Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 9, September 2014, pg.176 – 189

RESEARCH ARTICLE

On the Node Clone Detection in Wireless Sensor Networks

T. Shivaiah[1], M. Vazralu[2]

[1]M. Tech IV semester, Dept. of CSE, Malla Reddy College of Engineering and Technology, Hyderabad
t.shiva548@gmail.com

[2]Associate Professor, Dept. of CSE, Malla Reddy College of Engineering and Technology, Hyderabad
Vazram4u@gmail.com

Abstract— Wireless sensor networks are vulnerable to the node clone, and several distributed protocols have been proposed to detect this attack. However, they require too strong assumptions to be practical for large-scale, randomly deployed sensor networks. In this paper, we propose two novel node clone detection protocols with different tradeoffs on network conditions and performance. The first one is based on a distributed hash table (DHT), by which a fully decentralized, key-based caching and checking system is constructed to catch cloned nodes effectively. The protocol performance on efficient storage consumption and high security level is theoretically deducted through a probability model, and the resulting equations, with necessary adjustments for real application, are supported by the simulations. Although the DHT-based protocol incurs similar communication cost as previous approaches, it may be considered a little high for some scenarios. To address this concern, our second distributed detection protocol, named randomly directed exploration, presents good communication performance for dense sensor networks, by a probabilistic directed forwarding technique along with random initial direction and border determination. The simulation results uphold the protocol design and show its efficiency on communication overhead and satisfactory detection probability.

Index Terms—Distributed detection, distributed hash table, node clone attack, randomly directed exploration, wireless sensor networks (WSNs)

I. INTRODUCTION

Wireless sensor networks (WSNs) have gained a great deal of attention in the past decade due to their wide range of application areas and formidable design challenges. In general, wireless sensor networks consist of hundreds and thousands of low-cost, resource-constrained, distributed sensor nodes, which usually scatter in the surveillance area randomly, working without attendance. If the operation environment is hostile, security mechanisms against adversaries should be taken into consideration. Among many physical attacks to sensor networks, the node clone is a serious and dangerous one [1]. Because of production expense limitation, sensor nodes are generally short of tamper-resistance hardware components; thus, an adversary can capture a few nodes, extract code and all secret credentials, and use those materials to clone many nodes out of off-the-shelf sensor hardware. Those cloned nodes

that seem legitimate can freely join the sensor network and then significantly enlarge the adversary's capacities to manipulate the network maliciously. For example, those vicious nodes occupy strategic positions and cooperatively corrupt the collected information. With a large number of cloned nodes under command, the adversary may even gain control of the whole network. Furthermore, the node clone will exacerbate most of inside attacks against sensor networks.

In this paper, we present two novel, practical node clone detection protocols with different tradeoffs on network conditions and performance. The first proposal is based on a *distributed hash table* (DHT) [2], by which a fully decentralized, key-based caching and checking system is constructed to catch cloned nodes. The protocol's performance on memory consumption and a critical security metric are theoretically deducted through a probability model, and the resulting equations, with necessary adjustment for real application, are supported by the simulations.

In accordance with our analysis, the comprehensive simulation results show that the DHT-based protocol can detect node clone with high security level and holds strong resistance against adversary's attacks.

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 mechanism is 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 outperforms all other detection protocols in terms of communication cost, while the detection probability is satisfactory.

The rest of the paper is organized as follows. First, the previous countermeasures are discussed in Section II. Then, we present system preliminaries in Section III. Afterwards, we elaborate the DHT-based detection protocol, analyze its performance, and provide the simulation results in Sections IV–VI, respectively. The randomly directed exploration protocol is detailed in Section VII, and its supportive simulation results are addressed in Section VIII. Finally, we conclude our work in Section IX.

II. PREVIOUS WORK

In general, countermeasures against node clone can be categorized into three categories: prevention schemes that inherently forbid cloned nodes to join network, centralized detection in which there exists a central, powerful party responsible for receiving reports and making judgements of node clone, and distributed detection where all nodes cooperatively process information and detect node clone in a distributed manner.

A. Prevention

Zhang *et al.* [3] proposed the use of location-based keys to defend against several attacks, one of which is node clone attack. The identity-based cryptography is used in their protocol such that nodes' private keys are bounded by both their identities and locations. Once nodes are deployed, some trusted mobile agents travel around the sensor network and issue the location-based keys to sensor nodes. Since those location-based keys cannot be used in nodes at other locations, node clone attack is inherently frustrated.

By similar arguments, we review key distribution protocols for sensor networks, and it can be claimed that some of them prevent node clone as well. For example, in schemes [4], [5] based on initial trust which assume that it takes adversaries a certain amount of time to compromise nodes after their deployment, valid keys only can be established during that safety period, and henceforth compromising nodes will not grant adversaries extra advantages, including the ability to cloned nodes.

Those prevention schemes might be useful on particular applications, but their assumptions as trusted mobile agents and initial trust are too strong to be applicable in general cases.

B. Centralized Detection

In a simplest centralized detection approach, each node sends a list of its neighbor nodes and their locations to a base station, which then can find cloned nodes. The SET protocol [8] manages to reduce the communication cost of the approach above by constructing exclusive subsets such that each node belongs to one and only one disjointed subset, and the subset nodes information is reported to the base station by a subset leader.

However, in order to prevent malicious nodes, an authenticated subset covering protocol has to be performed, which considerably increases the communication burden and complicates the detection procedure.

Brooks *et al.* [9] proposed a clone detection protocol in the context of random key predistribution [10]. Technically, it is detecting compromised keys rather than cloned nodes. The basic idea is that the keys employed in random key predistribution scheme should follow a certain pattern, and those keys whose usage exceeds a threshold can be thought to be suspicious. In the protocol, every node reports its keys to a base station, and then the base station performs an abnormality-based intrusion-detection-like statistical analysis to catch cloned keys. A common concern for this kind of approach is its high false negative and positive rates. Furthermore, the authors do not address how to assure malicious nodes to honestly report their keys, which is critical to the protocol effectiveness.
As pointed out in [1], centralized approaches are prone to single point of failure, and the nodes surrounding the base station suffer an undue communication burden that may shorten the network's life expectancy. In general, a distributed, balanced detection scheme is more desirable.

