Random-Walk Based Approach to Detect Clone Attacks in Wireless Sensor Networks

Yingpei Zeng, Jiannong Cao, *Senior Member, IEEE,* Shigeng Zhang, Shanqing Guo and Li Xie

***Abstract*—Wireless sensor networks (WSNs) deployed in hostile environments are vulnerable to clone attacks. In such attack, an adversary compromises a few nodes, replicates them, and inserts arbitrary number of replicas into the network. Consequently, the adversary can carry out many internal attacks. Previous solutions on detecting clone attacks have several drawbacks. First, some of them require a central control, which introduces several inherent limits. Second, some of them are deterministic and vulnerable to simple witness compromising attacks. Third, in some solutions the adversary can easily learn the critical witness nodes to start *smart attacks* and protect replicas from being detected. In this paper, we first show that in order to avoid existing drawbacks, replica-detection protocols must be non-deterministic and fully distributed (NDFD), and fulfill three security requirements on witness selection. To our knowledge, only one existing protocol, Randomized Multicast, is NDFD and fulfills the requirements, but it has very high communication overhead. Then, based on random walk, we propose two new NDFD protocols, RAndom WaLk (RAWL) and Table-assisted RAndom WaLk (TRAWL), which fulfill the requirements while having only moderate communication and memory overheads. The random walk strategy outperforms previous strategies because it distributes a core step, the witness selection, to every passed node of random walks, and then the adversary cannot easily find out the critical witness nodes. We theoretically analyze the required number of walk steps for ensuring detection. Our simulation results show that our protocols outperform an existing NDFD protocol with the lowest overheads in witness selection, and TRAWL even has lower memory overhead than that protocol. The communication overheads of our protocols are higher but are affordable considering their security benefits.**

***Index Terms*—Wireless sensor networks, computer network security, clone attacks, node replication, random walk.**

I. Introduction

WIRELESS sensor networks (WSNs) have been used in various applications, e.g., military, environmental, and health applications [1]. When WSNs are deployed in hostile scenarios, such as surveillance on the battlefield, they must confront the threats from attackers (e.g., enemies on the battlefield). This is because the attackers may intend to learn information from the WSNs or disable the functions of the WSNs. For example, on the battlefield, the enemies would hope to learn the private locations of soldiers from, or inject wrong commands into the sensor network. So it is critical to ensure the security of sensor networks in such scenarios.

Clone attack [2] (also called node replication attack) is a severe attack in WSNs. In this attack, an adversary captures only a few of nodes, replicates them and then deploys arbitrary number of replicas throughout the network. The capture of nodes is plausible [3], [4] because sensor nodes are usually unprotected by physical shielding due to cost considerations [2], and are often left unattended after deployment. If we do not detect these replicas, the network will be vulnerable to a large class of internal attacks [5]. For example, the adversary now can overhear the traffic passing the replicas (which may contain the aforementioned locations of soldiers), inject false data into the network (which may be false commands), defame other nodes and even revoke legitimate nodes [2]. Existing authentication techniques [6], [7], [8] cannot detect such attacks, because all the replicas hold legitimate keys.

Various approaches have been proposed to detect clone attacks [2], [9], [10], [11], [12], [13]. They essentially try to detect the abnormal symptoms caused by replicas (e.g., a node ID is associated with two different locations). However, existing approaches may be *deterministic*, or cannot defend against *smart attacks*, or need a central control. Firstly, *deterministic* means that which nodes detect the abnormal symptoms of a given node is fixed (usually these nodes are called the *witness nodes* of the given node). In this case, if the adversary compromises all the witness nodes of a captured node, he can then safely deploy any number of replicas of that node. Secondly, the adversary may protect his replicas by starting a special witness compromising attack, the *smart attack*. In this attack the adversary finds out the witness nodes that will detect the replicas (we call them *critical witness nodes*), and only compromises these witness nodes to avoid detection. Finally, as pointed out in [2], the detection protocols involving a central control have inherent limits such as a single point of failure. Table I shows the classification of some existing approaches.

In this paper, firstly, we show that in order to avoid the drawbacks of existing approaches, replica-detection protocols must be NDFD and fulfill three security requirements on witness selection. To our knowledge, Randomized Multicast [2] is the only existing protocol fulfilling the requirements, but it has very high communication overhead (i.e., $O(n)$ per node). Secondly, based on random walk, we propose two new NDFD protocols fulfilling the requirements, while

Manuscript received 3 April 2009; revised 7 November 2009. This work is supported in part by Hong Kong RGC GRF under Grant PolyU5102/08E, Hong Kong PolyU under Grant 1-BB6c, China 973 Project under Grant 2009CB320702, the National Natural Science Foundation of China (NSFC) under Grant 60673154, and the Natural Science Foundation of Jiangsu Province under Grant BK2009465.

Y. Zeng, S. Zhang and L. Xie are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, P.R. China. (e-mail: {zyp,zsg}@dislab.nju.edu.cn, xieli@nju.edu.cn).

J. Cao is with the Department of Computing, Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong. (e-mail: csjcao@comp.polyu.edu.hk).

S. Guo is with the School of Computer Science and Technology, Shandong University, Jinan, P.R. China. (e-mail: guoshanqing@sdu.edu.cn).

Digital Object Identifier 10.1109/JSAC.2010.100606.

TABLE I
SECURITY AND COST COMPARISONS OF DIFFERENT PROTOCOLS

Protocols	Non-deterministic	Resilient to Smart Attack	Communication	Memory
Deterministic Multicast [2]	No	No	$O(\sqrt{n})$[1]	$O(1)$
Randomized Multicast [2]	Yes	Yes	$O(n)$	$O(\sqrt{n})$
LSM [2]	Yes	No	$O(\sqrt{n})$	$O(\sqrt{n})$
RED [9]	Yes	No	$O(\sqrt{n})$	$O(1)$
SDC [10]	No	Yes	$O(\sqrt{n})$	$O(1)$
RAWL	Yes	Yes	$O(\sqrt{n}\log n)$	$O(\sqrt{n}\log n)$
TRAWL	Yes	Yes	$O(\sqrt{n}\log n)$	$O(1)$ [2]

