Section IX.

II. Related Work

Goals

For a given sensor organize, we might want to identify a hub replication assault, i.e., an endeavor by the enemy to add at least one hubs to the system that utilization an indistinguishable ID from another hub in the system. In a perfect world, we might want to identify this conduct without concentrated observing, since brought together arrangements experience the ill effects of a few inalienable disadvantages (see Section 3.1). The plan ought to likewise renounce the duplicated hubs, so non broken hubs in the system stop to speak with any hubs infused in this form. We assess every convention's security by inspecting the likelihood of distinguishing an assault given that the enemy embeds L imitations of a subverted hub. The convention must give strong discovery regardless of whether the enemy catches extra hubs. We likewise assess the efficiency of every convention. In a sensor organize, correspondence (both sending and accepting) requires no less than a request of extent more power than some other activity [14], so our first need must limit correspondence, both for the system all in all and for the individual hubs (since hotspots will rapidly debilitate a hub's capacity supply). Also, sensor hubs normally have a constrained measure of memory, frequently on the request of a couple of kilobytes [14]. Consequently, any convention requiring a lot of memory will be unrealistic.

Sensor Network Environments

A sensor network typically consists of hundreds, or even thousands, of small, low-cost nodes distributed over a wide area. The nodes are expected to function in an unsupervised fashion even if new nodes are added, or old nodes disappear (e.g., due to power loss or accidental damage). While some networks include a central location for data collection, many operate in an entirely distributed manner, allowing the operators to retrieve aggregated data from any of the nodes in the network. Furthermore, data collection may only occur at irregular intervals. For example, many military applications strive to avoid any centralized and fixed points of failure. Instead, data is collected by mobile units (e.g., unmanned aerial units, foot soldiers, etc.) that access the sensor network at unpredictable locations and utilize the first sensor node they encounter as a conduit for the information accumulated by the network. Since these networks often operate in an unsupervised fashion for long period of time, we would like to detect a node replication attack soon after it occurs. If we wait until the next data collection cycle, the adversary has time to use its presence in the network to corrupt data, decommission legitimate nodes, or otherwise subvert the network's intended purpose. We also assume that the adversary cannot readily create new IDs for nodes. Newsome et al. describe several techniques to prevent the adversary from deploying nodes with arbitrary IDs [27]. For example, we can tie each node's ID to the unique knowledge it possesses. If the network uses a key pre-distribution scheme [6, 13], then a node's ID could correspond to the set of secret keys it shares with its neighbors (e.g., a node's ID is given by the hash of its secret keys). In this system, an adversary gains little advantage by claiming to possess an ID without actually holding the appropriate keys. Assuming the sensor network implements this safeguard, an adversary cannot create a new ID without guessing the appropriate keys (for most systems, this is infeasible), so instead the adversary must capture and clone a legitimate node.

Adversary Model

In analyzing the security of a sensor arrange, we adopt a preservationist strategy by accepting that the foe can clandestinely catch a predetermined number of honest to goodness sensor hubs. We restrict the level of hubs caught, since a foe that can catch most or the majority of the hubs in the system can clearly subvert any convention running in the system. Having caught these hubs, the foe can utilize subjective assaults on the hubs to remove their private data. For instance, the foe may misuse the unshielded idea of the hubs to peruse their cryptographic data from memory. The enemy could then clone the hub by stacking the hub's cryptographic data onto different non specific sensor hubs. Since sensor systems are inalienably intended to encourage specially appointed arrangement, these clones would then be able to be effortlessly embedded into self-assertive areas inside the system, subject just to the imperative that each embedded hub shares no less than one key with a portion of its neighbors. We permit the greater part of the hubs under the foe's control to convey and team up, however we make the streamlining supposition that any cloned hub has no less than one true blue hub as a neighbor. In Section 6, we indicate how we can expel this supposition while holding security. We expect that the enemy works in a stealthy way, endeavoring to stay away from recognition, since discovery could trigger a mechanized convention to clear the system, utilizing a method, for example, SWATT [32] to evacuate bargained hubs, or draw human consideration and additionally mediation. In the accompanying discourse, we will likewise accept that hubs under the foe's control (both the subverted hubs also, their clones) keep on following the conventions depicted. This enables us to center around the points of interest of the conventions, however in Section 10, we will propose techniques for unwinding this suspicion. As depicted over, our enemy demonstrates varies from the Dolev-Yao foe [9] in a few regards. Customarily used to break down cryptographic conventions, the Dolev-Yao display enables the enemy to peruse and compose messages at any area inside the system. In any case, in our exchange, we limit the foe to peruse and compose messages utilizing just the hubs under its control. Then again, our model additionally enables the enemy to subvert and imitate existing hubs in a versatile way, capacities not accessible to the Dolev-Yao foe. These abilities enable the foe to alter both the system topology and the "trust" topology, since the arrangement of genuine hubs changes as the foe subverts hubs and supplements extra copies.

III. PRELIMINARY APPROACHES

We discuss two possible NDFD solutions which are easy to figure out. Although both of them have some flaws, we discuss them to provide background for our two primary protocols, RAWL and H-RAWL, introduced later in Sections VI and VII respectively. Similar to existing protocols like LSM [2] and RED [9], our four protocols all can be scheduled to run periodically. Then if an adversary deploys replicas after one round of clone detection, the replicas will be detected in the next round.

A. *Random Witness Selection*

In this approach, each node signs its location claim and broadcasts the claim to its neighbors, and then its neighbors flood the claim in the network. Each node stores the claim with probability $\frac{c}{n}$. If a node receives a claim conflicting with another claim in its memory (i.e., a collision), it revokes the corresponding node. The difference between this approach and the Node-To-Network Broadcasting approach [2] is that here the witness nodes of a given node are not the neighbors of that node but randomly scattered in the network.

