Imposter Detection in Mobile Wireless Sensor Networks

Ebrahim A. Alrashed and Mehmet Hakan Karaata

***Abstract*—Mobile wireless sensor networks (MWSNs) are wireless networks of small sensors moving around in a specified coverage area, conveying their readings and data to static or mobile base stations. In contrast to a static wireless sensor network where the fixed node position can be used to authenticate a sensor node, the changing position of a node in a MWSN cannot be used in the authentication process. This can lead to imposters posing as legitimate nodes anywhere in the network. An imposter can use the identity of a legitimate node to communicate within the network and eavesdrop on confidential communication or to send false information. In this paper, we propose a novel scheme which employs nonce-values and blackout-time mechanisms to detect imposters in the network. The proposed scheme employs a quarantining mechanism to quarantine the detected imposters, and a node-restoration mechanism to appropriately authenticate and restore quarantined nodes into the network.**

***Index Terms*—Imposter detection, mobile sensor networks, wireless network security, wireless networks.**

I. Introduction

A mobile wireless sensor network (MWSN) is a wireless network of small, mobile sensors deployed in a specified coverage area to sense physical conditions such as temperature, pressure etc. Sensor nodes convey the sensed information to base stations or sink nodes which may also be mobile. The movement of sensor nodes can be controlled in such a way that there is an optimal distribution of sensors throughout the coverage area [1], [2]. This can help in covering the entire monitored area using lower number of sensors compared to a stationary Wireless Sensor Network (WSN), where the sensor nodes are stationary [3]. Alternatively, nodes can be made to move towards a particular region of interest to increase number of nodes in that area, e.g., the site of an explosion, thus providing improved resolution of detected events [4].

In a WSN, where the sensor nodes are stationary, the sink or other sensor nodes can use node position to verify a node's identity. The base station in such a network can ascertain the authencity of sensor nodes using algorithms such as the intrusion detection technique proposed by Bhuse and Gupta [5]. In a MWSN, how-ever, nodes are in constant motion and it becomes difficult to verify whether a particular sensor node is legitimate or an imposter, based on its location. As a sensor node moves within the coverage area, it needs to establish connections with its neighboring nodes in order to route data through them towards the sink node. This requires a strong authentication protocol which enables the sensor node to verify the identity of a neighboring sensor node as a legitimate entity in the network before communicating with that node. If the identity of the neighboring sensor node is verified to be legitimate, the sensor node in question may choose to route its data through the neighboring node.

It is possible that an adversary will assume the identity of a legitimate node and try to communicate with its neighboring nodes. An adversary is a malicious entity controlled by an attacker. We refer to an adversary as an imposter if it uses the identity of a legitimate sensor node to communicate with the sink and with other legitimate sensor nodes. By assuming a false identity, an imposter can send misleading information or replay old data packets, which could trigger an undesirable series of events in the network e.g., wrong alarms signaled by an event detection system [6]. The imposter could also portray multiple node identities causing a Sybil attack [7].

In this paper, we propose a novel scheme to detect imposters in a MWSN using blackout-time and nonce-value mechanisms. If the sink identities a sensor node to be an imposter, it prevents the detected node from communicating with other sensor nodes by means of a quarantine mechanism. Nodes that have been quarantined can be restored into the network by means of a node-restoration mechanism. The scheme also restricts malicious activity by frequently changing the decryption keys within the network.

The paper is organized as follows. In Section II, related work on imposter identification in wireless sensor networks is discussed. In Section III, the proposed scheme is presented and analyzed. In Section IV, input test cases and simulation results are discussed and evaluated against the chosen metrics of interest. Finally, in Section V, conclusions and future scope of work are presented.

II. Related Work

The problem of imposter detection is to detect if one or several nodes are using the identity of a legitimate node to infiltrate the net-work. This has been studied in literature under the name node replication attacks. Detection of sensor node replication could be radio-based or network-based. Radio-based detection relies on a physical characteristic (the radio fingerprint) such as signal strength, to authenticate legitimate nodes, and subsequently detect imposters in the network [5], [8], [9]. Hall *et al*. use radio frequency fingerprinting to authenticate nodes and detect imposters by analyzing the transient portion of the received signal [8]. Bhuse and Gupta discuss an anomaly intrusion technique which uses Received Signal Strength Indicator (RSSI) to help sensor nodes detect an intruder [5]. Such techniques are outside the realm of autonomous network intrusion detection, thus not feasible to be used in unattended and geographically spread WSNs. Thus we focus on network-based detection

Manuscript received December 30, 2013; revised June 2, 2014.