[1] n is the number of nodes in the network
[2] without counting the trace table

having only moderate communication and memory overheads. Our random walk strategy outperforms previous strategies because it naturally distributes the responsibility of witness node selection to every passed node of random walks, and then adversaries cannot easily find out the critical witness nodes. The first protocol, RAndom WaLk (RAWL), starts several random walks randomly in the network for each node a, and then selects the passed nodes as the witness nodes of node a. Our analysis shows that $O(\sqrt{n}\log n)$ walk steps are sufficient to detect clone attacks with high probability. The second protocol, Table-assisted RAndom WaLk (TRAWL), is based on RAWL and adds a trace table at each node to reduce memory cost. Usually the memory cost is due to the storage of location claims; in TRAWL each node only stores $O(1)$ location claims now (although the size of the trace table is still $O(\sqrt{n}\log n)$, the size of a table entry is much smaller than the size of a location claim). Our simulation results show that our protocols outperform the best existing NDFD protocol LSM [2] (having the lowest overheads) in witness selection, and TRAWL even has lower memory overhead than LSM. The communication overheads of our protocols are higher than LSM, but are affordable considering the security benefits.

The rest of the paper is organized as follows. In Section II we review related work. Then in Section III we discuss the design principles of replica-detection protocols. In Section IV, we describe the network and the adversary models. After discussing some preliminary approaches which address some of the existing drawbacks listed in Section V, we present and analyze our two primary protocols, RAWL and TRAWL, in Sections VI and VII respectively. Theoretical analysis, simulation results, and discussions of our protocols are given in Sections VIII, IX, and X respectively. Finally we conclude this paper in Section XI.

II. RELATED WORK

Some methods can prevent clone attacks, however only in certain circumstances. For example, we can bind each node's deployment location with its ID then replicas cannot be placed at other locations; however we need to know the location of each node in advance. We can also erase the primary key in each node after the node establishes pairwise keys with its neighbors [7]; however this method only works when the adversary does not launch clone attacks during such key establishment phase, and we will not deploy any new node.

We thus focus on the approaches applicable in general conditions. We roughly classify existing approaches into two categories: central control involved and fully distributed. Approaches in the first category usually need a central control, e.g., the base station (BS), in some critical steps. Approaches in the second category do not need any central control.

A. Central Control Involved Approaches

Finding conflicting location claims is a common method to detect replicas. In [2] Parno et al. presented a straightforward centralized detection approach. In this approach, each node sends a list of its neighbors and their location claims to the BS. Since there is a common assumption that the replicas cannot have different IDs (we will describe the reason later in Section IV-A), the BS can easily find and revoke nodes with the same ID but with different location claims. In [9] Conti et al. proposed RED, which is similar to Deterministic Multicast [2] (we will introduce it in next subsection), except that who are the witness nodes of a node is determined by a random value distributed by the BS in each execution of the protocol. In the above two protocols, the communication and memory costs per node are $O(\sqrt{n})$ and $O(1)$ respectively. Both protocols suffer from the inherent drawbacks we list later in Section III-A.

Finding other abnormalities is also used to detect replicas in the literature. Choi et al. [12] proposed SET to detect the abnormality that an ID appears in different exclusive subgroups. The network is partitioned into clusters based on a random value broadcasted by the BS, and trees are formed to check whether subgroups have IDs in common. SET may have false detections when insidious leaders in the trees forge IDs not in their clusters. Brook et al. [11] proposed to detect the abnormality that some keys are used too often in communications. The BS collects the times keys are used in each node, judges the abnormal keys and tells each node to terminate the links using these keys. This approach is designed for special key distribution schemes in [6]; it is not clear whether it works well with other schemes. Xing et al. [13] proposed to detect the abnormality that a node has different fingerprints. Fingerprint is generated from node's neighbor list, and the BS detects the replicas if it receives different fingerprints for the same ID. This scheme requires each node to periodically communicate with the BS.

B. Fully Distributed Approaches

All existing approaches in this category detect replicas based on finding conflicting location claims. In [2] Parno et al. proposed four approaches. The first one is Node-To-Network Broadcasting. When executing the protocol, each node broadcasts its location claim to the whole network, and all nodes store the location claims of their neighbors only. Then if a node receives two conflicting claims of some node ID, it can revoke that node by flooding the network with the two claims. The second approach is Deterministic Multicast. A fixed mapping function is used to map each node ID to g

nodes (witness nodes), and then each node's neighbors will forward the node's location claim to these g nodes.

Another two approaches proposed in [2] by Parno et al are Randomized Multicast and Line-Selected Multicast (LSM). In Randomized Multicast the neighbors of each node randomly select $\sqrt{n}$ nodes as that node's witnesses. Then if a node is replicated, according to the birthday paradox problem, at least one witness will receive two conflicting location claims with high probability. LSM improved on Randomized Multicast. In LSM the nodes in the paths from a node's neighbors to the randomly selected witnesses are used; these nodes become the node's witnesses too. Such change reduces the communication cost per node from $O(n)$ to $O(\sqrt{n})$.

Zhu et al. [10] divided the network into cells, and proposed two approaches: SDC and P-MPC. In SDC, each node ID is mapped to one cell, and the location claim of each node is forwarded to the mapped cell and broadcasted within the cell. Nodes in the cell store the claim and become that node's witnesses with some probability. P-MPC is different from SDC in that each node ID is mapped to multiple cells with different probabilities, however, the set of possible mapped cells is still deterministic.

Melchor et al. [14] proposed an active detection approach, in which witness nodes actively obtain location claims. Each node first randomly chooses several nodes and becomes their witness node. Then if a node is node a's witness node, it will send location-claim request through several relay nodes to node a. These relay nodes are randomly chosen by the witness node for a. Thus if a has a replica, the replica will have high probability to receive the request as well, and reply a conflicting location claim to the witness node.

Our protocols also belong to this category. We would like to delay the analysis of these approaches to the next section.

III. Design Principles

In this section we discuss why replica-detection protocols must be NDFD and fulfill the three security requirements identified here.

A. Protocol Type Selection

Central control involved vs. fully distributed. Usually, a central control (e.g., the BS) can reduce the complexity of protocols; however, centralized system has some inherent drawbacks compared with distributed system [15], and the specific drawbacks of BS-involved schemes in replica detection were identified by Parno et al. in [2], e.g., making the BS become a single point of failure, aggravating the energy depletion of nodes surrounding the BS, and failing to work in no-BS networks. We specially note here that the absent of a BS may be due not only to cost issues, but also to security issues. For example, a fully distributed network without BS usually is more suitable in the battlefield application; otherwise the BS will be an attractive target for the enemies.