We give a brief analysis to show the $\frac{c}{n}$ probability is enough for clone detection. Suppose two nodes a, b have the same ID. If P_{of} is the probability that a node fails to detect the collision and P_s is the probability that at least one node in the network detects the collision, we have $P_s = 1 - (P_{of})^n$. Next we

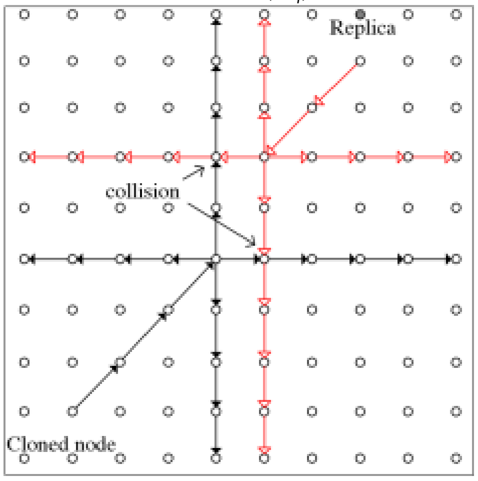

Fig. 2. The DRBased approach.

compute P_{of}. For any node, if we denote the probability that a's claim arrives at the node earlier than b's claim by ρ, then the probability that the node fails to detect the collision is equal to the probability that it does not store the claim of a:The communication and memory costs per node are $O(n)$ and $O(1)$ respectively. It is easy to see that this approach fulfills all the security requirements in Section III. However, the communication cost may be affordable in small networks, but it is too high for large networks.

A. *B. Double Ruling Based Detection*

The collision finding problem is similar to the read/write quorum problem in distributed file systems [15], and the storage/query problem in sensor networks [24]. All of them try to form two sets which share common elements (i.e., the witness node sets of the cloned node/the replica, the read/write quorums, and the storage nodes of a datum/the queried nodes by a user). Inspired by Double Ruling [24] for querying in sensor networks (the rectilinear case), we propose a DRBased approach. Similarly to LSM[2], every node broadcasts its location claim, and each of its neighbors, with probability p, forwards the claim to g random nodes. Then as shown in Fig.2, each of these random nodes starts to *broadcast* the claim in a horizontal line and a vertical line (forming a cross). There are total $r = p \cdot d \cdot g$ such crosses.Considering two nodes with the same ID, when at least one cross is formed for each node, a collision will always be detected. Both the communication and memory costs per node are $O(\sqrt{n})$. The costs are moderate; however, the protocol does not fulfill requirement 3. The attacker can learn the crosses, calculate the critical witness nodes, and disable them. Also it only works in rectangular deployment fields.

IV. RANDOM WALK BASED DETECTION (RAWL)

We may get an intuition from the DRBased approach that among all the requirements, requirement 3 may be mostdifficult to be fulfilled with moderate costs. However, in this section, we propose a new protocol RAWL based on random walk to fulfill the requirement, with only moderate costs.

A. *Protocol Description*

At a high level, RAWL works with following steps in each execution (recall that our four protocols all can be scheduled to run periodically). (1) Each node broadcasts a signed location claim. (2) Each of the node's neighbors probabilistically forwards the claim to some randomly selected nodes. (3) Each randomly selected node sends a message containing the claim to start a random walk in the network, and the passed nodes are selected as witness nodes and will store the claim. (4) If any witness receives different location claims for a same node ID, it can use these claims to revoke the replicated node. An example is shown in Fig.3.

We here describe the protocol more specifically. Each node a broadcasts a signed location claims to its neighbors. The claim has such a format: $\langle ID_a, l_a, \{H(ID_a \| l_a)\}_{K_a^{-1}} \rangle$, Where l_a is a's location (e.g., location (x,y) in 2D) and is the concatenation. When hearing the claim, each neighbor verifies the signature and checks the plausibility of l_a (e.g., the distance between two neighbors cannot be bigger than the transmission range). Then with probability p, each neighbor randomly selects g nodes (or g locations[1]) and uses geographic routing (e.g., GPSR [25]) to forward the claim to the g nodes (or nodes closest to the chosen g locations).

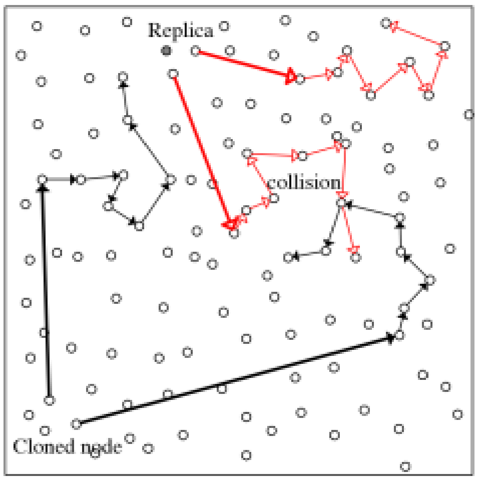

Fig. 3. The RAWL protocol.

Each chosen node that receives the claim of *a*, first verifies the signature. Then it stores the claim and becomes a witness node of *a*. Also, it will start a *t*-step random walk in the network (*t* is a system parameter, and we will analyze its value in Section VIII-A), by sending the location claim together with a counter of walked steps (s_c) initiated to 1, to a random neighbor. The neighbor will also become a witness node of *a*. It adds counter $s_c$by one and continues to forward the message to a random neighbor, unless counter $s_c$reaches *t*. When a node finds a collision (two different location claims with a same node ID), the node will broadcast the two conflicting claims as evidence to revoke the replicas. Each node receiving the two claims independently verifies the signatures. If the two signatures are valid, it terminates the links with replicas.