The authors are with the Department of Computer Engineering, Kuwait University, Kuwait (e-mail: dr_ebrahim@mac.com).

DOI: 10.7763/IJCCE.2014.V3.364

techniques.

Network-based detection techniques differ for stationary WSNs and MWSNs. In WSNs, each sensor node is associated with a unique deployment position, and if one node ID is associated with several locations, then this indicated a node replication. Network-based techniques to combat node replication attacks have been classified as centralized and distributed solutions. Centralized solutions heavily rely on a powerful base station which is responsible for data collection regarding imposters and decision making. For example, Parno et al. describe a centralized detection scheme in which each node sends a list of its neighbors and the physical location claimed by these nodes (location claim) to the base station, which then examines each neighbor list, looking for replicated sensor nodes [10]. This scheme is a basic approach to centralized detection of node replication, suitable only for stationary WSNs.

Distributed solutions utilize what is known as the claimerreporter-witness framework, as proposed by Parno *et al.* [10]. These solutions utilize location information for a sensor node being stored at one or more witness nodes in the network. When joining the network, nodes are required to send their location information (location claim) to witness nodes, which then detect replicas of a node, if they receive more than one location for a single sensor node. Most existing detection techniques assume stationary WSNs and adopting these techniques to MWSNs is not easy. For example, the claimerreporter-witness framework for stationary WSNs cannot be easily adopted in MWSNs for the following reasons. Nodes in MWSNs are in constant movement, and thus they are not associated with a unique location. Also, it becomes difficult to route any location claim from a certain reporter to a certain witness because of the continuously changing network topology. Also, issues like how to denote a witness (which needs to be linked to a certain physical location) become difficult to address.

However, few existing solutions have been proposed for MWSNs. Ho *et al*. proposed a scheme which utilizes a sequential probability ratio test [11]. A claimer node locally broadcasts its location claim to its neighbors from time to time. This information is collected by the neighboring nodes and sent to the base station, which calculates the node speed as a function of the location of the node and time of reporting. If the calculated node speed exceeds the system-configured maximum speed for a sensor node, then the presence of a replica is detected. This technique has several drawbacks including the requirement for an accurate measurement of location, which necessitates a dynamic and precise localization system along with tight time synchronization, both of which are not affordable in the current generation of WSNs.

Another approach to node replica detection was discussed by Conti *et al*. where disappearance from the network is associated with capture by an adversary [12]. In this technique, each sensor node observes the time it encounters other sensor nodes and if a particular sensor node has not been encountered for a sufficiently long period of time, an alarm is raised. Sensor nodes can also update their meeting times by exchanging timing information with other sensor nodes. However, this technique suffers from a major drawback that a malicious node can spread incorrect information that an absent sensor node is still present in the network. Also, this technique requires that each sensor node be able to flood the network with an alarm, which may not be possible.

III. Proposed Scheme

In this section, we describe the proposed imposter detection scheme. First, we make a set of assumptions with regard to the net-work model and then propose a novel imposter detection scheme to operate within the previously defined network model.

A. The Network Model

We assume a mobile wireless sensor network consisting of a large number of mobile sensor nodes deployed in a rectangular-shaped physical area. Sensor nodes route their sensed data, through the network of nodes, to a stationary base station which acts as a gate-way to some external network. A mobile sink node moves around in the network, delivering keys and other routing information to the network nodes. The sink and the base station are secure, trust-worthy, tamper proof, and possess powerful resources in terms of energy, memory and computational capability. The sensor nodes, however, have limited memory, energy and computational power. In order to conserve energy, sensor nodes follow a sleeping pattern based on the available energy and event detection frequency. All sensor nodes are similar in terms of energy, memory and computational capabilities.

We define a mobility model for the sink and sensor nodes. Sensor nodes move randomly within the specified coverage area with a common average speed. The sink, however, follows a predefined mobility pattern and moves around within the coverage area with a significantly higher speed compared to the sensor nodes. The movement of the sink is as follows. The sink starts from the lower leftmost corner of the coverage area, moving horizontally towards the right edge. When the sink is very close to the right edge, it changes direction, turns to its left and then moves vertically. After moving a short distance, the sink changes its direction to turn to its left and starts moving horizontally towards the left edge of the coverage area. Once the sink reaches the left edge of the coverage area, it changes direction and turns to its right to move vertically for a short distance after which, it turns right and starts moving horizontally towards the right edge of the coverage area. The sink repeats the above sequence of movements until it has scanned the entire coverage area completely and reached the rightmost corner of the top edge of the coverage area. The sink then moves diagonally across the coverage area to reach its starting point at the lower leftmost corner of the coverage area and repeat the same set of movements to traverse the entire coverage area. Fig. 1 illustrates the sink mobility model.