Deterministic vs. non-deterministic. It is more difficult for an adversary to successfully launch clone attacks in non-deterministic protocols. In *deterministic* protocols, e.g., Deterministic Multicast [2] and SDC [10], the witnesses of a node are fixed in each execution of the protocol (detection protocols may be scheduled to run periodically to prevent an adversary from inserting replicas after protocol execution [2]). Note that here we consider the deterministic property between different executions of the protocol. In deterministic protocols, if an adversary captures and replicates a node, and compromises that node's witness nodes, he can then safely deploy any number of replicas. Repeating the execution of such detection protocols has no effect. We note here that the adversary can succeed because the number of witness nodes of each node is usually a relatively small value[1], then the adversary can keep on compromising them *during the whole lifetime of the network* and succeed. In contrast, in *non-deterministic* protocols [2], the witnesses of a node are different in each execution of the protocol; the adversary now cannot keep on compromising a node's witnesses by compromising a subset of nodes in the network.

However, if a NDFD protocol is not properly designed, it will still be vulnerable to witness compromising attacks. For example, if a subset of nodes are always more likely to be node a's witnesses, then the adversary can use this knowledge to compromise a's witnesses faster. We thus further analyze the requirements NDFD protocols need to fulfill to resist adversaries.

B. Security Requirement of Equally Being Witness of a Node

Suppose an adversary wants to compromise a given node's witnesses, then to prevent him from using any knowledge to assist his compromising, during the lifetime of a network all the nodes[2] should have the same probability to be the witnesses of that node. So the following requirement must be fulfilled:

Requirement 1: During the lifetime of the network, for any given node, all the nodes have equal probability to be its witness nodes.

An existing protocol that violates this requirement is LSM [2]. In LSM, nodes near a given node have higher probabilities to be passed by paths (lines) from the node and thus become the node's witnesses with higher probabilities. We will show this phenomenon by simulation in Section IX-A.

C. Security Requirement of Resistance to Smart Attacks

Since only a few witness nodes (i.e., *the critical witness nodes*) will actually detect the abnormality in each execution of detection protocol in the whole network (next we call one execution one *round*), an adversary may protect his replicas by starting *smart attack*, a special witness compromising attack. In this attack, a smart adversary finds out and disables (i.e., compromises or jams) only the critical witness nodes. Take LSM protocol [2] for example. Fig.1 shows that two nodes (i.e., the cloned node and the replica) have the same ID. If the adversary learns the destinations of paths[3], he calculates

[1]I.e., the number is far less than n, otherwise both the communication and memory costs of such protocol will be $O(n)$, which are only affordable in small networks.

[2]In this paper nodes are identified by their invariant attributes such as IDs and locations.

[3]When the network is protected by TinySec [8], the adversary can learn the paths because he can decrypt the overheard messages using the key obtained from the cloned node. Or he can just compromise the neighbors forwarding the claims.

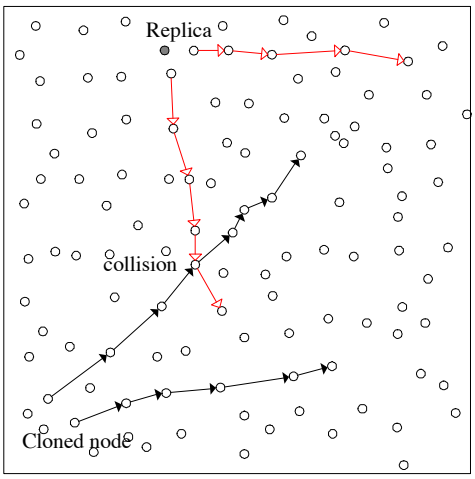


Fig. 1. In LSM [2] the adversary can protect the replica by jamming or compromising the intersection node.

whether the paths will intersect and guesses the intersection nodes. If he finds an intersection node (i.e., a critical witness node), he will directly move to there and disable the node. Then the replica will escape from detection.

Note that as we will show in Section IV-B, we make an assumption that the number of nodes (k) an adversary can disable for one replica is limited, i.e., $k \ll \sqrt{n}$.

To make an adversary unable to learn the critical witness nodes in each round, we can easily get an ideal requirement that replica-detection protocols should satisfy (recall that we call one execution of detection protocol one round):

Requirement 2 (Ideal): In each round, for any given node a, all the nodes have equal probability to be its witness nodes. Also, even if the adversary compromises a limited number of nodes, he cannot learn anything about the rest of the nodes which are or have higher probability to be a's witness nodes.

Requirement 2 is a strong requirement (if a protocol fulfills requirement 2, it is easy to deduce that the protocol also fulfills requirement 1). This ideal requirement is costly to fulfill (the protocol we will present in Section V-A currently is the only protocol that fulfills this requirement). Thus we relax the ideal requirement to the below practical one:

Requirement 3 (Practical): In each round, for any given node a, nodes may have different probabilities to be a's witness nodes. However the adversary cannot disable $O(k)$ nodes to disable the majority of a's critical witness nodes.

D. Security Requirement of Equally Being Witness

When the adversary wants to blindly disable *all the nodes' witnesses*, if some nodes are more likely to be witnesses, they will be chosen by the adversary as targets to disable witnesses more efficiently. Also, if a node serves as the witness of too many nodes, it will quickly use up its memory and energy and break down [9]. So we have such requirement:

Requirement 4: In each round, all the nodes have equal probability to be witness nodes.

Note that requirement 4 is different from requirement 1 because it is about the probability for *just being witness* (any node's witness) *in one round*. LSM [2] violates this requirement, because in each round, LSM tends to select the nodes in the central area as witness nodes. Conti et al. first noticed this phenomenon in [9]. Our simulation in Section IX-A also confirms their discovery.

E. Discussion

To avoid the known drawbacks, replica-detection protocols must be NDFD, and fulfill the three security requirements: 1, 3 (or 2), and 4. The fully distributed property guarantees that a protocol does not have the inherent limits caused by the central control. The non-deterministic property and three requirements guarantee that a protocol can resist the witness compromising attack (also its special type, the smart attack) discussed above.