B. Security Analysis

The number of walk steps (*t*) is closely related to the detection ability of this protocol. Intuitively, the longer the random walks, the higher the probability that the random walks for replicas intersect. However, longer random walks will result in more communication and memory (storage) overloads. So determining the required number of walk steps is critical to the protocol. We will show in Section VIII-A that $O(\sqrt{n\log n})$ steps is sufficient for high detection probability.RAWL apparently does not have a central control in all the steps. Also, it is a non-deterministic protocol since the witness nodes of each node are different in each round. We further explain here that RAWL satisfies security requirements 1 and 4 (defined in Section III-B). For any given node, random walks are started from a random node, and each node in a torus (we model the network as a torus in Section VIII-A) has the same geographic property, so all the nodes have equal probability to be walked and become that node's witnesses. Also all the nodes obviously have equal probability to be witnesses if we consider the network as a torus. We will confirm that RAWL fulfills the two requirements in simulations.
RAWL satisfies security requirement 3, because the smart adversary cannot find out the critical witness nodes and move to disable them now. Firstly, we show that even sometimes a physical node may be selected more than once by a random walk; the number of physical nodes that are selected as witnesses is still on the same order of *t* in general settings. The detailed analysis is in Section VIII-B. Secondly, we discuss two possible cases for the adversary to find out the critical witness nodes. *In the first case*, the adversary can learn the next walked node when he has compromised the previous walked node (e.g., by finding packet history in memory). He still has to sequentially compromise all the following witness nodes from the starting node, to discover the critical witness nodes. Then the number of nodes needed to be compromised is on the same order of *t* (i.e., $O(\sqrt{n\log n})$) and is beyond the ability of the adversary.*In the second case*, the adversary cannot learn the next walked node when he has compromised the previous walked node (e.g., the packet history is erased). Then he will have to carry out a brute force attack by compromising all the neighbors around the current walked node. So the number of nodes needed to be compromised is more than the number in the first case, and is also beyond the adversary's ability. The resistance of RAWL can be intuitively explained by that it dispatches the witness-node-selection responsibility to every passed node of random walks, not only several nodes.

V. HYBRID RANDOM WALK BASED DETECTION (H-RAWL)

We want to find a method to reduce the memory cost of RAWL protocol, because sensor nodes usually have limited size of memory, e.g., on the order of a few kilobytes [26], which is also a precious resource. In this section, based on the previous protocol RAWL, we propose H-RAWL. Our basic idea is to employ a trace table at each node to record the traces (represented by "digests") of random walks.

A. Protocol Outline

Our new protocol is modified from RAWL. When a randomly chosen node starts a random walk, all the passed nodes will still become witness nodes. However, now they do not definitely store the location claim, instead, they store the location claim independently with probability $\frac{c_2}{\sqrt{n}\log n}$, where c_2 is a constant. Also, each witness node will create a new entry in its trace table (we will describe the table later) for recording the pass of a location claim. We describe the details on the trace table and the process of detection below.
We first describe a critical component, the trace table maintained by each node. Every entry of the table corresponds to the pass of a random walk (with a location claim). The table has the two columns: *NodeID*, *ClaimDigest*. The *NodeID*is the *ID* field of a claim (see Section VI-A). The *ClaimDigest*is a truncated message authentication code (MAC) of the whole location claim. An 8-bit *claimDigest*can be computed by $claimDigest = \{MAC_{rand}(Claim)\}_{mod(256)}$, (1) where *rand* is a random value generated by each node itself to prevent the adversary from generating a false claim with the same digest value, and $MAC_{rand}(^{Claim})$ is a message authentication code of location claim.Then we describe the process of detection. When receiving a location claim, a node will first find the entries which have the same node ID as the claim in its trace table. Then if any entry is found, the node will compute the digest of the claim using equation 1 and compare the digest with the digest in the entry. When the two digests are different, the node detects a clone attack. If the node stored the location claim of the entry, it will flood the network with the two location claims to revoke replicas. Otherwise it will flood a HELPREV request with only one location claim. Any node

receiving the HELPREV message will check locally that if it stored a location claim conflicting with the received one. If such a location claim is found, it will flood the stored location claim into the network as evidence. In such revocation process an algorithm for duplicate message suppression can be employed.

B. Security Analysis and Efficiency Analysis

H-RAWL has nearly the same detection ability as RAWL; only two issues will potentially degrade the detection ability. *The first issue* is that the *claimDigests*may be the same when two different location claims with a same node ID pass a witness node (i.e., *false negative*). However, such case occurs with low probability when we use a perfect MAC function (i.e., the *claimDigest*of a claim uniformly distributed within [0, 255]): the two *claimDigests*will be the same with probability only $\frac{1}{256}$. That means two different location claims with a same node ID will still result in collision with probability over 0.996. (We note here that using *claimDigest*does not lead to *false positive* detection. This is because when receiving a location claim, a witness node will compare the claim's *nodeID*in its trace table at first. Thus, even if two nodes' location claims passing the witness node have the same *claimDigest*, given that the two nodes have different node IDs, they will not be falsely detected as a clone attack.) *The second issue* is now the location claim of a random walk is not stored in all the nodes passed by the random walk. Similar to the analysis in Section V-A, we can deduce that there is a high probability that at least one witness stores the location claim. Considering a $c_1\sqrt{n}\log n$-step random walk, and in each step, node will store location claim with probability $\frac{c_2}{\sqrt{n\log n}}$ (both c_1 and c_2 are constant values), then the probability (P_{none}) that the location claim is not stored in all the steps is given by

$$P_{none} = \left(1-\frac{c_2}{\sqrt{n}\log n}\right)^{c_1\sqrt{n}\log n} \approx \frac{1}{e^{c_1c_2}}.$$

Thus, the probability (P_s) that at least one witness node stores the location claim is given by

$$P_s = 1 - P_{none} = 1 - \frac{1}{e^{c_1c_2}}.$$

We analyze the costs of H-RAWL. The communication cost of H-RAWL is apparently $\sqrt{n}\log n$, the same as RAWL. The memory cost of H-RAWL is smaller, because now most of the nodes in a random walk only store a table entry but not the location claim. They store the location claim independently with probability $\frac{c_2}{\sqrt{n\log n}}$, so the memory cost per node is

$O(c_1c_2 \cdot Claim + c_1\sqrt{n}\log n \cdot Entry)$. Here the size of a location claim is about 46 bytes: ID (2 bytes), location (4 bytes), signature (at least 40 bytes, e.g., ECDSA [20], [27]). However, the size of a table entry is just 3 bytes: nodeID (2 bytes), claimDigest (1 byte). Then theoretically H-RAWL reduces the memory cost of RAWL (whose memory cost is $c_1\sqrt{n}\log n\cdot Claim$) more than 10 times when $\sqrt{n}\log n \to \infty$.