Each sensor node is identified by a unique node ID, known to the sink, base station, and other sensor nodes. Each sensor node can communicate with the sink and the neighboring sensor nodes if they lie within its communication range. Similarly, the sink node can communicate with sensor nodes which lie within its communication range. The communication range of the sink is significantly larger

compared to that of the sensor nodes. Sensor nodes securely communicate with each other and with the sink by encrypting their messages using symmetric cryptographic keys. Each pair of sensor nodes in the network shares a unique pair wise cryptographic key, used to encrypt all the communications between them. Each sensor node communicates with the sink using a unique pair-wise key. When a node is ready to send data to another sensor node or to the sink, it encrypts its message using the unique encryption key shared with the recipient of the message. The recipient of the message authenticates the sender by verifying the use of the correct pair-wise key, corresponding to each sender-receiver node pair.

Due to the memory limitations of sensor nodes, we assume that a node can store only a limited number of pair-wise keys, i.e., at any given point of time, a sensor node stores the encryption key that it shares with the sink, and shared keys to the sensor nodes in its immediate neighborhood. Because of this, when a node g moves, it cannot communicate with the sensor nodes in its new neighbor-hood as it does not possess shared keys to communicate with those nodes. So node g waits until the sink is within its communication range and then requests the sink for keys. The sink then generates new pair-wise keys for node *g* to communicate with its neighboring nodes, and sends these keys to node g and its neighboring nodes. After obtaining shared keys to its neighboring nodes, node g can communicate with them and route its sensed data through them. This key distribution model ensures that any sensor node can maintain local connectivity anywhere within the coverage area. The assumption that the sink moves at a significantly higher speed compared to the sensor nodes ensures that sensor nodes can obtain keys and routing information frequently. We also assume that, the sink periodically updates the pair wise keys used by sensor nodes to communicate with each other.

Each sensor node possesses a unique node-restoration key used to restore the node back into the network in the event of being quarantined due to the presence of an imposter. Corresponding to each unique node-restoration key in possession of a legitimate sensor node, the sink possesses a public-node-restoration key. The sink uses this public-node-restoration key to encrypt its communication with a quarantined node, in order to restore a previously quarantined node into the network. The sink and node employ asymmetric cryptography during node-restoration, whereas, the sink and nodes use symmetric cryptography for all other communications.

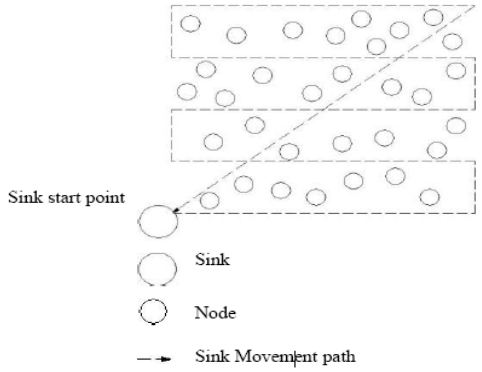

Fig. 1. Mobility model for the sink.

B. Threat Model

We define an imposter to be a malicious entity who uses the identity of a legitimate sensor node to communicate with the sink node or other legitimate sensor nodes in the network. In our model, we assume that the imposter has obtained the shared encryption key between the sink node and a legitimate node g, possibly by way of breaking the cryptographic keys shared between g and the sink node. The imposter then uses this shared key to communicate with the sink, posing as the legitimate node g. In the rest of the paper, we discuss our proposed algorithm to detect such an imposter.

C. The Proposed Scheme

In this section, we describe our scheme to detect and quarantine imposters in a MWSN. Our scheme uses nonce-values and blackout-time mechanisms to enable the sink to detect imposters in the net-work. We prevent detected imposters from communicating with sensor nodes and the sink by using a quarantine mechanism. Since the quarantine mechanism prevents both the imposter and the legitimate node from communicating within the network, we use a node-restoration mechanism to add the legitimate nodes back into the network. In this discussion, we assume an arbitrary legitimate sensor node *g*, located at an arbitrary position in the network. As described in the network model, node g randomly moves around within the coverage area of the MWSN. We also assume that there is an imposter node *i* which impersonates node g using its node ID and encryption keys.