Existing approaches are not satisfactory. The approaches in Section II-A all need a central control. Among the fully distributed approaches in Section II-B, both SDC [10] and Deterministic Multicast [2] are deterministic, active detection [14] violates requirement 3, and LSM [2] violates requirements 1, 3, and 4. Randomized Multicast [2] is NDFD protocol and satisfies all the three requirements (1, 3, and 4); however its communication and memory overheads are $O(n)$ and $O(\sqrt{n})$ respectively, which are too high for large networks.

IV. Network and Adversary Models

A. Network Model

We assume nodes uniformly distribute in the deployment field. We assume nodes know their own locations; many proposed localization algorithms [16], [17] can be used. We assume nodes are stationary, at least during the execution of replica-detection protocol. Each node a has a private key K_a^{-1} and can use the private key to sign its location claim. Other nodes are also able to verify the signature. Now several public-key libraries for sensor networks are available [18], [19], [20]. We also assume the communications between any two nodes are protected by pairwise keys. Same as previous works [2], [9], [10], [12], [13], we assume that the adversary cannot create new IDs for replicas. Some key management schemes already provide such property [6], [7], and other measures [21] can also be introduced into key management schemes to enforce such property (e.g., mapping ID to the indices of keys with a one-way function).

B. Adversary Model

The adversary can launch a clone attack: he compromises a few nodes [4], uses the cryptographic information obtained from the compromised nodes to produce replicas, and then inserts the replicas into the network. The compromised nodes and replicas are fully controlled by the adversary and can communicate with each other at any time. Also, same as previous protocols [2], [10], we assume nodes controlled by the adversary still follow the replica-detection protocol, since the adversary always wants to keep him unnoticed to others. We will further discuss this assumption in Section X.

The adversary will try to protect its replicas. This is because if any replicas are detected, besides starting a revoke process to revoke the replicas, the network may start a sweeping process to sweep the compromised nodes out [22] and may draw

TABLE II
NOTATION

n	Number of nodes in the network
c, c_1, c_2	Constant values
d	Average degree of each node
p	Probability a neighbor will forward location claim
g	Number of nodes selected by each neighbor
l_a	Location node a claims to occupy
K_a	a's public key
K_a^{-1}	a's private key
$\{M\}_{K_a^{-1}}$	a's signature on M
$H(M)$	Hash of M
$MAC_K(M)$	Message authentication code of M with key K
r	Number of random walks for each node
t	Number of walk steps

human attention. We assume during the execution of replica-detection protocol, the adversary can select a limited number of nodes to disable (i.e., compromise [4] or jam [23]) for protecting his replicas. He is able to do that because the time taken by the execution of protocol may be long enough (e.g., the delay caused by synchronization error, processing delay in each hop, and sleep schedule of the network). Also, since jamming a node is more quickly than compromising it, the adversary can jam a node first and compromise it later. We give a general assumption that the adversary can disable a small number of nodes k ($k \ll \sqrt{n}$) for protecting one replica. We make such assumption because $\sqrt{n}$ is usually the number of witnesses of one node [2].

C. Notation

For clarity, we list the notation used in this paper in Table II.

V. 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 TRAWL, 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. Broadcasting with 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 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: $1-\frac{c}{n}$. The case that the claim of b arrives first (with probability $1-\rho$) is similar. So we have $P_{of} = \rho(1-\frac{c}{n}) + (1-\rho)(1-\frac{c}{n}) = (1-\frac{c}{n})$. Combining the two equations and using the approximation $(1-\frac{1}{x})^x \approx \frac{1}{e}$, we have $P_s = 1-(1-\frac{c}{n})^n \approx 1-\frac{1}{e^c}$. So the probability $\frac{c}{n}$ is sufficient.

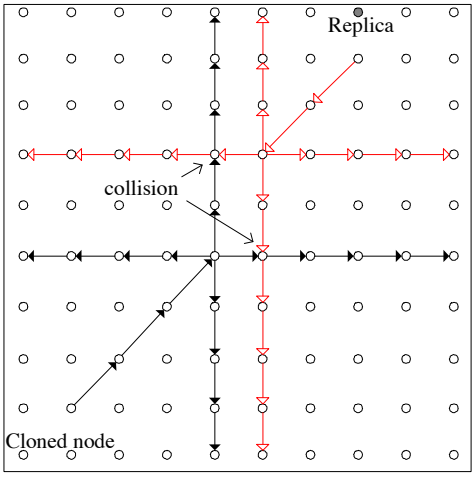


Fig. 2. The DRBased approach.

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.

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.

VI. Random Walk Based Detection (RAWL)

We may get an intuition from the DRBased approach that among all the requirements, requirement 3 may be most

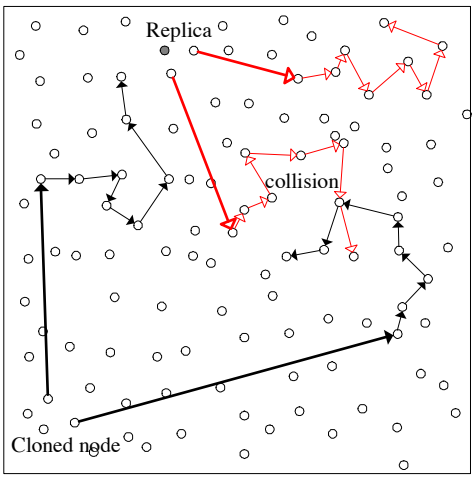


Fig. 3. The RAWL protocol.

difficult 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 claim 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[4]) and uses geographic routing (e.g., GPSR [25]) to forward the claim to the g nodes (or nodes closest to the chosen g locations).

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.

[4]In [9] the authors claimed that choosing location is better than choosing node ID since the available node IDs in the network may be dynamic.

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 Section IX-A by 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.

VII. Table-Assisted Random Walk Based Detection (TRAWL)

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 TRAWL. 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

$$claimDigest = \{MAC_{rand}(Claim)\}_{mod(256)}, \qquad (1)$$

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

TRAWL 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 $P_{none} = (1-\frac{c_2}{\sqrt{n}\log n})^{c_1\sqrt{n}\log n} \approx \frac{1}{e^{c_1c_2}}$. 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 TRAWL. The communication cost of TRAWL is apparently $\sqrt{n}\log n$, the same as RAWL. The memory cost of TRAWL 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 TRAWL reduces the memory cost of RAWL (whose memory cost is $c_1\sqrt{n}\log n \cdot Claim$) more than 10 times when $\sqrt{n}\log n \to \infty$.