C. Distributed Detection

The straightforward node-to-network broadcasting [1] is a quite practical way to distributively detect the node clone, in which every node collects all of its neighbors identities along with their locations and broadcasts to the network. The main problem in this approach is its extremely high communication overhead.
Parno *et al.* [1] provided two probabilistic detection protocols in a completely distributed, balanced manner. Randomized multicast scheme distributes node location information to randomly selected nodes as inspectors, exploiting the birthday paradox to detect cloned nodes, while line-selected multicast scheme uses the topology of the network to improve detection—that is, in addition to inspector nodes, the nodes along the multicast path check the node clone as well. Unfortunately, to obtain acceptable detection probability, nodes have to buffer a great many of messages. Moreover, the communication cost in the randomized multicast is similar to that in the node-to-node broadcasting.
For the procedure of choosing random inspectors, both schemes imply that every node is aware of all other nodes' existence, which is a very strong assumption for large-scale sensor networks and thus limits their applicability. Based on the geographic hash table, which maps a key into a geographical coordination, Zhu *et al.* [7] and Conti *et al.* [6] proposed several clone detection schemes. Their approaches rely on the nodes' knowledge of the general deployed geography of sensor networks. This prerequisite may hold in some circumstances, but cannot be guaranteed generally. Table I compares those distributed detection protocols along with our two proposals in terms of requirements, communication cost, memory consumption, and detection level.

III. PRELIMINARIES

A. Network Model

We consider a large-scale, homogeneous sensor network consisting of resource-constrained sensor nodes. Analogous to previous distributed detection approaches; we assume that an identity-based public-key cryptography facility [11] is available in the sensor network. Prior to deployment, each legitimate node is allocated a unique ID and a corresponding private key by a trusted third party. The public key of a node is its ID, which is the essence of an identity-based cryptosystem. Consequently, no node can lie to others about its identity. Moreover, anyone is able to verify messages signed by a node using the identity-based key. Let and denote the public and private keys of node, respectively, and represent the signature of signed by node.
We also assume that every sensor node can determine its geographic location and current relative time via a secure localization protocol and a secure time synchronization scheme, respectively.
A number of those mechanisms have been proposed, which can be referred to in [12]. We do not specify the particular selections of secure localization and time synchronization schemes for our protocols since they are comparatively irrelevant to our proposals.
There may or may not be a powerful base station in our modeled network, but there should exist a trusted role named *initiator* that is responsible for initiating a distributed detection procedure. Otherwise, an adversary can readily launch a denial-of-service (DoS) attack to the system by repeatedly mobilizing the sensor network to conduct the clone detection protocol and exhausting nodes energy.

B. General Detection Guidelines

Relying on the identity-based cryptography, secure localization, and secure time synchronization used in our network model, node clone in sensor networks can be determined by the occurrence of nodes with same ID appearing on reasonably distant locations at a designated time. Specifically, at the beginning time of a round of detection that is specified by the initiator, the information regarding the ID and location of every node is claimed by its neighbors for the clone detection. In this sense, the neighbors of a node are its *observers*. Subsequently, some nodes will be selected as *inspectors* to examine claiming messages for the purpose of clone detection. If an inspector

successfully finds a clone, it becomes a *witness*, which will broadcast necessary evidence to inform all connected nodes revoking the cloned nodes. While the initiator is presumably trusted, the other roles (observer, inspector, and witness) might be compromised by the adversary and behavior maliciously.
The four roles in our protocols are summarized in Table II.

C. Performance Metrics

The following metrics are used to measure a protocol's performance and evaluate its practicability.
• *Detection probability and security level:* As a primary security requirement, a practical detection scheme should detect the occurrence of the attack with high probability.
Thus, the *detection probability* is the most important security metric for a *probabilistic* clone detection scheme. On the other hand, if a detection protocol is *deterministic* in the sense that cloned nodes are always caught by witnesses, and it is also a fully *symmetric* approach in which nodes are equally likely to become witnesses prior to a round of detection procedure, we will use the number of witnesses to evaluate the security level because more witnesses improve protocol resilience against the adversary's potential attacks to witnesses.

• *Communication cost:* Communication cost is always a crucial performance metric for sensor network protocols because usually energy is the most valuable resource for nodes, and message transmission consumes at least one order of magnitude power than any of the other operations [13]. For simplicity, we use the average number of messages sent per node to represent a protocol's communication cost.
• *Storage consumption:* Ordinary, low-cost sensor nodes are only equipped with a limited amount of memory; thus, any schemes requiring high storage will be considered as impractical.
• *Balance:* In a homogeneous sensor network, schemes are supported to consume the energy and memory in a balanced fashion. It should be avoided to create hot nodes that would be buffer-overflowed or die away quickly.

D. Adversary Model

We consider a threat model in which sensor nodes are deployed in a hostile environment and are subject to capture and complete control by an adversary, but the adversary only can compromise a limited number of sensor nodes, and then the adversary uses the compromised nodes to clone many nodes and deploys the replicas in places that are intelligently decided.
Specifically cloned nodes are minority in the network. In addition, we assume each cloned node has at least one neighbor that remains intact.
The adversary definitely wants to conceal the existence of clone. In our settings, this enemy is allowed to interfere with a detection protocol as follows. First, the cloned nodes may not participate in the regular detection procedures. Second, the nodes controlled by the adversary may fake, drop, or manipulate claiming messages that they forward. Third, the adversary can capture some nodes accordingly, but it would take time, and the total number of nodes that an adversary can compromise is limited. All nodes that are not controlled by the adversary will be referred to as *integrity nodes*.
The adversary may also try to abuse a detection protocol to frame innocent nodes as cloned such that they will be expelled from the network. This is called framing attack, and approaches should be provided to address this issue.