Assume that the sink encounters node g at point p_0, while moving around in the coverage area. After moving a certain distance in the network, the sink reaches point p_1. Let us assume that node

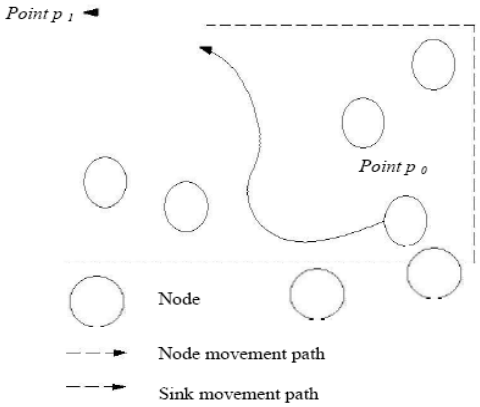

Fig. 2. Node movements cause frequent requests to sink from same node ID.

G contacts the sink again at point p_1. The sink does not know whether the node at point p_1 is indeed the legitimate node *g*, or an imposter *i* posing to be node g. The node at point p_1 could be the legitimate node *g* which has moved and reached point p_1 where it encountered the sink again as illustrated in Fig. 2. Alternatively, the node at p_1 could be an imposter *i* who has obtained the shared keys between the sink and node *g*, and used it to fake the identity of node *g*. It is also possible that the imposter has replayed an earlier message sent by node *g* to the sink. In the above cases, it is not possible to validate the identity of a node claiming to be node *g*, solely based on the encryption keys used. Thus, it becomes essential to identify whether a node is an imposter, despite the use of

correct encryption keys.

The proposed scheme uses a nonce-values mechanism to detect imposters anywhere in the network. As discussed earlier, sensors are constantly in motion and thus, need to obtain encryption keys to the neighboring nodes in order to route their sensed data. Sensor nodes communicate with the sink, requesting encryption keys to neighboring nodes in their vicinity. The sink needs to authenticate each sensor node requesting encryption keys, and grant the encryption keys to only those nodes which are authenticated. The sink node uses a nonce-values mechanism to authenticate sensor nodes. At the time of installation, each legitimate node in the net-work is issued a unique nonce-value, known only to the node and the sink. When node *g* needs to communicate with the sink, it sends its nonce-value to the sink as part of the message. The sink receives the message and verifies whether the nonce-value sent by node *g* matches the nonce-value maintained by the sink for node *g*. If the nonce-values match, the sender node is assumed to be a legitimate node and the communication can proceed. The sink, then, generates and sends new encryption keys enabling node *g* and its neighboring nodes to communicate with each other. At the end of such a successful communication between the sink and the node *g*, the sink generates a new nonce-value for node g, stores it in its memory and sends a copy to node *g*. This new nonce-value will be used to authenticate the identity of node g in the next communication between the sink and node *g*. The next time when node *g* communicates with the sink, it sends this new nonce-value to the sink to verify its identity. If the nonce-value sent by node g does not match the nonce stored by the sink for node *g*, the sink identifies node *g* to be an imposter.

When a sink detects an imposter *i* impersonating node *g*, it per-forms a quarantining mechanism by which any node using the identity of node g is not allowed to communicate with the sink again. As a result, the imposter will not be able to use the identity of node g to obtain any more encryption keys from the sink, thus curtailing its ability to communicate with legitimate sensor nodes. (The quarantining mechanism is discussed later in this section). In effect, the nonce-value of a node acts as a temporary authentication code to authenticate a single communication between the node and the sink. Nonce-values are never repeated, and this feature allows the sink to easily identify replayed messages. If an imposter *i* tries to communicate with the sink by replaying an old message sent by node *g* to the sink, the sink can immediately detect this node as an imposter because the previously used nonce-value is no longer valid.

