DHT-Based Detection of Node Clone in Wireless Sensor Networks

Zhijun Li and Guang Gong

University of Waterloo, Waterloo ON N2L3G1, Canada
leezj@engmail.uwaterloo.ca, ggong@calliope.uwaterloo.ca
http://comsec.uwaterloo.ca

Abstract. Wireless sensor networks are vulnerable to the node clone attack because of low-cost, resource-constrained sensor nodes, and uncontrolled environments where they are left unattended. Several distributed protocols have been proposed for detecting clone. However, some protocols rely on an implicit assumption that every node is aware of all other nodes' existence; other protocols using an geographic hash table require that nodes know the general network deployment graph. Those assumptions hardly hold for many sensor networks. In this paper, we present a novel node clone detection protocol based on Distributed Hash Table (DHT). DHT provides good distributed properties and our protocol is practical for every kind of sensor networks. We analyze the protocol performance theoretically. Moreover, we implement our protocol in the OMNeT++ simulation framework. The extensive simulation results show that our protocol can detect clone efficiently and holds strong resistance against adversaries.

Keywords: node clone attack, distributed detection, wireless sensor networks, distributed hash table, key-based routing.

1 Introduction

Wireless sensor networks (WSNs) have gained a great deal of attention in the recent years. In general, wireless sensor networks consist of hundreds and thousands of low-cost, resource-constrained, distributed commodity sensor nodes that collaboratively collect information through wireless networks. Those sensor nodes usually scatter in the targeted area randomly, working without attendance. Some nodes perish over time, due to failure or running out of battery. To maintain or enhance the network functionality, new nodes have to be deployed into the current network.

If the targeted environment is hostile, security mechanisms against adversaries have to be taken into considerations. Among many physical attacks to sensor networks, the node clone is a severe and dangerous one. Because of production cost limitation, sensor nodes are short of tamper-resistance hardware components. 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. Then those nodes that seem legitimate are able to join the sensor network and cause severe damages. For example, the cloned nodes can occupy strategic positions and cooperatively corrupt the collected information. They can even let the adversary control the whole network. And the node clone would exacerbate most of inside attacks against sensor networks. Chan and Perrig [1] cataloged a number of attacks that can be launched using cloned nodes.

Due to reliability and balance consideration, the distributed detection approaches against the node clone attack are more desirable. Several distributed schemes have been proposed to detect cloned nodes. However, some protocols reply on an implicit assumption that every node knows the existence of all other nodes, while other protocols using geographic hash tables demand that nodes realize the outline of the network geography. Apparently, those assumptions cannot hold for many sensor networks. Inspired by Distributed Hash Table (DHT), which provides extraordinary distributed properties, we propose a novel node clone detection protocol in this paper. The theoretical analysis and the extensive simulation results show that our protocol can detect clone efficiently and holds strong resistance against adversaries.

The rest of the paper is organized as follows. First, the previous approaches against the node clone attack are discussed in Section 2. Then we define the security goals, performance metrics, network model and adversary model in Section 3. Afterwards, we detail our proposed detection protocol in Section 4, and analyze its performance in Section 5. The extensive simulation results are provided in Section 6. Finally, we conclude our work and give further research directions in Section 7.

2 Previous Protocols

We categorize node clone detection approaches into two categories. One is centralized, in which there exists a central and powerful party (the base station at most of time) which is responsible for receiving reports and making decisions of node clone. The other is distributed, in which all nodes cooperatively process information and detect node clone.

2.1 Centralized Detection

In a straightforward centralized detection approach, each node sends a list of its neighbor nodes and their claimed locations to a base station. If the base station finds that there are two far distant locations for one node ID, then the node clone must have occurred. The base station simply broadcasts through the whole network to expel the cloned nodes.

SET, proposed by Choi, Zhu, and Porta [2], manages to reduce the communication cost of the preceding approach by computing set operations of exclusive subsets in the network. First, SET launches an exclusive subset maximal independent set (ESMIS) algorithm which forms exclusive unit subsets among one-hop neighbors in a distributed way. As a result, each node belongs to one and only one disjointed subset which is controlled by a randomly decided leader. Then those subsets, in the basic scheme, are transmitted by leaders to the base station such that it can construct all nodes locations and detect clones. Since the subset division procedure eliminates redundancy in the node location reports, SET lowers the communication cost. However, in order to prevent malicious nodes in the ESMIS algorithm, an authenticated subset covering protocol has to be performed, which increases the communication overload and complicates the detection procedure. SET also employs a tree structure to compute non-overlapped set operations and integrates interleaved authentication to prevent unauthorized falsification of subset information during forwarding. Randomization is used to further make the exclusive subset and tree formation unpredictable to an adversary.