VI. ANALYSIS

In this section, we analyze an important parameter (t) of our protocols, and the number of physical nodes selected as witnesses by our protocols (we used the analysis results here to support the security analysis in Section VI-B).

A. The Required Number of Walk Steps for Detecting Replicas

We would like to study the relation between the probability of detection and the number of walk steps t. First, we consider the simplest case that two replicas each have one random walk. Then we give formula on that L replicas each have r random walks. The first case is equal to the problem that two random walks (which start from the stationary distribution, i.e., start from each node with probability $1/n$ here) have at least one intersection. We assume that both the two random walks have t steps, $t \ll cn\log n$. We consider the network as a torus (i.e. a grid graph that is wrapped in both the north-south and the east-west directions) [28], [29] to simplify the analysis.

Next we will prove that $t$should be on the order of $O(\sqrt{n}\log n)$ for high detection probability. We notice that the result that $O(\sqrt{n}\log n)$ steps are sufficient for two random walks to collide in *fast mixing networks* is available (e.g., in [30], the proof is based on the general birthday paradox problem). However the torus is not fast mixing. Another closely related work is Rumor Routing [31] in sensor networks, where the authors used random walks for both the event distribution and query. However they found the needed number of walk steps only by simulations, without giving theoretic analysis on such number.

First, we consider the hitting time H_i for a node i. The hitting time H_i is the steps needed to hit the node i when a random walk starts from the stationary distribution (which is a distribution satisfying the balance equations [29]). The stationary distribution of a torus is $1/n$, so the random walk starts from all the nodes with the same probability $1/n$. Generally, in a torus the hitting time H_i for node i satisfies [28], [29]

$$P(H_i > x) \approx \exp\left(\frac{-x}{cn\log n}\right), \tag{2}$$

where c is a constant, and $O(n\log n)$ is the average hitting time of a torus [29]. Then the probability (P_h) that a t-step random walk (which starts from the stationary distribution) hits a node is given by

$$P_h = 1 - P(H_i > t). \tag{3}$$

For each node in the network, the probability that the two random walks have intersection at it is given by

$$P_{h2} = (P_h)^2 = (1 - P(H_i > t))^2. \tag{4}$$

Then the probability that the two random walks do not have any intersections in the network (i.e., at all nodes) is given by

$$P_{none} = (1 - P_{h2})^n. \tag{5}$$

If P_s is the probability that at least one collision is detected, then we have

$$P_s = 1 - P_{none}. \tag{6}$$

Combing the above equations, we have

$$\begin{aligned} P_s &= 1-(1-P_{h2})^n \\ &= 1-\left(1-(1-P(H_i>t))^2\right)^n \\ &= 1-\left(1-(1-e^{\frac{-t}{cn\log n}})^2\right)^n. \end{aligned} \quad (7)$$

Let $M=(1-e^{\frac{-t}{cn\log n}})^2$, then P_s can be written as

$$P_s = 1-(1-M)^n. \quad (8)$$

It is easy to see that $M \approx 0$ since we assumed $t \ll cn\log n$.

We know that the Binomial theorem allows us to approximate $(1-x)^y$ as $(1-xy)$ when x is small. So we have

$$P_s \approx 1-(1-M\cdot n) = M\cdot n = (1-e^{\frac{-t}{cn\log n}})^2 n. \quad (9)$$

Also since $t \ll cn\log n$ and $\frac{-t}{cn\log n}\approx 0$, then we can use the standard approximation $e^x \approx 1+x$. So we have

$$P_s \approx \left(1-(1+\frac{-t}{cn\log n})\right)^2 n = \frac{t^2}{c^2 n\log^2 n}. \quad (10)$$

Then if we want P_s to be a given value P, using equation 10, we can calculate that

$$t = c\sqrt{P}\sqrt{n\log n}. \quad (11)$$

From equation 11, we can see that for any given detection probability, the needed number of steps is on the order of $O(\sqrt{n\log n})$.

We can further calculate the detection probability (P_{sL}) when there are L replicas and for each node there are r random walks ($r = p\cdot d\cdot g$). Considering two replicas each with r random walks, if there is no collision between the two groups of random walks, then it means that any two random walks from them do not result in a collision. So the probability (P_{none2}) that two replicas do not result in a collision is given by

$$P_{none2} = (1-P_s)^{r^2}. \quad (12)$$

Then the probability (P_{s2}) that there is at least one collision with two replicas is given by

$$P_{s2} = 1-P_{none2} = 1-(1-P_s)^{r^2}. \quad (13)$$

Similarly, when there are L replicas and each with r random walks, if there is still no collision, then it means any combination of these L replicas does not have a collision. So the probability (P_{noneL}) that these replicas do not result in a collision is given by

$$P_{noneL} = (1-P_s)^{r^2\frac{L(L-1)}{2}}. \quad (14)$$

Then the probability (P_{sL}) that there is at least one collision with L replicas is given by

$$P_{sL} = 1-P_{noneL} = 1-(1-P_s)^{r^2\frac{L(L-1)}{2}}. \quad (15)$$

Our above results all are based on modeling the network as a torus (a d-regular graphs, $d = 4$), whose average hitting time is $O(n\log n)$ [29]. Another related graph is Hypercube (a $d$regular graph, $d = \log n$), whose average hitting time is $O(n)$ [29]. We can also consider the sensor network as a graph that lies between the two types of graphs as in [32], then following above analysis we can deduce that the required t is between $O(\sqrt{n})$ and $O(\sqrt{n\log n})$.