In the above discussion of the nonce-values mechanism, we have considered an imposter located in an arbitrary position within the coverage area. We now consider the behavior of the nonce-values mechanism when the imposter *i* is located at a close proximity to node *g*. When an imposter *i* is located close to node *g*, it can intercept the messages sent and received by node *g*. Furthermore, since the imposter knows the shared key of node g with the sink, the imposter can obtain the nonce-value issued by the sink to node g at the end of each communication. The imposter can then use this nonce-value together with the correct encryption key to fake the identity of node g and interact with the sink. When this happens, the sink will not be able to detect the imposter because it supplies the correct nonce-value corresponding to node g, thus allowing the imposter to freely communicate with the sink and obtain shared keys to additional number of sensor nodes.

Once an imposter *i* has obtained the correct nonce-value for node g in the above manner, two cases are possible. Both these cases result from the relative difference in speeds of the sink, node g and imposter *i*. The sink moves faster than the node because of which, the sink, the node and its imposter will not remain in the same communication range for an extended period of time. First, let us consider the case that the sink has moved out of the communication range of node g and that the imposter *i* continues following the sink. The imposter can now interact with the sink, request keys to its neighboring nodes, and at the same time, inevitably update the nonce-value for node *g*. In this case, the sink will be able to detect this imposter if and only if it meets node g again. When the sink encounters node *g* again, node *g*, unaware of the recent updates to its nonce-value, supplies an old nonce-value to the sink. Subsequently, the sink detects the presence of the imposter due to the incorrect nonce-value supplied by node g. The quarantining mechanism will then be initiated, effectively preventing both the imposter *i* and node *g* from communicating within the network.

The second possible scenario is that the imposter *i* moves out of the communication range first, such that the sink and node *g* are in the same communication range for a while longer. In this case, node g can continue communicating with the sink, requesting keys to its neighboring nodes, thereby updating the nonce-value for node g. As with the first case, the imposter is detected when the sink meets the imposter again, and if the imposter communicates with the sink. The imposter supplies an old nonce-value causing the sink to detect its presence. Thus, in summary, for both of the above cases, the sink can detect the imposter *i*, based on the nonce-values scheme; however, the time taken to detect the imposter in this manner depends on when the sink encounters the imposter *i* or the legitimate node g, for a second time.

To quickly detect the imposter when both the imposter *i* and node *g* are in the communication range of the sink, the proposed scheme uses a blackout-period mechanism. The blackout-period mechanism restricts a node *g* from communicating with the sink unless a certain time period, $T_{blackout}$, has elapsed since its last communication with the sink. The blackout period denoted by $T_{blackOut}$, is a time duration during which node g is not allowed to communicate with the sink even if the sink remains within the communication range of node *g*. The value for $T_{blackout}$ is set to be the maximum amount of time it takes for the sink to move out of a common communication range with the node. For the worst case calculation, this value is computed starting from the instant a node encounters the sink and ending at the instant the sink moves out of the communication range of the node. This ensures that, when the sink is in the communication range of a node *g*, only a single communication takes place between the sink and the node. If a node communicates with the sink before its blackout-period has elapsed, the sink detects the node to be an imposter.

The blackout-period restriction is useful in detecting an imposter located close to node *g*. When the node *g* communicates with the sink, the imposter *i* can overhear the nonce-values issued to node. However, if the imposter *i* communicates with the sink before the blackout-time has elapsed, the sink can detect this node as an im-poster. Thus, enforcing the blackout-period mechanism restricts the ability of an imposter to freely communicate with the sink even though it knows the nonce-value of a node *g*.

Blackout-time can be calculated using the worst case condition for which the sink remains in the communication range of the node for the longest period of time. This happens when the sink and the node are moving in the same direction. Let S_s m/s be the speed at which the sink moves, following the motion pattern described in the mobility model. Let an arbitrary node *i* with a communication range of radius R_s m be moving at an average speed of S_n m/s. For the worst case, assume that node *i* and the sink are moving in the same direction. Assume that node *i* communicated with the sink right at the point when the sink enters the communication range of the node. The sink will continue to remain in the communication range of the node for a period of time given by $T_{max} = 2R_s = (S_s\text{-}S_n)$

Thus, the blackout period should be set to a value greater than T_{max} calculated above, in order to ensure that any node in the net-work communicates exactly once with the sink for the entire time during which the sink is within the communication range of the node.

When imposter *i*, claiming to be node g, is detected by the sink as an imposter, the sink performs a quarantining mechanism on node ID *g*. The sink first marks node *i* to be an imposter and cuts off all communication with any node using node ID *g*. As a result, both imposter *i* and node *g* cannot interact with the sink to obtain shared keys to neighboring nodes thus restricting their communication within the network. Next, the sink changes the shared keys used by node g to communicate with its neighboring nodes by generating new keys and distributing them to all nodes it encounters in its path. This makes it impossible for the imposter to communicate with other sensor nodes. It should be noted that, by performing the quarantining mechanism, both the imposter node i and the legitimate node *g* are quarantined.