Brooks *et al.* [3] proposed a clone detection protocol in the context of random key predistribution [4]. Its assumptions and application scenarios are quite different from other approaches. In fact, it is detecting cloned keys rather than cloned nodes. The basic idea is that the keys employed in the random key predistribution scheme should follow a certain pattern, and those keys whose usage exceeds a threshold can be thought of as suspicious. In the protocol, every node reports its keys to a base station and then the base station uses an abnormality-based intrusion-detection-like statistical method to find cloned keys. A big problem in this kind of approaches is the high false negative and positive rates. Furthermore, the authors do not address how to assure malicious nodes to honestly report their keys.

As mentioned in [5], the centralized approaches may create a single-point of failure. Besides, the nodes surrounding the base station would suffer an undue communication burden that may shorten the network's life expectancy. In many cases, a distributed and balanced detection scheme is more desirable.

2.2 Distributed Detection

Node-to-node broadcasting [5] is a quite practical, effective way to distributively detect the node clone. Every node collects all of its neighbors identities along with their locations, and broadcasts to the network. When a node receives a broadcasted message from others, it compares those nodes listed in the message with its own neighbors and revokes neighbor nodes that have collision locations. The main problem in this approach is its high communication overload.

Parno, Perrig, and Gligor [5] provided two probabilistic inspector-based protocols. Randomized multicast scheme distributes node location information to randomly-selected inspectors, exploiting the birthday paradox to detect cloned nodes, while line-selected multicast scheme uses the topology of the network to detect replication, that is, in addition to inspector nodes, the nodes within the multicast path check the node clone. The storage consumptions in the both schemes are too high to afford. And the communication cost in the randomized multicast is similar to that in the node-to-node broadcasting. Thus there is no benefit to use the randomized multicast. For the procedure of choosing random inspectors, those schemes both imply that every node is aware of all other nodes' existence, which is an impractical assumption and limits their applicability in sensor networks.

A Geographic Hash Table (GHT) [6] maps a key into a geographical coordination. Based on GHT, Zhu *et al.* [7] proposed a localized multicast to detect the node clone. In the system, the inspector nodes for a examined node are chosen from nodes that are located within a geographical limited region (named cell) which is determined by a GHT hash result of the node identification. They presented two variants of localized multicast: single deterministic cell, in which only one unique cell is determined for one node, and parallel multiple probabilistic cell, in which the location claim is mapped and forwarded to multiple deterministic cells with various probabilities. Conti *et al.*[8] proposed another GHT-based clone detection approach. Those approaches rely on the nodes awareness of the general deployed geography of sensor networks. This prerequisite may hold in some circumstances, but cannot be guaranteed in the general cases.

In a word, the previous proposals of distributed clone detection cannot be practically applied to most of sensor networks. Innovative approaches are demanded.

3 Goals and Models

3.1 Security Goals and Performance Metrics

In this paper, we deal with the node clone attack, that is, the attempt by an adversary to add one or more nodes to the sensor networks by cloning captured nodes. The proposed scheme should detect the occurrence of the attack with an overwhelming probability. In addition, the scheme should make genuine nodes to stop communicating with clones nodes, expelling all clones off the network.

As discussed before, we would like to detect the attack in a distributed way. The proposed scheme should not count on a centralized, powerful base station, though a faithful base station might be of help. Similar to other distributed approaches, inspector nodes are used to detect clone. If an inspector successfully finds a clone, it becomes a *witness*. As more witnesses improve the resilience against the adversary, the number of witnesses represents a major security measurement for the scheme.

Usually, energy is the most valuable resource in wireless sensor networks. Communication consumes at least one order of magnitude power than any of other operations [9]. Therefore, we evaluate the communication overload as the main performance of the scheme. For simplicity, we use the average number of sent messages per node to measure the protocol's communication cost. On the other hand, sensor nodes are equipped with a limited amount of memory; thus any schemes requiring high storage would be considered as impractical. The memory requirement would be another performance metric for efficiency. Besides, the scheme should consume the energy and memory in a balanced way. There should not be hot nodes which would be buffer-overflowed or die away soon.

3.2 Network Model

We consider a homogeneous sensor network consisting of N resource-constrained sensor nodes. The number of nodes N can be huge. Therefore, any protocol should scale to the network size. Sensor nodes operate without supervision at most of time, and they can function correctly in a dynamic network, where new nodes are added, or old nodes disappear. In addition to neighbor nodes, sensor nodes only know some of other distant nodes. Nodes do not realize the network geographic outline. The average degree of node, that is, the number of its neighbors, denoted by d, varies with networks.