IV. DHT-BASED DETECTION PROTOCOL

The principle of our first distributed detection protocol is to make use of the DHT mechanism to form a decentralized caching and checking system that can effectively detect cloned nodes. Essentially, DHT enables sensor nodes to distributively construct an *overlay network* upon a physical sensor network and provides an efficient key-based routing within the overlay network. A message associated with a key will be transmitted through the overlay network to reach a destination node that is solely determined by the key; the source node does not need to specify or know which node a message's destination is—the DHT key-based routing takes care of transportation details by the message's key. More importantly, messages with a same key will be stored in one destination node. Those facts build the foundation for our first detection protocol. As a beginning of a round of DHT-based clone detection, the initiator broadcasts the action message including a random *seed*. Then, every observer constructs a claiming message for each neighbor node, which is referred to as an *examinee* of the observer and the message, and sends the message with probability independently. The introduction of the claiming probability is intended to reduce the communication overwork in case of a high-node-degree network. In the protocol, a message's DHT key that determines its routing and destination is the hash value of concatenation of the seed and the examinee ID. By means of the DHT mechanism, a claiming message will eventually be transmitted to a deterministic destination node,

which will cache the ID-location pair and check for node clone detection, acting as an inspector. In addition, some intermediate nodes also behave as inspectors to improve resilience against the adversary in an efficient way.

A. Distributed Hash Table

Before diving into the detection protocol, we briefly introduce DHT techniques. In principle, a distributed hash table is a decentralized distributed system that provides a key-based lookup service similar to a hash table: (key, record) pairs are stored in the DHT, and any participating node can efficiently store and retrieve records associated with specific keys. By design, DHT distributes responsibility of maintaining the mapping from keys to records among nodes in an efficient and balanced way, which allows DHT to scale to extremely large networks and be suitable to serve as a facility of distributed node clone detection.

There are several different types of DHT proposals, such as CAN [14], Chord [15], and Pastry [16]. Generally, CAN has least efficiency than others in terms of communication cost and scalability, and it is rarely employed in real systems. By contrast, Chord is widely used, and we choose Chord as a DHT implementation to demonstrate our protocol. However, our protocol can easily migrate to build upon Pastry and present similar security and performance results.

The technical core of Chord [15] is to form a massive virtual ring in which every node is located at one point, owning a segment of the periphery. To achieve pseudo-randomness on output, a hash function is used to map an arbitrary input into a -bit space, which can be conceived as a ring. Each node is assigned with a Chord coordinate upon joining the network. Practically for our protocol, a node's Chord point's coordinate is the hash value of the node's MAC address. All nodes divide the ring into segments by their Chord points. Likewise, the key of a record is the result of the hash function. Every node is responsible for one segment that ends at the node's Chord point, and all records whose keys fall into that segment will be transmitted to and stored in that node.

As the kernel of efficient key-based routing, every node maintains a *finger table* of size to facilitate a binary-tree search. Specifically, the finger table for a node with Chord coordinate contains information of nodes that are respectively responsible for holding the keys: for .

If two nodes are within the ring-segments distance, they are each other's predecessor and successor by the order of their coordinates, with respect to predefined . In theory, a Chord node only needs to know its direct predecessor and finger table. To improve resilience against network churn and enhance routing efficiency, every node additionally maintains a successor table, containing its successors. Typical values of are between 10 and 20.

A demonstrative example of a Chord system with small parameters is given in Fig. 1. In this system, if node wants to query a record with key , it first looks up its successor table. Since 97 is not in (109, 61], namely (direct predecessor, the last successor], node proceeds with the finger table and finds that the next forwarding node should be because 97 [72: the first item in finger table corresponding to 109: direct predecessor). When receives this query about, by checking its successor table with two nodes of and , it determines the destination should be , as (88: itself, 109: the first successor]. When node gets the query, it knows itself be the destination because 97 (88: direct predecessor, 109: itself].

By this routing mechanism, on average of queries toward a destination pass through one of predecessors. For node as the destination, its two predecessors are and the latter's direct predecessor is . The line from and has the length of and is divided by and into three segments with lengths of, and . Therefore, the probabilities of a query passing through and are and ,respectively.

B. Protocol Details

As a prerequisite, all nodes cooperatively build a Chord overlay network over the sensor network. Cloned node may not participate in this procedure, but it does not give them any advantage of avoiding detection. The construction of the overlay network is independent of node clone detection. As a result, nodes possess the information of their direct predecessor and successor in the Chord ring. In addition, each node caches information of its consecutive successors in its *successors table*. Many Chord systems utilize this kind of cache mechanism to reduce the communication cost and enhance systems robustness. More importantly in our protocol, the facility of the successors table contributes to the economical selection of inspectors.

One detection round consists of three stages.

Stage 1: Initialization

To activate all nodes starting a new round of node clone detection, the initiator uses a broadcast authentication scheme to release an *action message* including a monotonously increasing nonce, a random round seed, and an action time. The nonce is intended to prevent adversaries from launching a DoS attack by repeating broadcasting action messages.

Stage 2: Claiming neighbors information

Upon receiving an action message, a node verifies if the message nonce is greater than last nonce and if the message signature is valid. If both pass, the node updates the nonce and stores the seed. At the designated action time, the node operates as an observer that generates a claiming message for each neighbor (examinee) and transmits the message through the overlay network with respect to the claiming probability. The claiming message by observer for examinee is constructed by where are locations of and , respectively.

Nodes can start transmitting claiming messages at the same time, but then huge traffic may cause serious interference and degrade the network capacity. To relieve this problem, we may specify a sending period, during which nodes randomly pick up a transmission time for every claiming message.

Stage 3: Processing claiming messages

A claiming message will be forwarded to its destination node via several Chord intermediate nodes. Only those nodes in the overlay network layer (i.e., the source node, Chord intermediate nodes, and the destination node) need to process a message, whereas other nodes along the path simply route the message to temporary targets. Algorithm 1 for handling a message is the kernel of our DHT-based detection protocol. If the algorithm returns NIL, then the message has arrived at its destination. Otherwise, the message will be subsequently forwarded to the next node with the ID that is returned by Algorithm 1.

Criteria of determining inspectors: During handling a message in Algorithm 1, the node acts as an inspector if one of the following conditions is satisfied.