Every time the sink completely traverses the MWSN coverage area, it resets the encryption keys used by each pair of nodes in the network. This is done to restrict any imposter from gradually collecting the encryption keys to a progressively large number of nodes in the network. In each traversal of the coverage area, the sink initiates node restoration for legitimate nodes which have been previously quarantined, by means of a node-restoration mechanism. Assume that node *q* is a legitimate node, previously quarantined by the sink. When the sink encounters this node *q* in a new traversal of the coverage area, it initiates a node-restoration mechanism to restore node *q* back into the network. To do so, the sink sends a new sink-to-node encryption key encrypted with the public-node-restoration-key. The node *q*, if legitimate, will be able to decrypt the message using the node-restoration-key stored in its memory. Upon successfully decrypting the message, the node *q* obtains the new encryption key to communicate with the sink. The node *q* then sends an acknowledgment to the sink, upon receiving which, the sink restores the node *q*, back into the network. Note that, by employing the node-restoration mechanism, legitimate nodes which were previously quarantined can be restored into the network, thereby addressing the possibility of an imposter disabling the entire network by gradually impersonating more nodes.

D. Two-Sink Solution

In order to speed up the imposter detection process, an additional sink can be employed. The starting position of the second sink is set to be diagonally opposite to the first sink i.e., the second sink is located at the far right end of the top edge of the coverage area. While the first sink moves from bottom to top following the motion pattern described in the sink mobility model, the second sink moves top to bottom using the same motion pattern and same speed as the first sink, but with reversed directions compared to the first sink. When the first sink reaches the rightmost point of the top edge, the second sink reaches the leftmost corner of the bottom edge of the coverage area. Both the sinks then move diagonally to reach their respective starting points and repeat their motion pattern all over again. The second sink scans the top half of the coverage area during the time in which the first sink scans the bottom half of the coverage area.

In order to facilitate faster detection of imposters, the two sinks share network information by interacting with each other at regular intervals. The sinks exchange information such as the shared keys of sensor nodes with their neighboring nodes, the most re-cent nonce-values for each sensor node, and information about imposters that have already been discovered. The presence of an additional sink increases the local connectivity of a sensor node be-cause the probability of a sensor node encountering a sink to obtain keys, is higher when there is an additional sink present. Also, due to the periodic exchange of network information between the sinks, they will be able to detect the presence of imposter nodes faster. In fact, increasing the rate at which the two sinks communicate with each other decreases the average detection time required to detect an imposter.

IV. Simulation

We evaluate the proposed scheme using the Java JDK 1.6. We consider an abstracted sensor network consisting of N nodes randomly moving in an area of 1000 m × 1000 m, where $100 \; N \; 500$. At the time of installation, sensor nodes are evenly distributed in the form of a grid in the coverage area. The position of the sink is initialized to be the leftmost corner of the bottom edge of the coverage area. After installation, nodes start moving in random directions with a common average speed, following a Random Way-point Mobility (RWM) model of movement. In the RWM model, each node moves from its initial position to a randomly selected position in a straight line. After reaching the random position, the node stays for a predefined time, and then randomly selects another position to which it moves in a straight line. This process repeats during the given simulation time.

We define some metrics of interest to evaluate the effectiveness of the proposed scheme in detecting the

presence of an imposter. Time taken to detect/quarantine an imposter is an important criterion which indicates how quickly the sink identifies an imposter. This metric also indicates how quickly a corrective action can be taken, to reduce the security risks to the network. We analyze the behavior of our proposed algorithm by varying several parameters such as the relative speeds of the sink and sensor nodes, and varying node densities in the network coverage area. We also study the effect of having multiple sinks in the network.

In this simulation, the average speed of node movement is set as 3:0 m/s. The sink moves according to the sink mobility pattern with an average speed of 4:00 m/s. For the two-sink scenario, an additional sink with same speed, but moving in the opposite direction is added. The two sinks exchange node and imposter information at an interval of 10 s.