Like in the previous distributed detection approaches [5,6,8], we assume that an identity-based public-key cryptography [10] is employed in the sensor network system. Before deployment, each legitimate node is assigned with 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 identity-base cryptosystem. Therefore, a node cannot lie to others about its ID. Moreover, recipients can verify messages signed by a node using the identity-based key. Let K_α and K_α^{-1} denote the public and private keys of node α respectively, and $\{M\}_{K_\alpha^{-1}}$ represent the signature of M signed by node α.

Compared to traditional public key cryptosystems, which can satisfy the requirements of our system as well, the identity-based systems alleviate the heavy burden of public key certificates. Our system, as described afterwards, only uses this primitive for authentication rather than for encryption. Therefore, it is not necessary for the identity-based cryptosystem to be pairing-based. Though the usage of public-key in sensor networks might be doubted several years ago, there are more and more WSNs security protocols relying on public key systems. The applicable implementation of such cryptosystems in typical sensor nodes platform have been addressed in [11,12,13].

We also assume that every sensor node can determine its position L via a secure localization mechanism. A number of those mechanisms have been proposed, which can be referred to in [14]. During a round of node clone detection, we suppose the sensor network to be stationary; so a collision of locations for one node ID indicates a clone.

There might be or not be a powerful base station in our modeled network, but there does exist a trusted role named *initiator* that is responsible for initiating a round of distributed detection. Otherwise, an adversary can readily launch a denial-of-service (DoS) attack to the system by keeping launching detection procedures and exhausting nodes energy. The initiator can be the base station if one exists, or can be selected among all nodes via a distributed leader election, such as one in [15].

3.3 Adversary Model

We consider a threat model in which sensor nodes are deployed in a hostile environment and are subject to being captured and completely controlled by an adversary, but the adversary only can compromise a small portion of sensor nodes. If the adversary gains control of the majority of legitimate nodes, all security mechanisms will ultimately fail. The adversary can use the compromised nodes to clone many nodes and deploy the replicas in places that are intelligently decided. However, we assume that each cloned node has at least one integrity neighbor.

The adversary definitely wants to conceal the existence of clone. In our settings, the adversary is allowed to interfere with the detection algorithm in three ways. First, the cloned nodes may not participate the regular detection procedures. Second, the cloned nodes may drop or manipulate the reporting messages which they forward. Lastly, the adversary can capture some nodes accordingly, but it would take some time, and the total number of nodes that an adversary can compromise is limited.

4 Proposed Protocol

The main idea in our protocol is to make use of the Distributed Hash Table (DHT) mechanism to build a decentralized, key-based caching and checking system, which can detect the node clone. This kind of systems share many common characteristics with the P2P indexing systems, which greatly benefit from DHT in the recent years. Indeed, DHT enables nodes to distributively construct an upper *overlay network* upon the sensor network and provides the key-based routing in the overlay network. A message associated with a key would be forwarded through the overlay network to the destination node, which is determined by the key. The source node does not need to know which the destination is — the DHT key-based routing would take care of transportation.

The information about the ID and the location of every node would be claimed by its neighbors for clone detection. In this sense, the neighbors of the node are its *observers*. At the beginning of a round of clone detection, the initiator broadcasts the action command along with a random *seed*. Then every observer starts to create a claim message for each of its neighbors and send the message with probability p. The introduction of the claim probability p intends to reduce the communication overload in case of a high node degree network. In our system, the DHT key that determines the routing and destination is the hash value of concatenation of the seed and the claimed node's ID. The message will be finally forwarded 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 like inspectors to improve resilience against the adversary in an efficient way. In order to detect clone, an inspector maintains a *cachetable* of dynamic number of records, each of which contains an ID-location pair. The cachetable contributes to the main memory cost of our protocol; therefore we measure the protocol's storage cost by the average size of cachetable. Any witness that detects clone would broadcast the evidence such that the whole network expels the cloned nodes.

4.1 Distributed Hash Table

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 the record associated with a specific key. DHT distributes responsibility for maintaining the mapping from keys to records among the nodes in an efficient, balanced way, which allows DHT to scale to extremely large networks and suitable for an infrastructure of distributed node clone detection.

There are different types of DHT proposals, such as CAN [16], Chord [17], Pastry [18]. Generally, CAN has worse performances than others in terms of communication cost and scalability, and it is rarely employed in real systems. Chord might be the most widely implemented DHT, and we choose Chord to demonstrate our protocol. Our protocol can easily migrate to be Pastry-based and get similar security and performance results.