VIII. 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]

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

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

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

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

$$P_s = 1 - \big(1 - M\big)^n. \tag{8}$$

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

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

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

$$P_s \approx \Big(1 - \big(1 + \frac{-t}{cn\log n}\big)\Big)^2 n = \frac{t^2}{c^2 n\log^2 n}. \tag{10}$$

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

$$t = c\sqrt{P}\sqrt{n}\log n. \tag{11}$$

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

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

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

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

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

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

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

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

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 $O(\sqrt{n}\log n)$.

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 of

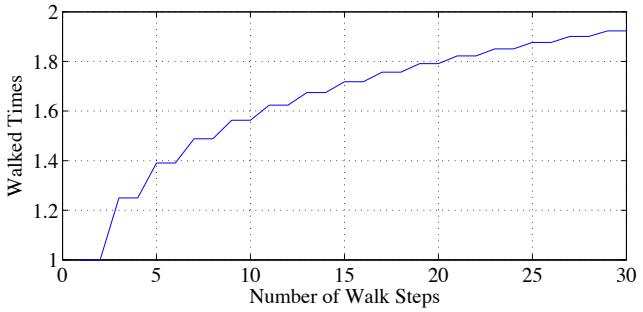


Fig. 4. The walked times of the starting node with different ts in a torus.

each node (includes the starting node) in all the t steps by the formula:

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

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 $t \leq 30$ (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.

IX. 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$\times$1000m square. 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. 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])[5]. 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 TRAWL have exactly the same witness selection, we only present the witness distribution of RAWL.

Fig.5 and Fig.6 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.5, 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.6 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 and Fig.8 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. 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 steps

[5]We 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.

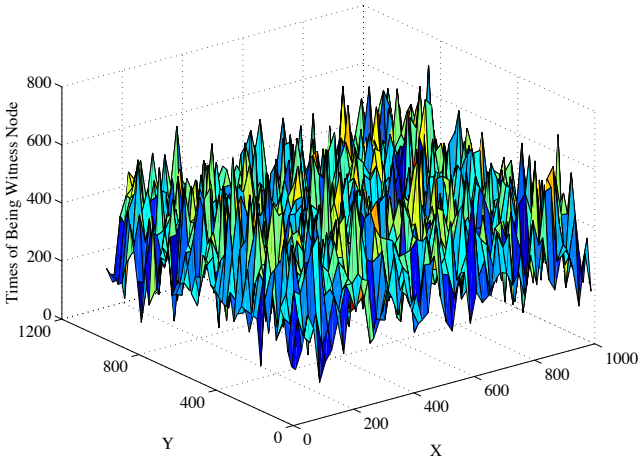


Fig. 5. Witness distribution of RAWL.

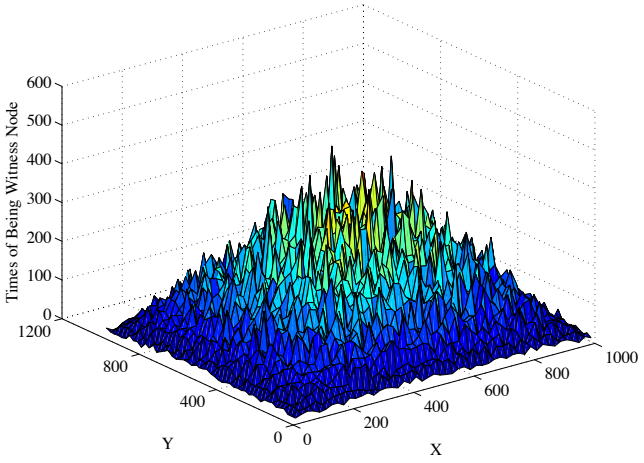


Fig. 6. Witness distribution of LSM.

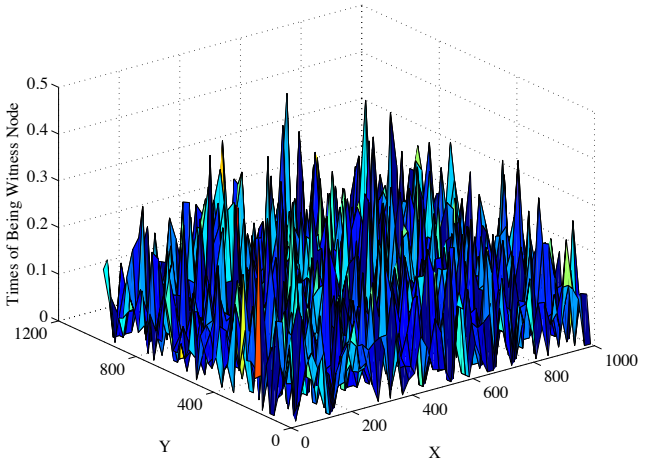


Fig. 7. Witness distribution of RAWL for a node at location (916,813).

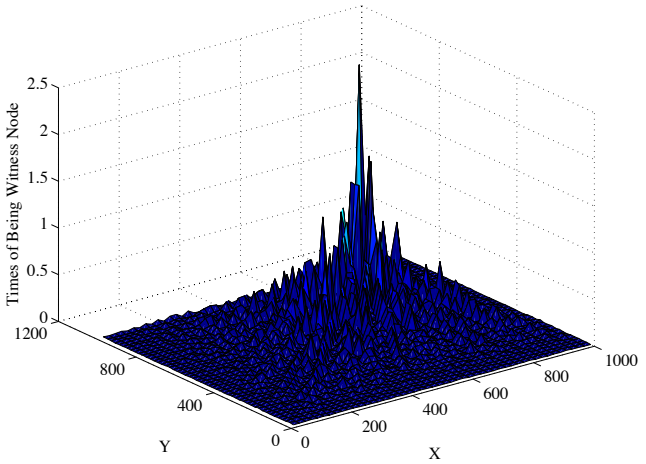


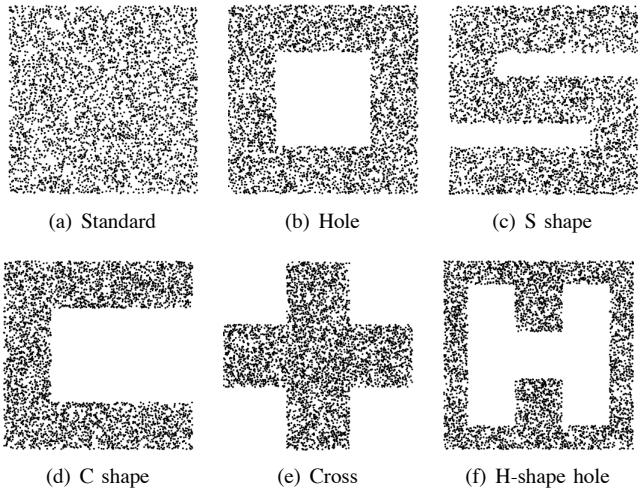
Fig. 8. Witness distribution of LSM for a node at location (916,813).