Algorithm 1: : handle a message in the DHT-based detection, where is the current node's Chord coordinate, is the first node on the ring that succeeds key is the next th successor,

Output: NIL if the message arrives at its destination; otherwise, it is the ID of the next node that receives the message in the Chord overlay network

1:
2: **if then** has reached destination
3: act as an inspector, see Algorithm 2
4: **return** NIL
5: **for** to **do**
6: **if then** destination is in the
next Chord hop
7: act as an inspector, see
Algorithm 2
8: **return**
9: **for** to **do** for normal DHT routing process
10: **if then**
11: **return**
12: **return**

Algorithm 2: : Inspect a message to check for
clone detection in the DHT-based detection protocol
1: verify the signature of
2: **if** found in cache table **then**
3: **if** has two distinct locations found clone,
become a witness
4: broadcast the evidence
5: **else**
6: buffer into cache table

1) This node is the destination node of the claiming message.
2) The destination node is one of the successors of the node.

In other words, the destination node will be reached in the next Chord hop. While the first criterion is intuitive, the second one is subtle and critical for the protocol performance. By Algorithm 1, roughly of all claiming messages related to a same examinee's ID will pass through one of the predecessors of the destination. Thus, those nodes are much more likely to be able to detect a clone than randomly selected inspectors.

As a result, this criterion to decide inspectors can increase the average number of witnesses at a little extra memory cost. We will theoretically quantify those performance measurements later.

In Algorithm 1, to examine a message for node clone detection, an inspector will invoke Algorithm 2, which compares the message with previous inspected messages that are buffered in the cache table. Naturally, all records in the cache table should have different examinee IDs, as implied in Algorithm 2. If detecting a clone, which means

that there exist two messages and satisfying and , the witness node then broadcasts the evidence to notify the whole network.

All integrity nodes verify the evidence message and stop communicating with the cloned nodes. To prevent cloned nodes from joining the network in the future, a revocation list of compromised nodes IDs may be maintained by nodes individually. It is worth noting that messages and are authenticated by observers and , respectively. Therefore, the witness does not need to sign the evidence message. If a malicious node tries to launch a DoS attack by broadcasting a bogus evidence message, the next integrity node receiving it can immediately detect the wicked behavior by verifying the signatures of and before forwarding to other nodes.

C. Security Discussions

Validity of Detection: The identity-based cryptographic system provides reliable identity authentication and message authentication for the DHT-based protocol. As a result, the adversary cannot falsify cloned nodes' IDs; neither can the modify messages signed by integrity nodes. Moreover, a cloned node cannot lie to its observers about its location since a forged location would be far deviated from the communication range of the observers, which suffices to alert observers. Therefore, the detection guidelines are robust provided observers are honest.

Thwarting Framing Attack: A witness cannot forge an evidence to frame integrity nodes since the evidence eventually are composed of claiming messages from different observers, and any node can verify them. However, for any witness-based detection protocol, including our two proposals and protocols in Table I, there exists a realistic threat that some malicious observers try to frame innocent nodes by claiming wrong locations for them. To thwart this attack, we introduce a mechanism that requires nodes to buffer evidence messages they received and to maintain a debit table for those observers in evidence. When a node is declared as a clone in one or more evidences, assume one of its distinct locations is claimed by different observers, then each of those observers should be debited by . If a node's balance in the debit table exceeds a threshold, it should be revoked as well. We recommend one for the threshold. In this case, if a malicious node tries to frame an integrity node by claiming a false location for it, both nodes will be revoked. This one-exchanging-one strategy is quite fair and efficient as we do not need to distinguish which one is bad. When there is no framing attack in the network, integrity nodes are rarely revoked because a clone node's location may be claimed by many observers. At most, the number of revoked integrity nodes will not exceed the number of malicious nodes.

Protecting Witnesses: Technically, the hash functions used in DHT do not need to be cryptographic hash functions. In practice, cryptographic ones are usually employed in the DHT systems because of their well uniformly random distribution of outputs and preventing potential abuses. For our protocol, a cryptographic hash function is indeed required because it will restrain the adversary's abilities by application on as he cannot distinguish which nodes could more likely be witnesses before a round of detection. After disclosure of the random seed, the adversary may want to comprise witnesses to defeat detection.

However, there are potential witnesses that are geographically randomly distributed in the network. Determining and capturing all the witnesses will be troublesome for the adversary. Moreover, those witnesses vary round by rounds. Consequently, the adversary cannot stop the detection by trying to capture a few witnesses.

Coping With Message-Discarding: The cloned nodes may discard claiming messages through them. Our protocol is resilient against it due to the characteristic of full distributiveness and balance of the DHT-based protocol. If there are only a few cloned nodes, the impact of this malicious action will be insignificant. When the number of cloned nodes increases, more claiming messages will assure sufficient number of witnesses.

The simulations later clearly indicate this result.

In summary, the DHT-based detection protocol is secure in the adversary model.

V. PERFORMANCE ANALYSIS OF DHT-BASED PROTOCOL

For the DHT-based detection protocol, we use the following specific measurements to evaluate its performance:

• *average number of transmitted messages*, representing the protocol's communication cost;

• *average size of node cache tables*, standing for the protocol's storage consumption;

• *average number of witnesses*, serving as the protocol's security level because the detection protocol is deterministic and symmetric.

A. Communication Cost

We denote the average path length between two random nodes by , which varies from to , dependent on underlying sensor networks. According to the Chord's properties [15], the average Chord-hop of a message—that is, the number of transfers in the Chord overlay network—is , where is a constant number, usually less than 1. Therefore, the average path hop length of a message is . There are claiming messages in total for a round of detection.

B. Storage Consumption and Security Level

For simplicity, we hereby assume that all nodes, including compromised ones, abide by the detection protocol. Later in Section VI, we will see that the malicious behaviors such as discarding claiming messages only slightly affect those performance measurements.

In this detection protocol, claiming messages associated with a same examinee's ID will be transported to one destination node. Because there are examinees and potential destinations, and due to the good pseudo-randomness of the Chord system, on average, every node stores one record in its cache table associated with one examinee's ID as its destination, regardless of the number of claiming messages per examinee. In addition, for a designated examinee, the predecessor nodes of the destination can act as inspectors; thus, they probably hold up to one record related to the examinee.