A. B. The Number of Physical Nodes Selected as Witness

We analyze how many physical nodes are selected by a $t$step random walk. As we mentioned in Section VI-B, since each node selects the next hop randomly, a t-step random walk may actually select only a small number of physic nodes.Next we show that in general settings (e.g., $d = 4, t \le 30$ and $d = 12, t \le 400$), the number of physical nodes selected by a t-step random walk as witnesses is no less than $t/2$, still on the same order of t. It is easy to see that the physical node of the starting node has the biggest walked times in a t-step random walk. If the walked times of this node is still less than 2, then the walked times of all the other visited physical nodes are also less than 2, and the number of selected physical nodes by a random walk must be no less than $t/2$. When analyzing the walked times of the starting node, we find it is hard to use general concepts in random walk, such as hitting time and commute time. Fortunately, we can transform the discrete-time random walk on an unweighted graph to be a Markov chain with transition matrix (P) [29] by

$$p_{vx} = \begin{cases} 1/d_v & \text{if } (v,x) \text{ is an edge} \\ 0 & \text{if not} \end{cases},$$

where d_v is the degree of vertex v. Then with an initial distribution μ, the distribution after i steps can be obtained by μP^i[33]. Thus we then can compute the walked times ofeach node (includes the starting node) in all the t steps by the formula:

$$D_t = \sum_{i=0}^{t-1}\mu P^i. \quad (16)$$

We assume the random walk starts from an arbitrary node a. Then initial distribution μ is set to $\mathbf{v}^T$, where $\mathbf{v}$ is a vector:

$$v_i = \begin{cases} 1 & \text{if } i = a \\ 0 & \text{if not} \end{cases}.$$

We apply the above method to a torus and a d-regular ($d = 12$) graph. Fig.4 shows in a torus the times the starting node are walked. We can see that when $t \le 30$ (later in Section IX-B, we show that $t = 18$ is enough when $r = 9$), the walked times of the starting node are less than 2. When the node degree of a network (d) increases, we can image that the random walk will go away from the starting node more quickly, so the walked times of the starting node will be smaller. Then we repeat the test in a d-regular ($d = 12$) graph, a denser network topology. We find that even $t = 400$, the walked times of the starting node is only 1.75, still less than 2. So according to previous discussions, we can conclude that a t-step random walk selects no less than $t/2$ physical nodes as witnesses in general network settings.

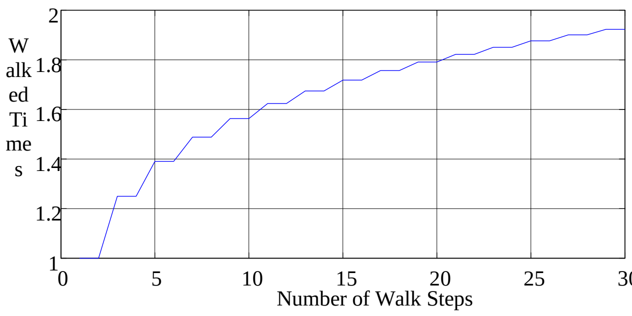

Fig. 4. The walked times of the starting node with different *t*s in a torus.

VII. SIMULATIONS

In this section we evaluate our protocols by simulations. We illustrate the witness distribution of our protocols, verify our theoretical analysis about the needed number of walk steps, and also study the communication and memory overheads. We will use LSM [2] for performance comparisons, since it is the existing NDFD protocol with the lowest communication and memory overheads.In our simulations, we randomly deploy 4000 nodes within a 1000m×1000msquare. The transmission range is set to 50m. Also we test our protocols in a variety of irregular network topologies in Section IX-B. We assume all the packets of replica-detection protocols can successfully reach their next hops; we assume occasional packet losses can be solved by retransmission mechanisms in lower layer protocols. Simulation result is the average of 20 executions if we do not state explicitly. The performance metrics used in our simulations are listed here:

- *Probability of detection (P_s).* Similar to the simulation in [2], we focus on a single node replication (one replica). We repeat the following process for given times (200 in our simulations): randomly insert a replica into the network and then start the detection protocol. Then we calculate P_s as $\frac{\#successful\ detection\ times}{\#repeat\ times}$.
- *Communication overhead*. We use the average number of messages each node broadcasts as a measure of communication overhead.
- *Memory overhead*. We use the average number of bytes each node stores as a measure of memory overhead.

A. *A. Witness Distribution*

We simulate two kinds of witness distributions here. The first one is the distribution of all nodes' witnesses in a round (for checking requirement 4 in Section III), and the second one is the distribution of one given node's witnesses in many rounds (for checking requirement 1). In fact, we check whether both kinds of witness distributions are uniform in the deployment area (which is also called area-obliviousness in [9])[2]. To show the distribution, the whole deployment area is divided into 50 × 50 grids. Then we record how many times the nodes in each grid are selected as witness nodes. Since RAWL and H-RAWL have exactly the same witness selection, we only present the witness distribution of RAWL.

Fig.7.1and Fig.7.2 report the witness distributions of RAWL and LSM in a round respectively. Here we execute both the two protocols one time in the same 10 randomly generated topologies. From Fig.7.1, we can see that the witness distribution of RAWL is nearly uniform in the divided grids, except the grids at the boundary. These grids suffer from the lower connectivity at the boundary (i.e., the boundary effect). Fig.7.2 shows that the witness distribution of LSM is not uniform, same as the observation by Conti et al. in [9]. The grids in the center area have much higher probabilities to accommodate witness nodes. From the boundary to the center, we can clearly see the number of witnesses is increasing. Some grids in the center even have more than ten times witnesses than grids at the boundary. So we can conclude that LSM does not fulfill requirement 4, but RAWL does if we ignore the boundary effect.

Fig.7.3 and Fig.7.4 show the witness distributions of RAWL and LSM for a randomly selected node respectively. We randomly generate a topology, and execute the two protocols for a randomly selected node 50000 times to get the average values. In our experiment, the randomly selected node lies at location (916,813). It is not surprise to see that the witness distribution of RAWL still is nearly uniform if we ignore the boundary effect, which indicates that RAWL fulfills requirement 1. On the other side, in LSM, grids near the grid the node lies in are more likely to accommodate witness nodes, because in LSM the paths to the random destinations always need to travel through these grids. Thus LSM fails to fulfill requirement 1.