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.10 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 TRAWL to 4. From Fig.10, first, we can see that TRAWL 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 *claimDigest* 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.

Table III 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. 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.

Next we explain how to choose a suitable r. We here want to guarantee $P_s \geq 0.95$. r should result in the minimum communication overhead, because communication consumes more energy than other operations [26]. We know RAWL needs to start random walks from random distant nodes. Then besides the cost of random walks, whenever we add r by one, we should also add the cost of a path to a random node. Fig.11 shows the average lengths of such paths in different network topologies. In the Standard topology, the length is 15.26. Then $r = 10$ means about $15.26 \times 10 = 152.6$ communications need to be added to the total communication overhead (total communication overhead thus is $152.6 + 10 \times 15 = 302.6$, since t is 15 for guaranteeing $P_s \geq 0.95$). Similarly, we can calculate that when $r = 9$ and $r = 8$ the communication overheads are 299.34 and 314.08 respectively. So we choose $r = 9$ for the following simulations.



(a) Standard (b) Hole (c) S shape

(d) C shape (e) Cross (f) H-shape hole

Fig. 9. Different network topologies for simulation.

TABLE III
PROBABILITY OF DETECTION (P_s) VALUES WITH DIFFERENT NUMBERS OF RANDOM WALKS (r) AND DIFFERENT NUMBERS OF WALK STEPS (t) IN THE 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
t	12	15	18	21	24	27	30
r=10	0.890	0.948	0.976	0.988	0.996	0.995	0.997

C. Communication Overhead and Memory Overhead

Fig.12 shows the communication overhead of different protocols. We just select the Standard-shape topology for simulation, since the performances of our protocols in different topologies are similar. Here in LSM we vary the number of line segments from 1 to 10 in steps of 0.5 to get different detection probabilities. In both RAWL and TRAWL, we fix the r to be 9 (note that as we explained $r = 9$ is only optimal for $P_s \geq 0.95$, we actually should use a smaller r to reduce the communication overhead for a smaller P_s) and vary the number of walk steps (t) from 3 to 30 in steps of 3. In Double Ruling based (DRBased) protocol, we vary the number of crosses from 0.2 to 1 in steps of 0.05. From the figure we can see that to achieve the same detection probability, DRBased protocol consumes the least communication overhead. This is because it uses the ability that one broadcast can be simultaneously received by many neighbors. The communication overloads of RAWL and TRAWL are the same but RAWL has a bit higher detection probability as mentioned in the previous subsection. Also, to make P_s achieve 0.95, RAWL and TRAWL require more than twice the communication overhead of LSM (i.e., 332 vs. 122). Actually, RAWL and TRAWL trade increased communication overhead for stronger security properties.

Fig.13 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 TRAWL 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. TRAWL 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.

In Fig.15 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 TRAWL, $r = 8.5$ in LSM, and $r = 0.95$ in DRBased. 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 between 1.1kB and 12.4kB. The used memory of TRAWL 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.

X. DISCUSSION

The resistance of our protocols to smart adversary relies on that the adversary cannot find out the critical witness nodes. But if the adversary has the strong ability to globally monitor and analyze traffic of the whole network, he can succeed in discovering the whole paths of random walks, even if the communication between each node pair is encrypted. Then he can also find out the critical witness nodes at the intersections of paths. This kind of attack can be prevented by the techniques for *unobservability* [34], [35]. Generally such technique is to mix dummy messages with real messages and reschedule the transmission. During the execution of replica-detection protocol, each node will forward many packets, which may be naturally employed as dummy messages to reduce the communication overhead caused by additional dummy messages.

An adversary may compromise all the neighbors of a replica a to protect a. Then a does not need to send out any location claims during the execution of detection protocol, and thus can escape from detection. In [2] Parno et al. proposed a simple *pseudo-neighbor* method to thwart this attack. In their method each node maintains a list of nodes from which it has seen the most traffic, i.e., acting as their pseudo neighbor. Then, for example, as a's pseudo neighbor, node b can directly