We consider an ideal case that there are claiming messages for each examinee, and those messages are independently transmitted. Let denote the probability of a predecessor receiving a specific claiming message, then the probability of a predecessor holding a record for an examinee is Consequently, the average cache-table size can be calculated by If there are cloned nodes with a same ID in the network, then the destination node of claiming messages related to those nodes as examinees will deterministically become a witness to successfully catch the attack, while the predecessor nodes of the destination may become witnesses if and only if they receive at least two claiming messages associated with different cloned nodes. It is easy to see that this probability is minimized as when. Therefore, as a lower bound for witness number, we only consider the case that there are two cloned nodes. In this case, the average witness number can be obtained by

1) Continuous Probability Model: We formalize a probability model to calculate . Because any practical Chord coordinate space is massive , we should use continuous probability model rather than discrete one. Recall that in the demonstrative example in Section IV-A, there is a line divided by points into segments, and is determined by the ratio of segment length over line length. Based on the pseudo-randomness of the cryptographic hash function, we can safely assume that all points coordinates are independent. Therefore, for any line constituted by consecutive points, the intermediate points are random.

Definition 1 (Random Line Segmentation Model): There is a line of length 1, of which the starting and ending points coordinates are 0 and 1, respectively. We randomly generate values over as point coordinates, and let random variable represent the th point, for . Those continuous random variables divide the line into segments. We want to determine the continuous probability distributions of segment lengths.

We denote density function by , cumulative distribution function by , and expected value by . Since the sample spaces of all random variables in this paper are , then is implied for all cumulative distribution functions and density functions in this paper. Since intermediate points are randomly selected, all segment lengths have a same probability distribution. Let denote the random variable of the length of the first segment, which originates from 0.

The intermediate points in the proceeding model are corresponding to the predecessors, and the ending point represents the destination. In addition, due to pseudo-randomness of the Chord system, it is reasonable to assume that one claiming message randomly occurs in that line before the final transmission to the destination. Therefore, we can directly adopt as the probability distribution of a predecessor receiving one claiming message.

2) Analytic Formulas for Average Cache-Table Size and
Average Witness Number:

Theorem 1: In an ideal case where there are independent claiming messages for each examinee and is the successors table size, the average cache-table size is (1) and the average witness number, when there are two cloned nodes, is (2)

Proof: For a specific examinee's ID, let represent the probability of a predecessor receiving at least one claiming message, which results in one record in its cache table. By the analysis before, ; thus, we can get and the density function is obtained by Thus, the expected value of is When there are two cloned nodes, each of them incurs independent claiming messages. Let represent the probability of a predecessor becoming a witness. Therefore, in the ideal case, the average size of cache tables is and the average witness number is That concludes the proof.

It is worth mentioning that if is constant, the average witness number is maximized when because by (2) and the nonnegative denominator takes its minimum value when. Consequently, the maximum witness number is.

3) Remarks: In the reality of the DHT-based protocol, each examinee, on average, has claiming messages. Thus, we can use . However, the transmissions of claiming messages associated with a same examinee's ID actually converge over the Chord key-based routing system. For a specific status of a Chord system, each segment has a different length, which would affect the hitting probability. In the ideal case, we take into account one level of such unbalanced segment. Therefore, if all messages transmissions can be completed in two hops (that is, the first hop can reach the destination or one of its predecessors), the ideal case analyze is directly applicable.

However, most messages shall go through more than two hops; then, every previous hop transmission will introduce correlation on the next hop transmission. Consequently, they are not independent, and it is very difficult to directly model and analyze performance for the multilevel effect. Fortunately, through simulation studies, we discover that we may use those equations for the ideal case with some necessary adjustment to approximate the protocol practical performance on storage consumption and security level. Specifically, the message transmission correlation impact can be mimicked by decreasing to a certain proportion, when we apply (1) and (2) to estimate the average cache-table size and the average witness number.

In other words, even though the actual successors-table size is, when we calculate performance measurements by those two equations, we use , which is smaller than with respect to a certain ratio. We will demonstrate it in Section VI of thorough simulations.

VI. SIMULATIONS FOR DHT-BASED PROTOCOL

We implement the DHT-based detection protocol and run simulations to evaluate performance comprehensively on the OMNeT++ framework [17].

We design the simulations in two network scenarios. The first is an abstract network following a *random graph* model.

By definition, a random graph is a graph that is generated by starting with a set of vertices and adding edges between them at random. The other one is a practical *unit-disk graph*, in which nodes are uniformly deployed in a square and follow the standard unit-disk bidirectional communication model. In our simulations, node communication ranges are dynamically adjusted such that the average node degree approximates.

A. Performance on Varying Network Sizes

The following parameters are used in the simulations: finger table size , successors table size , and node degree. Two different values of claiming probability are used as 1.0 for pro-security and 0.2 for pro-communication cost.

We design and conduct the first simulation to measure the protocol's performance on different network sizes, ranging from 500 to 5000. In order to obtain relatively fair and comparable results, for each case, 10 different network instances in accordance with the parameter settings are constructed. Each of those simulation executions is quoted as a run; one run performs 20 rounds of detection. In each of those rounds, a random seed is generated, and two nodes are randomly chosen to set the same ID, that is, those two are cloned nodes.

The average number of messages sent per integrity node in this simulation is plotted in Fig. 2(a). In the graph, the communication payloads for groups of are exactly one fifth of those of with the same other parameters. The distinction between simulation results of random graph and those of unit-disk graph results from different average path hops for a pair of random nodes in those two network scenarios. Indeed, the average Chord-hop per message in the simulations, depicted in Fig. 2(b), is independent of the network scenarios and matches the Chord system properties—it is proportional to The average size of cache tables for integrity nodes and the average witness number are illuminated in Fig. 2(c) and (d), respectively, which clearly indicates that those two performance metrics are not notably affected by network scenarios and network size if is sufficiently large, and thus the detection protocol does scale to network size. Even though the network topology for unit-disk model used in the simulations is square, the protocol can effectively detect clone attack in irregular topologies, like "Thin H," "Thin Cross," "S," "Large H," "L," and "Large H," which are used in the simulations of [1]. The communication payload of our DHT-based protocol for those network topologies is proportional with the average path hop while the storage payload and the security level stay constant.