The principle behind Chord [17] is to form a massive virtual ring in which every node occupies one point, owning a segment of the periphery. A hash function is used to map an arbitrary input into an m-bit space, which can be conceived as a ring. Each node is assigned with a Chord position upon joining the network. Practically, the Chord position can be the hash value of the node's MAC address. All N nodes divide the ring into N segments using their Chord positions. Likewise, the key of a record is the result of the hash function. Every node is responsible for one segment which ends at the node's Chord position, and all records whose keys are located at that segment would be transmitted to and stored in that node. For example, if the Chord positions of nodes α, β, and γ are 21, 96, and 182 (for $m = 8$) respectively, then β is responsible for keys 22–96, γ for keys 97–182, and α for keys 183–21.

4.2 Protocol Details

As a preparation for the following detection procedures, all nodes cooperatively build a Chord overlay network over the sensor network. The construction of the overlay network is independent of node clone detection. Consequently, nodes hold the information of their direct predecessor and successor in the Chord ring. In addition, every node maintains a *finger table* of size $t = O(\log N)$ to facilitate a binary-tree search for key-based routing. To be specific, the finger table for a node with Chord position x contains information of t nodes that are respectively responsible for the t keys: $(x + 2^{m-i}) \mod 2^m$ for $i \in [1, t]$. Moreover, each node caches information of its g consecutive successors in its *successors table*. Many Chord systems utilize this kind of cache mechanisms to reduce the communication cost and enhance systems robustness. More important in our protocol, the facility of successors table can help increase the average number of witnesses at a little extra memory cost.

One round of node detection consists of three processes as follows.

Process 1: Initiate a round of detection
The initiator uses a broadcast authentication scheme to issue the message including the action order and a random round seed.

$$M_{command} = ACTION, seed, \{ACTION \mid\mid seed\}_{K^{-1}_{\text{initiator}}}$$

Process 2: Claim neighbors information
Upon receiving the command message, a node verifies the message signature. If it is valid, the node caches the seed and starts to operate as an observer, which

```
routine handlemessage($M_{\alpha 4\beta}$)
$key = H(seed \mid\mid ID_\beta)$
if $key \in (predecessor, x]$
    inspect($M_{\alpha 4\beta}$);
    return NIL;
for $i = 1$ to $g$
    if $key \in (x, successors[i]]$
        inspect($M_{\alpha 4\beta}$);
        return $successors[i]$;
for $i = 1$ to $t$
    if $key \in [finger[i], x)$;
        return $finger[i]$;
return $successors[g]$;
```

```
routine inspect($M_{\alpha 4\beta}$)
verify the signature of $M_{\alpha 4\beta}$;
if $ID_\beta$ found in cachetable
    if $ID_\beta$ has two distinct locations
        become a witness;
        broadcast the evidence;
else
    buffer $M_{\alpha 4\beta}$ into cachetable;
return;
```

Fig. 1. Algorithm for handling a message, where x is the current node's Chord position; $finger[i]$ is the first node on the ring that succeeds key $((x+2^{m-i}) \mod 2^m), i \in [1,t]$; $successors[i]$ is the next ith successor, $i \in [1,g]$

generates a claim message for each neighbor and forwards the message via the overlay network, with probability p. The claim message by observer α for node β is defined as follows.

$$M_{\alpha 4\beta} = ID_\beta, L_\beta, ID_\alpha, L_\alpha, \{ID_\beta \mid\mid L_\beta \mid\mid ID_\alpha \mid\mid L_\alpha\}_{K_\alpha^{-1}}$$

Process 3: Handle a message
A message will be forwarded to the destination node via several Chord intermediate nodes in the overlay networks. Only those nodes in the overlay network layer (i.e. the source node, Chord intermediate nodes and the destination node) need to handle a message, while other nodes in the path just route the message to the temporary target. When receiving a message $M_{\alpha 4\beta}$, a node calls routine *handlemessage*($M_{\alpha 4\beta}$), which is described by pseudo-code in Fig. 1, to process the message. If the routine returns NIL, then the message has arrived at its destination. Otherwise, the message will be forwarded to the node of the returned ID, through the underlying multi-hop sensor network.

During handling a message, the node would act as an inspector if it is the destination node or one of the g predecessors of the destination. Since almost all messages related to a same claimed node ID would go through one of the g predecessors to get to the destination, those g nodes would have a much higher probability to detect a clone for that claimed node ID than randomly selected inspectors. Therefore, this criterion to decide inspectors can increase the average number of witness at a little extra memory cost.