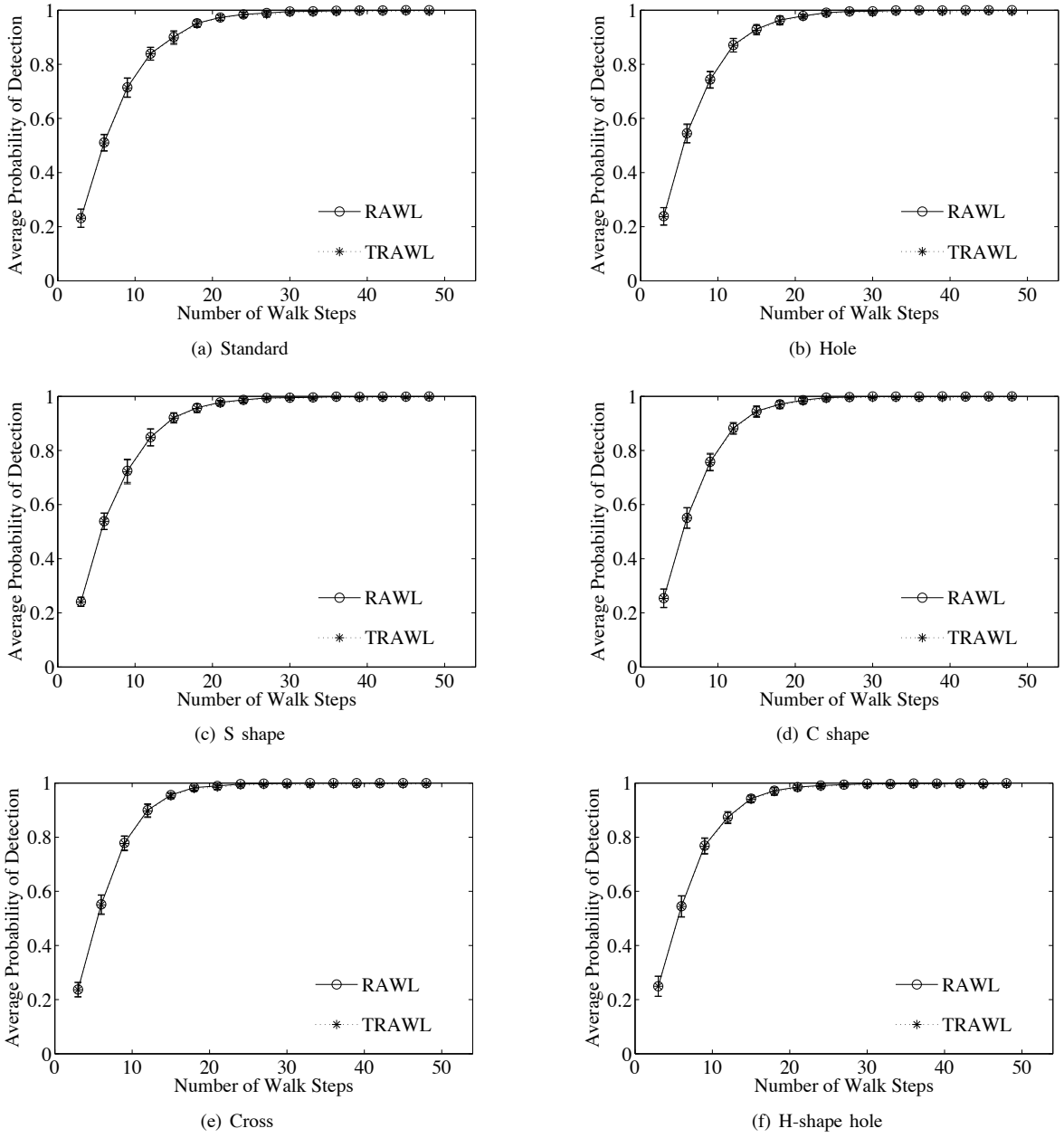


Fig. 10. The probability of detecting the replica in different network topologies, where the number of random walks (r) is 9.

request location claims from a. If a refuses to reply, b will stop forwarding traffic from a. Otherwise, b behaves the same as a common neighbor in the detection protocol.

Same as previous protocols [2], [10], we assume the smart adversary operates in a stealthy manner to avoid detection. Thus the nodes controlled by the smart adversary will still follow the replica-detection protocol. This is because otherwise any detection of misbehavior could make the network administrator start an automated protocol (e.g., SWATT [22]) to sweep the network and remove all the compromised nodes, or remove compromised nodes manually [2]. In [2], the authors proposed to use the sampling method [36] to relax this assumption. We also consider how to detect misbehaving nodes as our future work.

We here discuss the plausible implementations of our protocols. We choose ECDSA signature as the signature of location

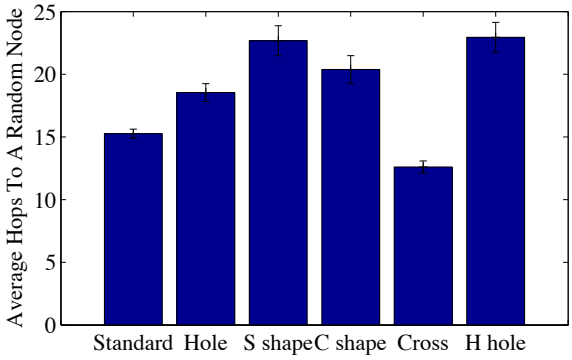


Fig. 11. The average number of hops of the shortest path between two random nodes in the network.

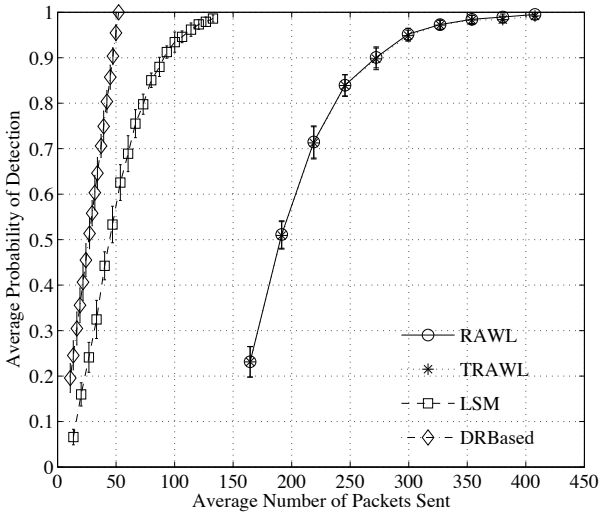


Fig. 12. Comparison of the communication overhead.

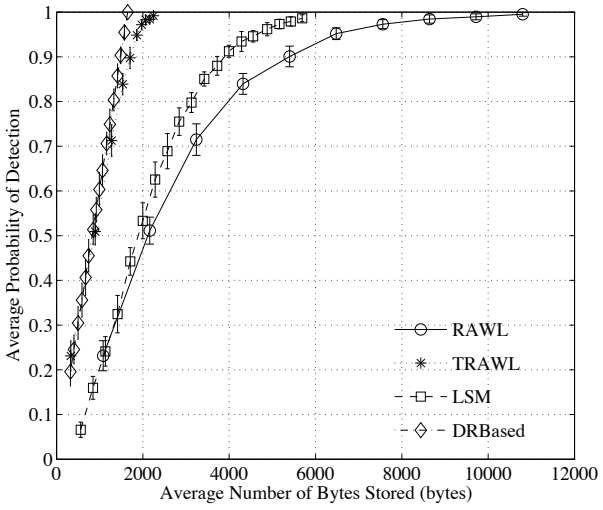


Fig. 13. Comparison of the memory overhead.