B. Results on Different Numbers of Cloned Nodes

We develop the second simulation to evaluate the protocol's performance on the different numbers of cloned nodes. We run simulations with one network size , and the cloned node number increases from 2 to 100. We test each case with 10 runs, and for each run we repeat 200 rounds of node detection, in each of which a seed is randomly generated and nodes are randomly chosen as clones.

Fig. 3 depicts the simulation results about the average size of cache tables for integrity nodes and the average number of witnesses, which support our security arguments in Section IV-C.

In particular, we can see that the protocol shows strong resilience against message-discarding by cloned nodes. Even if there are 10% nodes that maliciously discard messages, the number of witnesses is pretty high. In fact, the more cloned nodes, the less the size of cache tables for integrity nodes as storage consumption and the more witnesses as security level.

Therefore, we really only need to consider the boundary case of for performance measurements. In Fig. 3, when there are more than 1% cloned nodes, the simulation results for random graph and unit-disk graph are evidently distinct. This is because the message-dropping by malicious nodes affects the performance to a different extent. As

implied in Fig. 2(a), the average transmission hop of claiming messages in unit-disk graph is greater than that in random graph, then messages are more likely to be dropped by cloned nodes in unit-disk graph.

C. Verification of Performance Analysis

To evaluate its applicability of the theoretical analysis on cache-table size and witness number in Section V-B, we carry out a third simulation, in which , there are two cloned nodes, and the claiming probability increases from 10% to 100%. Since network scenarios do not affect the results on the two performance metrics, we run the simulation only on random graph. For the purpose of comparison, we test the performance for both cases that cloned nodes drop messages and comply with protocol (no-dropping).

We argued before that to apply (1) and (2), we should adjust the successors-table size to balance the effect of message transmission correlation. Fig. 4 depicts the simulation results along with theoretical outputs by adopting 80% of original in those two equations. From this figure, first we can see that message-dropping strategy by cloned nodes does not affect the performance notably. Second, the theoretical outputs do reflect the practical results after we choose a proper adjustment on.

Admittedly, the specific adjustment value is related to Chord system parameters. Interestingly, for several other testing cases, such as and , the 80% appears good enough for evaluation purpose.

D. Discussions

From the simulation results, we can see that the proposed DHT-based protocol can effectively detect clone for general sensor networks with high security level and efficient storage consumption, while its communication cost is in the same order of magnitude with previous detection schemes. One way to improve the communication performance is replacing the Chord overlay network with some specific DHT implementations on sensor networks, e.g., virtual cord protocol [18]. Combinations of our protocol with this kind of DHT scheme might be an interesting research topic. On the other hand, if it is very likely for the adversary to deploy many clones out of one ID, we can use small claiming probability for saving communication payload, without degrading the security level dramatically.

The protocol security level, by (2), is directly determined by message number, which is upper-bounded by node degree. If the DHT-based protocol operates in sparse sensor networks, to achieve a desirable witness number, several independent Chord systems by different round seeds may be used.

Thus, messages for one examinee will be indexed by a few deterministic destinations. In this case, the protocol's communication payload, storage consumption, and the average witness number are all multiplied by a same order. This is also a common practice in peer-to-peer (P2P) indexing systems to enhance robustness.

VII. RANDOMLY DIRECTED EXPLORATION

The DHT-based detection protocol can be applied to general sensor networks, and its security level is remarkable, as cloned nodes will be caught by one deterministic witness plus several probabilistic witnesses. However, the message transmission over a Chord overlap network incurs considerable communication cost, which may not be desired for some sensor networks that are extremely sensitive to energy consumption. To fulfill this challenge, we propose the randomly directed exploration (RDE), which tremendously reduces communication cost and presents optimal storage expense with adequate detection probability.

The RDE protocol shares the major merit with broadcasting detection: Every node only needs to know and buffer a neighbor-list containing all neighbors IDs and locations. For both detection procedures, every node constructs a claiming message with signed version of its neighbor-list, and then tries to deliver the message to others which will compare with its own neighbor-list to detect clone. For a dense network, broadcasting will drive all neighbors of cloned nodes to find the attack, but in fact one witness that successfully catches the clone and then notifies the entire network would suffice for the detection purpose.

To achieve that in a communicatively efficient way, we bring several mechanisms and effectively construct a multicast routing protocol. First, a claiming message needs to provide maximal hop limit, and initially it is sent to a random neighbor.

Then, the message subsequent transmission will roughly maintain a line. The line transmission property helps a message go through the network as fast as possible from a locally optimal perspective. In addition, we introduce border determination mechanism to significantly reduce communication cost. We can do all of those because every node is aware of its neighbors locations, which is a basic assumption for all witness-based detection protocols but rarely utilized by other protocols.

Algorithm 3: : An intermediate node processes a message
1: verify the signature of
2: compare its own neighbor-list with the neighbor-list in
3: **if** found clone **then**
4: broadcast the evidence;
5:
6: **if then**
7: discard
8: **else**
9: See Algorithm 4
10: **if then**
11: discard
12: **else**
13: forward to

A. Protocol Description

One round of clone detection is still activated by the initiator.

Subsequently, at the designated action time, each node creates its own neighbor-list including the neighbors IDs and locations, which constitutes the sole storage consumption of the protocol. Then, it, as an observer for all its neighbors, starts to generate a claiming message containing its own ID, location, and its neighbor-list. The claiming message by node is constructed by where is time to live (a.k.a. message maximum hop). Since will be altered by intermediate nodes during transmission, it should not be authenticated. The observer will deliver the claiming message times. In each time, the node transmits it to a *random* neighbor as indicated in Fig. 5(a). Note that can be a real number, and accordingly an observer transmits its claiming message at least , up to , and on average times.