The subroutine *inspect*($M_{\alpha 4\beta}$) in routine *handlemessage*($M_{\alpha 4\beta}$) deals with the clone detection. If detecting a clone, which means there exist two messages $M_{\alpha 4\beta}$ and $M_{\alpha' 4\beta'}$ where $ID_\beta = ID_{\beta'}$ but $L_\beta \neq L_{\beta'}$, the witness node would broadcast an evidence message, defined as follows, to notify the whole network.

$$M_{evidence} = M_{\alpha 4\beta}, M_{\alpha' 4\beta'}$$

Notice that messages $M_{\alpha 4\beta}$ and $M_{\alpha' 4\beta'}$ are authenticated by observers α and α' respectively. Therefore, the witness does not need to sign the evidence message. If a node tries to launch a DoS attack by broadcasting a faked evidence message, the next node receiving it can detect the malicious behavior by verifying the signatures of $M_{\alpha 4\beta}$ and $M_{\alpha' 4\beta'}$ before forwarding to next node.

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 cloned nodes IDs may be maintained by every node.

5 Analysis

5.1 Communication Cost

We denote the average path length between two random nodes by l, which varies from $O(\log N)$ to $O(\sqrt{N})$, relying on the underlying sensor network. According to the Chord's properties [17], the average Chord-hop of a message, that is, the number of forwarding in the Chord overlay network, is $c \log N$, where c is a constant number, usually less than 1. Therefore, the average path hop length of a message is $cl \log N$. There are npd claim messages in all for a round of detection. Thus the communication cost of our protocol would be $npdcl \log N$. Since the p, d, and c are constant, the asymptotic communication cost is between $O(N \log^2 N)$ to $O(N\sqrt{N} \log N)$. In comparison, the node-to-node broadcasting would require the communication cost of $O(N^2)$.

5.2 Storage Cost and Number of Witnesses

Suppose all nodes, including clone ones, abide by the detection protocol, we analyze the average size of cachetable and the average number of witnesses theoretically.

In our protocol, a claim message would be forwarded to a deterministic destination node, which should always be a witness if there are clones, and every node, on average, holds one record in its cachetable for a claimed node ID as the destination. In addition, the g predecessor nodes of the destination can act as inspectors, and thus may hold a copy of the record too, and have potential to become the witnesses.

Those g predecessors, defined as E_i for $i \in [1, g]$, together with the destination, occupy a part of periphery of the Chord ring and partition it into g consecutive segments. Considering the good randomness of Chord systems, the length proportions of g segments in the part of periphery are randomly distributed. We denote the probability of one specific message would go through predecessor E_i by P_i. According to the algorithm in Fig. 1, a message will pass one of the g predecessors before finally reaching the destination with probability $\frac{N-1}{N}$ (The probability of the source being the destination is $\frac{1}{N}$). Therefore, the g probabilities P_i are randomly distributed under the condition of $\sum_{i=1}^{g} P_i = \frac{N-1}{N}$.

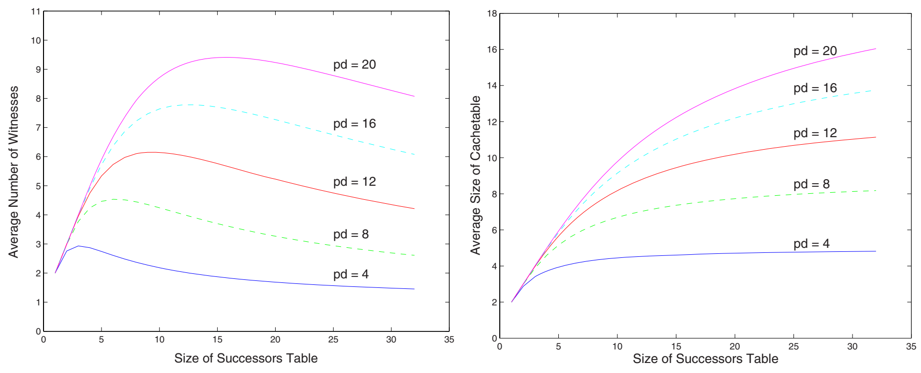


Fig. 2. Number of Witnesses and Storage Cost in the ideal case, where p is the claim probability, and d is the average node degree

On average, there are pd messages claiming location of each node. The average size of cachetable can be computed by

$$s = 1 + \sum_{i=1}^{g} \left(1 - (1 - P_i)^{pd}\right) \quad . \tag{1}$$

Suppose there are two cloned nodes in the network, then the average number of witnesses can be computed by

$$w = 1 + \sum_{i=1}^{g} \left(1 - (1 - P_i)^{pd}\right)^2 \quad . \tag{2}$$

In the ideal case, where the P_i's are uniform and we can assume $\frac{N-1}{N} \simeq 1$, the average size of cachetable would be

$$s' = 1 + g\left(1 - \left(1 - \frac{1}{g}\right)^{pd}\right) \quad ; \tag{3}$$

and the average number of witnesses for two clones would be

$$w' = 1 + g\left(1 - \left(1 - \frac{1}{g}\right)^{pd}\right)^2 \quad . \tag{4}$$

Fig. 2 depicts the plots of the average number of witnesses and the average size of cachetable as functions of the size of successors table with various values of pd in the ideal case.