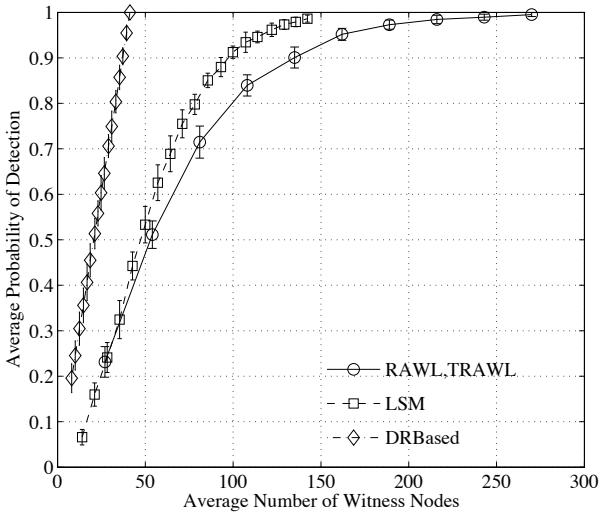


Fig. 14. The average number of witness nodes for each node.

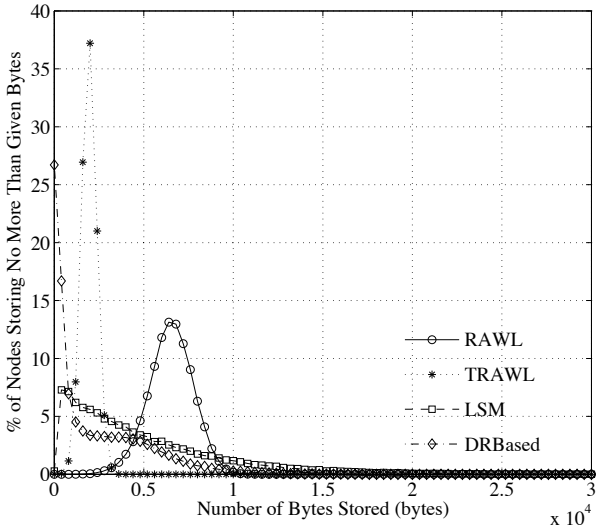


Fig. 15. Used memory distribution.

claim, because of its fast computation. ECDSA is also adopted by IEEE STD. 1609.2 [37] for using in Wireless Access in Vehicular Environments (WAVE). Signature generation and verification in sensor nodes can employ the TinyECC library [20], with which the optimal signature generation and verification on MICAz node need only 2.0s and 2.4s respectively. To reduce code size, we can use a block cipher (e.g., RC5, AES, Skipjack [38], [8]), in Matyas-Meyer-Oseas mode to construct the hash function, and in cipher block chaining (CBC) mode to construct the encryption and MAC functions.

XI. 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 TRAWL, 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 TRAWL 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: Real-world 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. Čapkun 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-2007-11, 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: software-based 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-of-message 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.

Yingpei Zeng received the B.Sc. degree in computer science from Nanjing University, P.R.China, in 2004. Currently he is a Ph.D. student in the Department of Computer Science and Technology at Nanjing University, under the supervision of Prof. Li Xie. He participated in several research projects in Nandasoft Co.,Ltd as an intern. Also he worked as a research assistant in the Department of Computing at Hong Kong Polytechnic University for one year with Prof. Jiannong Cao. His research interests include security in wireless networks and security management.

Jiannong Cao received the B.Sc degree in computer science from Nanjing University, Nanjing, China, and the M.Sc and the Ph.D degrees in computer science from Washington State University, Pullman, WA, USA. He is currently a professor and associate head in the Department of Computing at Hong Kong Polytechnic University. He is also the director of the Internet and Mobile Computing Lab in the department. His research interests include mobile and pervasive computing, computer networking, parallel and distributed computing, and fault tolerance. He has published over 250 technical papers in the above areas. His recent research has focused on wireless networks and mobile and pervasive computing systems, developing test-bed, protocols, middleware and applications. Dr. Cao is a senior member of China Computer Federation, a senior member of the IEEE, IEEE Computer Society, and the IEEE Communication Society, and a member of ACM. He is the Coordinator in Asia of the Technical Committee on Distributed Computing (TPDC) of IEEE Computer Society. Dr. Cao has served as an associate editor and a member of editorial boards of several international journals, including Pervasive and Mobile Computing Journal, Wireless Communications and Mobile Computing, Peer-to-Peer Networking and Applications, and Journal of Computer Science and Technology. He has also served as a chair and member of organizing / program committees for many international conferences, including PERCOM, ICDCS, IPDPS, ICPP, RTSS, SRDS, MASS, PRDC, ICC, GLOBECOM, and WCNC.

Shigeng Zhang received his B.Sc. degree in computer science from Nanjing University, P.R.China, in 2004. In the same year, he joined the Department of Computer Science and Technology of Nanjing University as a Ph.D. student under the supervision of Prof. Daoxu Chen. He worked as a research assistant in the Department of Computing at Hong Kong Polytechnic University from August 2007 to December 2008 under the supervision of Prof. Jiannong Cao. His current research interests include wireless sensor networks, vehicular ad hoc networks and distributed computing.

Shanqing Guo received the Ph.D. degree in computer science from Nanjing University, P.R.China, in 2006. Currently, he is an associate professor in the School of Computer Science and Technology at Shandong University, P.R. China. His research interests include network security and software security. He is a member of China Computer Federation and Chinese Association for Cryptologic Research. He has published more than 20 papers in journals and conferences.

Li Xie is a professor in the Department of Computer Science and Technology at Nanjing University, and now he is also the President of Jiangsu Nandasoft Co.,Ltd. He graduated from the Department of Mathematics of Nanjing University majored in mathematical logic in 1964. He had been the visiting scholar of the Department of Computer Science at New York State University at Albany from 1980 to 1982. He taught in the Department of Mathematics and Department of Computer Science at Nanjing University and had served successively as Deputy Director of the Computer Software Research Institute, Assistant to the University President, Deputy Dean of Studies, Dean of the Department of Computer Science, Director of the Computer Application Research Institute in Nanjing University, and the Vice President of Nanjing University. He has engaged in the research of computer software over a long period of time. With his research achievements, he won 12 awards, including 4 national class awards, and 2 provincial or ministry class special awards, in fields like operating system, distributed computing, parallel processing, and advanced operating system. He has published 4 monographs such as "the course of operating system" and "distributed data processing" as well as more than 220 academic papers.