When an intermediate node receives a claiming message, it launches , which is described by pseudocode in Algorithm 3, to process the message. During the processing, node , as an inspector, compares its own neighbor-list to the neighbor-list in the message, checking if there is a clone. Similarly, if detecting a clone, the witness node will broadcast an evidence message to notify the whole network such that the cloned nodes are expelled from the sensor network. To deal with routing, node decreases the message's by 1 and discards the message if reaches zero; elsewhere, it will query Algorithm 4 to determine the next node receiving the message.

Algorithm 4: : To determine the next node that receives the message
1: determine ideal angle, target zone, and priority zone
2: **if** no neighbors within the target zone **then**
3: **return** NIL
4: **if** no neighbors within the priority zone **then**
5: the node closest to ideal angle
6: **else**
7: a probabilistic node in the priority zone, with respect to its probability proportional to angle distance from priority zone border
8: **return**

Essentially, Algorithm 4 contains the following three mechanisms.

• *Deterministic directed transmission* [Fig. 5(b)]*:* When node receives a claiming message from previous node, the ideal direction can be calculated. In order to achieve the best effect of line transmission, the next destination node should be node , which is closest to the ideal direction.

• *Network border determination* [Fig. 5(c)]*:* This takes network shape into consideration to reduce the communication cost. In many sensor network applications, there exist outside borders of network due to physical constrains. When reaching some border in the network, the claiming message can be directly discarded. In our proposal for border local determination, another parameter *target range* is used along with ideal direction to determine a *target zone*. When no neighbor is found in this zone, the current node will conclude that the message has reached a border, and thus throw it away.

• *Probabilistic directed transmission* [Fig. 5(d)]*:* In the probabilistic directed transmission, parameter *priority range* along with the ideal direction is used to specify a priority zone, in which the next node will be selected. When no nodes are located in that zone, the deterministic directed candidate within the target zone will be selected as the next

node. If there are several nodes in the priority zone, their selection probabilities are proportional to their angle distances to priority zone border. In this way, the desired line transmission property is reserved, while a certain extent of important randomness is introduced.

If solely using deterministic directed mechanism for routing in a sensor network whose mobility is predictable, a claiming message will be transmitted in a deterministic manner, except for a limited number of initial random directions. As a result, the adversary may remove some nodes in strategic locations to reduce detection probability dramatically. Moreover, transmissions of claiming messages from a cloned node's neighbors are highly correlated, which affects the protocol communication and security performance. By the probabilistic directed mechanism, those drawbacks are overcome, and the protocol performance is improved significantly, as shown in the later simulation results.

B. Analysis Memory Requirement: The RDE protocol is exceedingly memory-efficient. It does not rely on broadcasting; thus, no additional memory is required to suppress broadcasting flood.

The protocol does not demand intermediate nodes to buffer claiming messages, overcoming the major disadvantage of line-selected multicast scheme [1]. All memory requirement lies on the neighbor-list, which, in fact, is a necessary component for all distributed detection approaches. Therefore, the protocol consumes almost minimum memory.

Communication Cost: The RDE's communication cost depends on the routing parameter settings. On average, there are claiming messages sent by each observer, and each message transmits at most hops. In practice, is a constant small number, say 1 for a dense network, but is generally related to the network size . We select because there are nodes in the network, and by the line property of our protocol routing, it is very likely for any two nodes to be reachable within hops for a normal network topology. In other words, would be sufficient for messages to go across the network. The choice of is also used in [1]. For some regular network topologies and by selecting a proper target range, the average message hop is actually much smaller than resulting from the border determination. Therefore, the upper bound of communication cost in the randomly directed exploration protocol is .

Security: Because of sharing similar detection guidelines with the same cryptographic authentication primitives, the arguments for the DHT-based detection validity in Section IV-C also apply to the RDE protocol. Our proposed technique to prevent framing attack is essentially applicable to all witness-based protocols, including RDE. Relieving message-discarding and protecting witness are achieved by random initial direction and probabilistic directed transmission. By them, there is no critical location to affect message transmission, which limits the capacity of message-discarding, and every neighbor of a cloned node has similar potential to become witness so it is hard for the adversary to get rid of witness in advance.

The RDE protocol's detection probability is determined by the number of nodes that are reached when randomly drawing lines where each has a random initial angular and fixed number of nodes along this direction with the border limitation. This is a covering problem in graph theory [19]. Let denote the reachable node number; it is a function of (an initial angular), (the number of maximum hops), and (the number of the claiming messages). Therefore, for a network with nodes, the detection probability is given by

VIII. EXPERIMENTAL RESULTS FOR RANDOMLY DIRECTED EXPLORATION

We implement the randomly directed exploration protocol on the same simulation framework as the previous protocol. Since the randomly directed exploration protocol relies on a local network topology, the random graph model cannot server for its simulations. Instead, we take the unit-disk graph as the sole network scenario. We choose a constant node degree and select as the priority range of the protocol. As a result, there are an average 2.5 neighbors in the priority zone of a node.

A. Performance on Different Network Sizes

We develop the first simulation, which not only measures the randomly directed exploration protocol's performance on varying network sizes, but also verifies if our protocol design really fulfills the intention. In order to do that, we mix the three routing mechanisms as the following three groups, run simulations respectively, and compare the results for analysis.

In *group I*, only the mechanism of deterministic directed transmission is used. Consequently, all claiming messages, if not dropped by malicious nodes, go exactly hops. Next, *group II* adds network border determination into group I. As the network shape is a regular square, two different target ranges are chosen for wider and narrower border determination: and . Finally, *group III* further adds probabilistic directed transmission into group II, and indeed constitutes Algorithm 4.

For other parameter selections, ranges from 1000 to 10000, message maximal hop , the average number of claiming messages per observer , and two nodes are randomly selected as cloned.

First of all, we can see that the protocol's communication overhead is quite impressive, while its security level is satisfactory for all the network sizes. Second, in accordance with the intuition, the comparison between groups I and II indicates that the boundary determination mechanism remarkably reduces the communication cost, while the detection probability is decreased in a moderate rate. Third, compared to group II, group III only slightly increases the average number of sent messages, but considerably improves the detected probability. This proves that the probabilistic directly mechanism fulfills the design objective, and our proposed priority zone candidates probability distribution both preserves the line characteristic of directed transmission and introduces splendid randomness on message transmission.