5.3 Security Analysis

The introduction of the identity-based system provides the identity authentication and message authentication for our protocol. As a result, the adversary cannot fake clone nodes' ID; neither can he modify messages signed by integrity nodes. A cloned node cannot lie to its observers about its location since the faked location would be far deviated from the communication range of the observers.

Technically, the hash functions used in DHT do not need to be cryptographic hash functions. In practice, the cryptographic ones are employed in the DHT systems because of their well uniformly random distribution of outputs. Indeed, our protocol requires a collision-resistant cryptographic hash function, such as SHA-1, because it would limit the adversary's abilities when it is used in $H(seed \,||\, ID)$.

The adversary may want to comprise the witnesses to thwart detection. However, there are $g + 1$ potential witnesses that are geographically randomly distributed in the network. Determining and capturing all the witnesses would be troublesome for the adversary. Moreover, those witnesses change in different rounds. Consequently, the adversary cannot stop the detection by trying to capture a few witnesses.

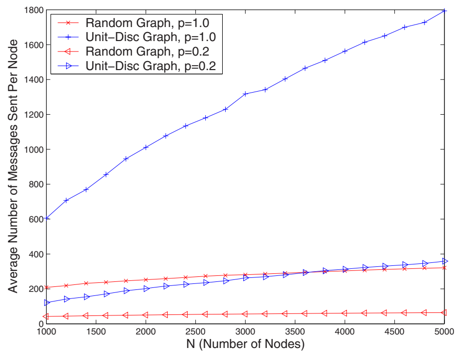
The cloned nodes are allowed to drop claim messages that pass through them. Our protocol is resilient against this countermeasure, due to the characteristic of full distributiveness and balance. If there are only a few clones nodes, the impact of this malicious action would be insignificant. As the number of cloned nodes increases, the more claim messages would guarantee sufficient number of witnesses. The simulations in the next section will show this result.

6 Simulations

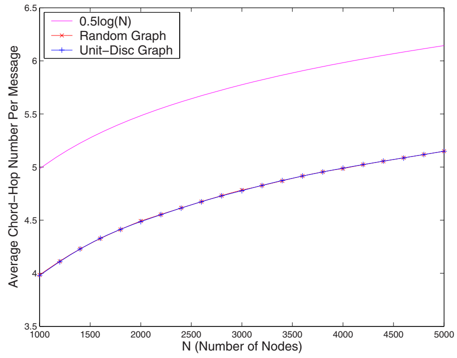
To evaluate the performance, we implement our protocol and run simulations in the OMNeT++ framework [19], because of its good scalability and high efficiency, which allows simulating large networks.

6.1 Settings

We run simulations in two network scenarios. The first is an abstract network following a *random graph* model. A random graph is a graph that is generated by starting with a set of N vertices and adding edges between them at random. In the Erdös-Rényi model [20], a random graph can be denoted by $G(N, d)$, in which every possible edge independently occurs with probability $\frac{d}{N}$. After Eschenauer and Gligor introduced Random Graph into wireless sensor networks in their classic paper [4], it is quite popular in sensor networks and builds foundation for many WSNs security protocols [21]. We use Random Graph as an ideal random network scenario. The other is the realistic *Unit-Disc Graph*, in which nodes are uniformly deployed in a 10000×10000 square and nodes follow the standard unit-disc bidirectional communication model. We adjust the node communication range such that the average node degree keeps the approximate d. This kind of simulation scenario has been used in [5] and many other literatures. Taking advantage of the high modularization of the OMNeT++ framework as well as the Chord's properties, the implementation of our detection protocol is well encapsulated and actually independent of the underling network topology. It is effortless to add new network scenarios and run simulations.



(a) Communication Overload



(b) Chord-Hop Per Message

Fig. 3. Simulation Results in Experiment One

We choose SHA-1 as the hash function in the implementation. The following fixed parameters are used in the simulations: the size of finger table $t = 16$, the size of successors table $g = 16$, and the node degree $d = 20$. Two different values of claim probability p are used as 1.0 and 0.2. Overall, we have four settings: Random Graph with $p = 1.0$, Unit-Disc Graph with $p = 1.0$, Random Graph with $p = 0.2$, Unit-Disc Graph with $p = 0.2$.