B. *B. Probability of Detection*

In Section VIII-A, we have analyzed that the needed number of walk steps (t) is on the order of $O(\sqrt{n}\log n)$. Next we simulate our protocols to confirm that indeed not many stepsWe do not study the witness distribution in node ID space, because it is obvious that in our protocol (also other replica detection protocols like LSM) the witness distribution in the node ID space is uniform.

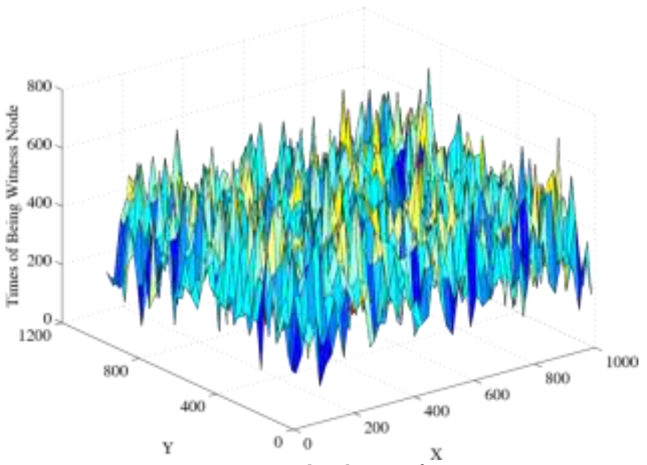

Fig.7.1 Witness distribution of RAWL.

are needed for replica detection. Our simulation in this section is carried out in 6 different network topologies, examples of which are shown in Fig.9. They represent both isotropic and anisotropic networks in real world deployments.

Fig.7.1 shows the probability of detection (P_s) values with different numbers of walk steps under different topologies. P_s is simulated using the method defined at the beginning of this section. We repeat the process for simulating P_s 20 times to get P_s's average value. Here the number of random walks for each location claim (r) is 9 (we will try other values later), and the number of walk steps (t) varies from 3 to 48. We set parameter c_2 of H-RAWL to 4. From Fig.10, first, we can see that H-RAWL has nearly the same probability of detection with RAWL (with less than 0.01 differences). We find that the small difference is mainly due to that two conflicting location claims have the same *claim Digest* value at intersection nodes. Second, the probability of detection grows rapidly with *t* in all the topologies. When the number of walk steps *t* is 18, P_ss of the six topologies all are greater than 0.95 (they are 0.95, 0.96, 0.96, 0.97, 0.98, and 0.97 respectively). So a relatively small *t* is indeed sufficient for detection.

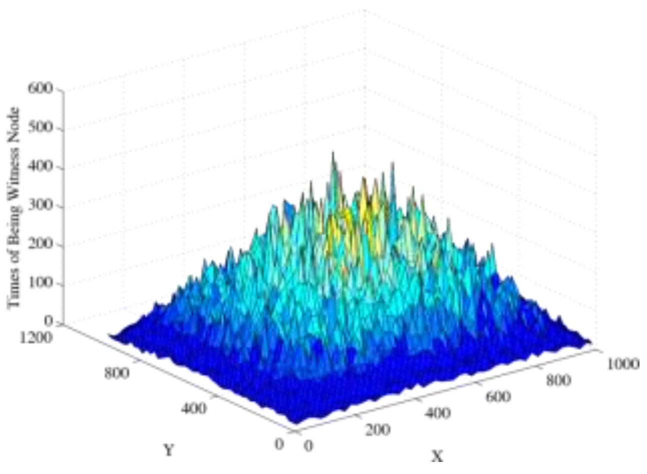

Fig. 7.2 Witness distribution of LSM.

Table 7.1 shows the probability of detection (P_s) values of RAWL with different numbers of random walks (*r*). We take the Standard topology as the simulation topology here.

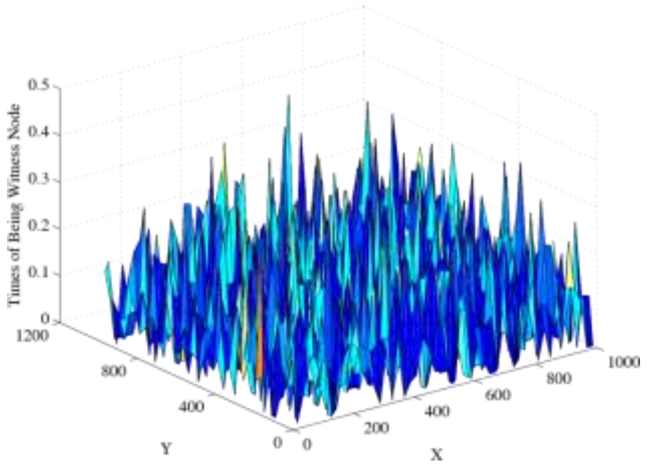

Fig. 7.3 Witness distribution of RAWL for a node at location (916,813).

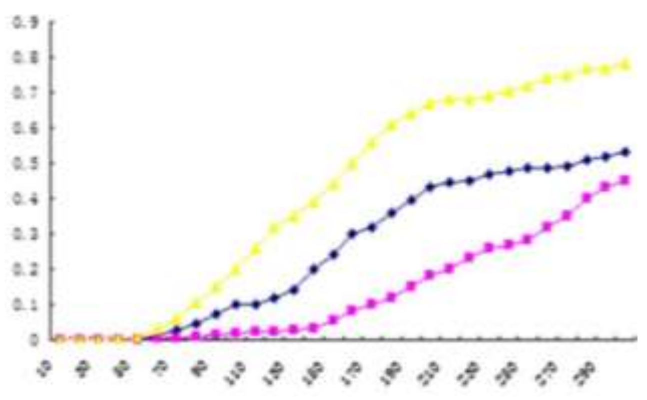

Fig. 7.4 Witness distribution of LSM for a node at location (916,813).