In other words, the simulation results support our protocol design arguments, and the protocol can properly detect node clone with light communication cost.

B. Impacts of Adjusting Parameters

We may select different values for parameters, and to achieve tradeoffs between communication cost and detection probability. As the first simulation indicates, the bigger, the more hops a message travels, and the more likely cloned nodes are being caught. In this simulation, to evaluate the effects of modifying and , we still use the two values of , select, and assume two randomly selected cloned nodes.

First, we fix and gradually increase from to, carrying out the simulation. Then, we keep and let grow from 1.0 to 2.0. The results of Fig. 7(a) and (b) show that because the border determination mechanism under a certain restrains the average message hop, the increasing of after some threshold will have insignificant impact on protocol performance. In contrast, while the transmission overhead per node is in direct proportion with , the detection probability is substantially improved. Therefore, is always suitable for performance adjustment to meet different requirements on communication cost and security level.

C. Discussions

From the analysis and simulation results, the randomly directed exploration protocol outperforms all other distributed detection protocols in terms of communication cost and storage requirements, while its detection probability is satisfactory, higher than that of line-selected multicast scheme in [1]. In addition, all nodes only need to know their direct neighbors information, and then inherent routing technique delivers messages in an efficient way to cover a great range of the network. On the other hand, because the three mechanisms of determining next node all rely on transmission direction, the protocol performance in specific sensor networks will be related to network physical shape. In general, if we can obtain a rough picture about the deployment environment, we may adjust parameters and to achieve acceptable detection probability at a cost of increasing communication overhead. For example, we may use bigger to tolerate irregular network outside shape, while is set to the estimated value of maximal hop distance between two nodes. Of course, if a network topology is so distorted that there is no way to achieve line transmission solely based to nodes local knowledge, the randomly directed exploration detection becomes unsuitable, and we may use the DHT-based protocol. In addition, if security is extremely important and it is required to certainly catch cloned nodes, the DHT-based protocol should be adopted.

IX. CONCLUSION

Sensor nodes lack tamper-resistant hardware and are subject to the node clone attack. In this paper, we present two distributed detection protocols: One is based on a distributed hash table, which forms a Chord overlay network and provides the key-based routing, caching, and checking facilities for clone detection, and the other uses probabilistic directed technique to achieve efficient communication overhead for satisfactory detection probability. While the DHT-based protocol provides high security level for all kinds of sensor networks by one deterministic witness and additional memory-efficient, probabilistic witnesses, the randomly directed exploration presents outstanding communication performance and minimal storage consumption for dense sensor networks.

REFERENCES

[1] B. Parno, A. Perrig, and V. Gligor, "Distributed detection of node replication attacks in sensor networks," in *Proc. IEEE Symp. Security Privacy*, 2005, pp. 49–63.

[2] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica,"Looking up data in P2P systems," *Commun. ACM*, vol. 46, no. 2, pp. 43–48, 2003.

[3] Y. Zhang,W. Liu,W. Lou, andY. Fang, "Location-based compromise tolerant security mechanisms for wireless sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 2, pp. 247–260, Feb. 2006.

[4] S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient security mechanisms for large-scale distributed sensor networks," in *Proc. 10th ACM CCS*, Washington, DC, 2003, pp. 62–72.

[5] R. Anderson, H. Chan, and A. Perrig, "Key infection: Smart trust for smart dust," in *Proc. 12th IEEE ICNP*, 2004, pp. 206–215.

[6] M. Conti, R. D. Pietro, L. V. Mancini, and A. Mei, "A randomized, efficient, and distributed protocol for the detection of node replication attacks inwireless sensor networks," in *Proc. 8thACMMobiHoc*,Montreal, QC, Canada, 2007, pp. 80–89.

[7] B. Zhu, V. G. K. Addada, S. Setia, S. Jajodia, and S. Roy, "Efficient distributed detection of node replication attacks in sensor networks," in *Proc. 23rd ACSAC*, 2007, pp. 257–267.
[8] H. Choi, S. Zhu, and T. F. La Porta, "SET: Detecting node clones in sensor networks," in *Proc. 3rd SecureComm*, 2007, pp. 341–350.
[9] R. Brooks, P. Y. Govindaraju, M. Pirretti, N. Vijaykrishnan, and M.T. Kandemir, "On the detection of clones in sensor networks using random key predistribution," *IEEE Trans. Syst.s, Man, Cybern. C, Appl. Rev.*, vol. 37, no. 6, pp. 1246–1258, Nov. 2007.
[10] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proc. 9th ACM Conf. Comput. Commun. Security*, Washington, DC, 2002, pp. 41–47.
[11] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. CRYPTO*, 1984, LNCS 196, pp. 47–53.
[12] R. Poovendran, C. Wang, and S. Roy, *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*. New York: Springer-Verlag, 2007.
[13] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.
[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," in *Proc. SIGCOMM*, San Diego, CA, 2001, pp. 161–172.
[15] I. Stoica, R.Morris, D. Liben-Nowell, D. R. Karger,M. F.Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, Feb. 2003.
[16] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proc. IFIP/ACM Int. Conf. Distrib. Syst. Platforms Heidelberg*, 2001, pp. 329–350.
[17] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proc. 1st Int. Conf. Simulation Tools Tech. Commun., Netw. Syst. Workshops*, Marseille, France, 2008, pp. 1–10.
[18] A. Awad, C. Sommer, R. German, and F. Dressler, "Virtual cord protocol (VCP): A flexible DHT-like routing service for sensor networks," in *Proc. 5th IEEE MASS*, 2008, pp. 133–142.
[19] R. Diestel, *Graph Theory*, 3rd ed. New York: Springer, 2006.

Authors Biography

First Author : T. Shivaiah
M.Tech in CSE from JNTUH,
Malla Reddy College of Engineering and Technology,
Hyderabad

Second Author M.Vazralu
M.Tech in CSE from JNTUH,
Assoc Prof, Dept of CS&E,
Malla Reddy College of Engineering and Technology,
Hyderabad