6.2 Protocol Performance

We design and run Experiment One to measure the protocol performance. For each of the four settings, we launch 21 runs of simulations, in which the network size N ranges from 1000 to 5000, with a step of 200. Every run performs 20 rounds of detection, in each of which a random seed is generated and two nodes are randomly chosen to set the same ID, that is, those two are cloned ones. In Experiment One, we make the cloned nodes to obey the detection protocol.[1]

We measure the communication overload by the average number of messages sent or received by a node in one round, which is depicted in Fig. 3a. In the graph, the communication overloads in the simulations of $p = 0.2$ are exactly the one-fifth of those of $p = 1.0$ with the same other parameters. The distinction between simulation results of Random Graph and those of Unit-Disc Graph results from the difference of the average path hop l for a pair of random nodes in those two network scenarios. In fact, the average Chord-hop per message in our system, depicted in Fig. 3b, is independent of the network scenarios. The Chord-hop results for $p = 1.0$ and $p = 0.2$ are same. Indeed, our detection protocol can run in all network topologies, like "Thin H", "Thin Cross", "S", "Large H", "L", and "Large H", which are used in the simulations of [5]. The communication

[1] Our simulations show little variance for performance even if we let clone nodes discard messages, since the number of clone nodes in this experiment is much less than N.

Table 1. Comparison of Witnesses Number and Storage Cost in Experiment One

Metrics	$pd = 20$			$pd = 4$		
	Ideal Case	Random Graph	Unit-Disc Graph	Ideal Case	Random Graph	Unit-Disc Graph
Number of Witnesses	9.41	6.82	6.64	1.83	2.17	2.19
Size of Cachetable	12.60	8.77	8.79	4.64	4.03	4.07

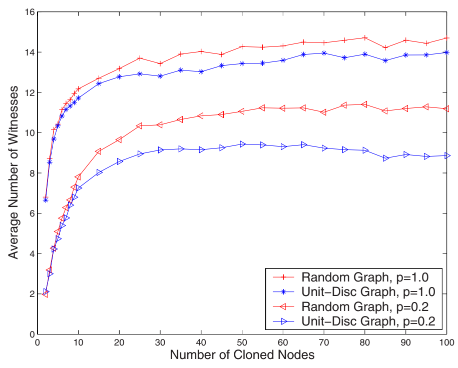
overload of our protocol for different network topologies is proportional with the average path hop l.

Since the average witness number s and the average cachetable size w are independent of the network size N if N is sufficiently large, we average the values in the 21 runs, and list the results in the Table 1. For comparison, we also show the values of witness number s' and cachetable size w' in the ideal case, computed by Equation 4 and Equation 3 respectively.

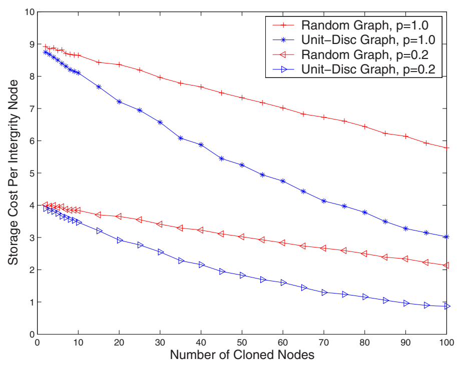
6.3 Resilience against Message-Discarding by Clones

We develop Experiment Two to evaluate the protocol's performance if cloned nodes are allowed to discard messages. Still using the four settings, we test with one network size $N = 1000$. The cloned node number c varies from 2 to 100. For each run, we repeat 200 rounds of node detection, in each of which a seed is randomly generated and c nodes are randomly chosen as clones. Fig. 4 depicts the simulation results about the average number of witnesses and the average size of cachetable for integrity nodes.

From Fig. 4a, we can see that our protocol shows strong resilience against message-discarding of cloned nodes. Even if there are 10% nodes that maliciously discard messages, the number of witnesses is still pretty high. On the other hand,



(a) Number of Witnesses



(b) Storage Cost Per Integrity Node

Fig. 4. Number of Witnesses and Storage Cost Per Integrity Node versus different number of cloned nodes in the Experiment Two, where $N = 1000$

if it is very likely for the adversary to deploy many clones out of one ID, we can safely use small claim probability p for saving communication overload, without degrading the security level (the average witness number) dramatically.

7 Conclusion and Future Work

Sensor nodes lack tamper-resistant hardware and are subject to the node clone attack. In this paper, we present a novel clone detection protocol based on distributed hash table. A Chord overlay network is built upon the sensor network, and provides the key-based routing, caching, and checking facilities for our protocol. The one deterministic witness and the additional memory-efficient, potential witnesses assure the good security properties. Furthermore, the mechanism of random round seeds limits the adversary's ability to conceal the clone by compromising witness nodes. The extensive simulation results of our protocol show the good performance.