Several typical values of *r* are set, and the table only shows the P_ss around 0.95. We know that in RAWL one walk step corresponds to a witness node, and then the smaller number of walk steps, the less communication and memory overheads a random walk results in.

Fig.7.2 shows the memory overheads of different protocols. This experiment follows the same setting with the previous one. We assume the size of a location claim is 40 bytes, and the size of a trace table entry is 3 bytes. We can see that RAWL requires about 1.4 times of the memory of LSM for 0.95 probability of detection, while H-RAWL requires less than 1/2 of the memory of LSM for 0.95 probability of detection. DRBased protocol also uses less memory than LSM. The memory overheads can be easier to understand if we refer to Fig.14. The figure shows the average numbers of witness nodes of one node in different protocols. DRBased protocol has the least witness number so it is not surprising its memory overhead is low. H-RAWL has the same number of witness nodes with RAWL, however many witnesses only store small table entries now. So the memory overhead is lower than RAWL and LSM.

Table III. PROBABILITY OF DETECTION (P_s) VALUES WITH DIFFERENT NUMBERSOF RANDOM WALKS (*r*) AND DIFFERENT NUMBERS OF WALK STEPS (*t*) INTHE STANDARD TOPOLOGY

t	40	45	50	55	60	65	70
r=5	0.887	0.919	0.940	0.965	0.967	0.976	0.979
t	30	35	40	45	50	55	60
r=6	0.907	0.936	0.966	0.973	0.980	0.990	0.989
t	20	25	30	35	40	45	50
r=7	0.866	0.930	0.957	0.979	0.987	0.993	0.994
t	15	18	21	24	27	30	33
r=8	0.839	0.903	0.942	0.964	0.974	0.987	0.989
t	12	15	18	21	24	27	30
r=9	0.840	0.900	0.952	0.973	0.985	0.990	0.995
	12	15	18	21	24	27	30
r=10	0.890	0.948	0.976	0.988	0.996	0.995	0.997

In Fig 7.3 we draw the number of bytes each node stores in its memory, to evaluate the used memory distribution of nodes. For a fair comparison, we set the parameters of each protocol to ensure each P_s is about 0.95: $r = 9$,$t= 18$ in RAWL and H-RAWL, $r = 8.5$ in LSM, and $r = 0.95$ inDRBased. We repeat each protocol in the same 20 randomly generated Standard-shape topologies. We can see that LSM has a long tail in the distribution. It is the only protocol that has nodes using more than 20kB memory (about 0.93% of the total nodes). The used memory of RAWL is between1.1kB and 12.4kB. The used memory of H-RAWL is between 0.3kB and 3.7kB. Both of them seem to follow the normal distribution. In DRBased protocol, many nodes only need less than 2kB memory (about 60.3% of the total nodes), and the used memory is between 0kB and 15kB.

VIII. CONCLUSION

In this paper we designed several new replica-detection protocols. We found that existing solutions have several drawbacks which greatly limit their usages, and then we explained that to avoid the drawbacks, replica-detection protocols must be non-deterministic and fully distributed (NDFD), and fulfill three security requirements on witness selection. Previously, only one NDFD protocol, Randomized Multicast, fulfills the requirements; however it has very high communication overhead which is only affordable in small networks. Another NDFD protocol LSM has the lowest communication and memory overheads, but it does not fulfill the security requirements. Our final protocols, RAWL and H-RAWL, which are based on random walk, fulfill the requirements and have higher but comparable communication overhead than LSM. We believe they provide a better trade-off between the communication overhead and security properties than previous protocols. We also gave theoretical analysis on the required number of random walk steps. Finally, we note here that we think the mechanism H-RAWL used to reduce the memory overhead of RAWL (i.e., using a table to cache the digests of location claims), could also be applied to other protocols like LSM.

REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: a survey,” Computer Networks, vol. 38, no. 4, pp. 393–422, 2002.

[2] B. Parno, A. Perrig, and V. Gligor, “Distributed detection of node replication attacks in sensor networks,” in Proc. IEEE Symp. Security and Privacy (S&P ’05), 2005, pp. 49–63.

[3] R. Anderson and M. Kuhn, “Tamper resistance: a cautionary note,” in Proc. Second USENIX Workshop Electronic Commerce (WOEC ’96), 1996, pp. 1–11.

[4] A. Becher, Z. Benenson, and M. Dornseif, “Tampering with motes: Realworld physical attacks on wireless sensor networks,” in Proc. Third Int. Conf. Security Pervasive Computing (SPC ’06), 2006, pp. 104–118.

[5] F. Liu, X. Cheng, and D. Chen, “Insider attacker detection in wireless sensor networks,” in Proc. IEEE INFOCOM, May 2007, pp. 1937–1945.

[6] H. Chan, A. Perrig, and D. Song, “Random key predistribution schemes for sensor networks,” in Proc. IEEE Symp. Security and Privacy (S&P ’03), May 2003, pp. 197–213.

[7] S. Zhu, S. Setia, and S. Jajodia, “LEAP: efficient security mechanisms for large-scale distributed sensor networks,” in Proc. 10th ACM Conf. on Computer and Communications Security (CCS ’03), 2003, pp. 62–72.

[8] C. Karlof, N. Sastry, and D. Wagner, “TinySec: a link layer security architecture for wireless sensor networks,” in Proc. Second ACM Conf. Embedded Networked Sensor Systems (SenSys ’04), 2004, pp. 162–175.

[9] M. Conti, R. D. Pietro, L. V. Mancini, and A. Mei, “A randomized, efficient, and distributed protocol for the detection of node replication attacks in wireless sensor networks,” in Proc. ACM MobiHoc, 2007, pp. 80–89.

[10] B. Zhu, V. Addada, S. Setia, S. Jajodia, and S. Roy, “Efficient distributed detection of node replication attacks in sensor networks,” in Proc. 23rd Ann. Computer Security Applications Conference (ACSAC ’07), Dec. 2007, pp. 257–267.

[11] R. Brooks, P. Govindaraju, M. Pirretti, N. Vijaykrishnan, and M. Kandemir, “On the detection of clones in sensor networks using random key predistribution,” IEEE Trans. Syst., Man, Cybern. C, vol. 37, no. 6, pp. 1246–1258, Nov. 2007.

[12] H. Choi, S. Zhu, and T. F. La Porta, “SET: Detecting node clones in sensor networks,” in Proc. Third Int. Conf. Security and Privacy in Communications Networks and the Workshops (SecureComm ’07), Sept. 2007, pp. 341–350.

[13] K. Xing, F. Liu, X. Cheng, and D. H.-C. Du, “Realtime detection of clone attacks in wireless sensor networks,” in Proc. 28th Int. Conf. Distributed Computing Systems (ICDCS ’08), June 2008, pp. 3–10.

[14] C. A. Melchor, B. Ait-Salem, P. Gaborit, and K. Tamine, “Active detection of node replication attacks,” Int. J. of Computer Science and Network Security, vol. 9, no. 2, pp. 13–21, 2009.

[15] A. S. Tanenbaum and M. V. Steen, Distributed Systems: Principles and Paradigms. Prentice Hall, 2001.

[16] A. Savvides, C.-C. Han, and M. Srivastava, “Dynamic fine-grained localization in ad-hoc networks of sensors,” in Proc. ACM MobiCom, 2001, pp. 166–179.

[17] S. Capkun and J. P. Hubaux, “Secure positioning in wireless networks,”˜IEEE J. Sel. Areas Commun., vol. 24, no. 2, pp. 221–232, Feb. 2006.

[18] D. J. Malan, M. Welsh, and M. D. Smith, “Implementing public-key infrastructure for sensor networks,” ACM Trans. Sen. Netw., vol. 4, no. 4, pp. 1–23, 2008.

[19] H. Wang, B. Sheng, C. C. Tan, and Q. Li, “WM-ECC: an Elliptic Curve Cryptography Suite on Sensor Motes,” College of William and Mary, Computer Science, Williamsburg, VA, Tech. Rep. WM-CS-200711, 2007.

[20] A. Liu and P. Ning, “TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks,” in Proc. Seventh Int. Conf. Information Processing in Sensor Networks (IPSN ’08), 2008, pp. 245–256.

[21] J. Newsome, E. Shi, D. Song, and A. Perrig, “The sybil attack in sensor networks: analysis & defenses,” in Proc. Third Int. Symp. Information Processing in Sensor Networks (IPSN ’04), April 2004, pp. 259–268.

[22] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla, “SWATT: softwarebased attestation for embedded devices,” in Proc. IEEE Symp. Security and Privacy (S&P ’04), May 2004, pp. 272–282.

[23] W. Xu, W. Trappe, Y. Zhang, and T. Wood, “The feasibility of launching and detecting jamming attacks in wireless networks,” in Proc. ACM MobiHoc, 2005, pp. 46–57.

[24] R. Sarkar, X. Zhu, and J. Gao, “Double rulings for information brokerage in sensor networks,” in Proc. ACM MobiCom, 2006, pp. 286–297.

[25] B. Karp and H. T. Kung, “GPSR: greedy perimeter stateless routing for wireless networks,” in Proc. ACM MobiCom, 2000, pp. 243–254.

[26] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, “Next century challenges: scalable coordination in sensor networks,” in Proc. ACM MobiCom, 1999, pp. 263–270.

[27] P. Ning, A. Liu, and W. Du, “Mitigating dos attacks against broadcast authentication in wireless sensor networks,” ACM Trans. Sen. Netw., vol. 4, no. 1, pp. 1–35, 2008.

[28] R. C. Shah, S. Roy, S. Jain, and W. Brunette, “Data MULEs: modeling and analysis of a three-tier architecture for sparse sensor networks,” Ad Hoc Networks, vol. 1, no. 2-3, pp. 215 – 233, 2003.

[29] D. Aldous and J. A. Fill, Reversible Markov Chains and Random Walks on Graphs. Manuscript under preparation, 2001. [Online]. Available: http://stat-www.berkeley.edu/users/aldous/RWG/book.html

[30] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, “SybilGuard: defending against sybil attacks via social networks,” in Proc. SIGCOMM, 2006, pp. 267–278.

[31] D. Braginsky and D. Estrin, “Rumor routing algorthim for sensor networks,” in Proc. First ACM Int. Workshop on Wireless sensor networks and applications (WSNA ’02), 2002, pp. 22–31.

[32] C. Avin and C. Brito, “Efficient and robust query processing in dynamic environments using random walk techniques,” in Proc. Third Int. Symp. Information Processing in Sensor Networks (IPSN ’04), 2004, pp. 277–286.

[33] S. Meyn and R. Tweedie, Markov chains and stochastic stability. Springer-Verlag, 1993.

[34] M. Shao, Y. Yang, S. Zhu, and G. Cao, “Towards statistically strong source anonymity for sensor networks,” in Proc. IEEE INFOCOM, April 2008, pp. 51–55.

[35] Y. Yang, M. Shao, S. Zhu, B. Urgaonkar, and G. Cao, “Towards event source unobservability with minimum network traffic in sensor networks,” in Proc. First ACM Conf. Wireless Network Security (WiSec ’08), 2008, pp. 77–88.

[36] J. McCune, E. Shi, A. Perrig, and M. Reiter, “Detection of denial-ofmessage attacks on sensor network broadcasts,” in Proc. IEEE Symp. Security and Privacy (S&P ’05), May 2005, pp. 64–78.

[37] “IEEE std 1609.2, IEEE trial-use standard for wireless access in vehicular environments - security services for applications and management messages,” Intelligent Transportation Systems Committee, 2006.

[38] Y. W. Law, J. Doumen, and P. Hartel, “Survey and benchmark of block ciphers for wireless sensor networks,” ACM Trans. Sen. Netw., vol. 2, no. 1, pp. 65–93, 2006.