One way to improve the communication performance of our detection protocol is replacing the Chord overload network with some specific DHT implementations on sensor networks. A number of such protocols, such as Scalable Source Routing [22], Virtual Ring Routing [23], and Virtual Cord Protocol [24], have been proposed. The combination of our protocol with those DHT proposals would be one of future research directions.

Most of the proposed node detection approaches, including ours, require nodes to know their locations. Proposals that do not rely on this requirement would be very interesting and more practical. Innovative detection principles have to be developed. This would be a part of our future work.

Acknowledgment

The research is supported by NSERC Strategic Project Grants.

References

1. Chan, H., Perrig, A.: Security and privacy in sensor networks. Computer 36(10), 103–105 (2003)
2. Choi, H., Zhu, S., La Porta, T.F.: SET: Detecting node clones in sensor networks. In: Third International Conference on Security and Privacy in Communications Networks and the Workshops (SecureComm 2007), pp. 341–350 (2007)
3. Brooks, R., Govindaraju, P.Y., Pirretti, M., Vijaykrishnan, N., Kandemir, M.T.: On the Detection of Clones in Sensor Networks Using Random Key Predistribution. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 37(6), 1246–1258 (2007)
4. Eschenauer, L., Gligor, V.D.: A key-management scheme for distributed sensor networks. In: Proceedings of the 9th ACM conference on Computer and Communications Security, Washington, DC, USA, pp. 41–47 (2002)
5. Parno, B., Perrig, A., Gligor, V.: Distributed Detection of Node Replication Attacks in Sensor Networks. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 49–63 (2005)

6. Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R., Shenker, S.: GHT: a geographic hash table for data-centric storage. In: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (WSNS), Atlanta, Georgia, USA. ACM, New York (2002)
7. Zhu, B., Addada, V.G.K., Setia, S., Jajodia, S., Roy, S.: Efficient Distributed Detection of Node Replication Attacks in Sensor Networks. In: Twenty-Third Annual Computer Security Applications Conference (ACSAC), pp. 257–267 (2007)
8. Conti, M., Pietro, R.D., Mancini, L.V., Mei, A.: A randomized, efficient, and distributed protocol for the detection of node replication attacks in wireless sensor networks. In: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing, Montreal, Quebec, Canada. ACM, New York (2007)
9. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. IEEE Communications Magazine 40(8), 102–114 (2002)
10. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
11. Liu, A., Ning, P.: TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks. In: International Conference on Information Processing in Sensor Networks (IPSN 2008), pp. 245–256 (2008)
12. Szczechowiak, P., Oliveira, L., Scott, M., Collier, M., Dahab, R.: NanoECC: Testing the Limits of Elliptic Curve Cryptography in Sensor Networks. In: Verdone, R. (ed.) EWSN 2008. LNCS, vol. 4913, pp. 305–320. Springer, Heidelberg (2008)
13. Oliveira, L.B., Scott, M., Lopez, J., Dahab, R.: TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. In: 5th International Conference on Networked Sensing Systems, INSS, pp. 173–180 (2008)
14. Poovendran, R., Wang, C., Roy, S.: Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks. Springer, Heidelberg (2007)
15. Chen, G., Branch, J.W., SzymanskiChen, B.K.: Local leader election, signal strength aware flooding, and routeless routing. In: Proceedings of 19th IEEE International of Parallel and Distributed Processing Symposium, IPDPS 2005 (2005)
16. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, San Diego, California, United States. ACM, New York (2001)
17. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Transactions on Networking (TON) 11(1), 17–32 (2003)
18. Rowstron, A.I.T., Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, p. 329. Springer, Heidelberg (2001)
19. OMNeT++ Community Site, `http://www.omnetpp.org/`
20. Erdös, P., Rényi, A.: On the evolution of random graphs. Bulletin of the Institute of International Statistics 38, 343–347 (1961)
21. Xiao, Y., Rayi, V.K., Sun, B., Du, X., Hu, F., Galloway, M.: A survey of key management schemes in wireless sensor networks. Computer Communications 30(21-12), 2314–2341 (2007)

22. Fuhrmann, T.: Scalable routing for networked sensors and actuators. In: Proceedings of IEEE SECON 2005, pp. 240–251 (2005)
23. Caesar, M., Castro, M., Nightingale, E.B., O'Shea, G., Rowstron, A.: Virtual ring routing: network routing inspired by DHTs. In: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM 2006), Pisa, Italy. ACM, New York (2006)
24. Awad, A., Sommer, C., German, R., Dressler, F.: Virtual Cord Protocol (VCP): A flexible DHT-like routing service for sensor networks. In: 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS 2008), pp. 133–142 (2008)