Nodes are preloaded with unique nonce-values which the sink uses to validate the legitimacy of a sensor node. Nodes have a communication range of 40 m; a sensor node can communicate with any device (sink or another sensor node) which is within this communication range. Each node waits until the sink is within its communication range before communicating with the sink to re-quest for shared keys of its neighboring nodes. The program also calculates the blackout time for each test case using the formula discussed in Section 3.2. If any node violates this blackout time or provides an incorrect nonce-value, then the sink marks this node as an imposter. Imposter nodes are introduced in the network at the time of installation. An imposter node operates from a random location and waits until the sink is within its communication range. Then, it communicates with the sink using a random node ID and using the encryption keys of the node which it is impersonating. The following cases were considered in the simulation:

Case 1: An imposter impersonates a legitimate sensor node *i* using only the node ID *i* to communicate with the sink. This is used to simulate a case in which the sensor node and its imposter are lo-cated far away from each other and the imposter has no information about the nonce-value for node *i*.

Case 2: The imposter is located close by the legitimate node which it impersonates. This corresponds to the case in which the imposter is located at a close distance to the node *i*, so it can over-hear the nonce-value given by the sink to the node *i*.

A. Implementation

Algorithm 1: Algorithm for Node Process in Node *i*

```
if SinkInRange() == TRUE then
  if T_BlackOut == 0 then
    Generate(NeighborList)
    RequestforKeys(NeighborList; NONCE_i)
    ReceivefromSink(Keys; NewNONCE_i) NONCE_i
    NewNONCE_i
    TBlackOut  Tmax
  else
    TBlackOut  TBlackOut  1

  end if
  end if
```

Algorithm 2: Algorithm for processing node requests at the sink

```
procedure PROCESS
REQUESTFORKEYS(NeighborList,
NONCE_i)
  receivedNONCE_i  NONCE_i
  if TBlackOut_i == 0 then
    if receivedNONCE_i == storedNONCE_i then
      GenerateKeys(NeighborList)
      GenerateNewNONCE(i) SendtoNode(Keys;
      NewNONCE_i) storedNONCE_i NewNONCE_i
      TBlackOut_i  TBlackOut_i
  else
      QuarantineNode(i)
  end if
  QuarantineNode(i)
 end if
end procedure
```

The algorithms detailed below are used to simulate the proposed scheme. When an event is detected, the sensor node processes the sensed data and stores it in local memory. When it detects the sink in its communication range, the node checks whether the blackout period has elapsed, in which case, it sends a message to the sink with a list of its neighboring nodes and the nonce-value which was previously assigned to it. This is depicted in Algorithm 1.

When the sink receives the request from the node, it first ascertains that the node is allowed to communicate with it by checking the blackout period. In case the blackout period has elapsed, the sink checks the nonce-value sent by the node against the nonce-value locally stored by the sink. If the nonce-value is verified, the sink can generate the new nonce and blackout period values along with the keys which the node has requested for. In any case, if either the blackout period or nonce-value fails, then the node is marked as an imposter and quarantined. This is given in Algorithm 2.

B. Results

The simulation results are shown in graphs below. Node speed is fixed at 3:0 m/s and sink speed is varied from 4:0 m/s to 10:0 m/s. A single imposter node is introduced into the network at the time of installation; this node impersonates a random network node and is located at a random position in the coverage area. When the sink is within the coverage area of the imposter, it communicates with the sink whereby it is detected by the sink as an imposter.

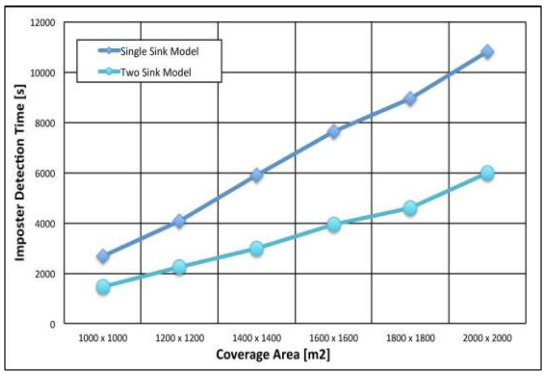

Fig. 3. Plot showing variation of imposter detection time with coverage area for the single sink and two-sink models.

Fig. 3 and Fig. 4 shows the imposter detection time with the coverage area and communication range of single sink and two sink model whereas Fig. 5 shows the variation of imposter detection time with number of imposters detected with a single sink and two sink models at a given time. We also notice that the time taken to detected imposters in two sink models is reduced when compared to single sink and the number of imposter detected by two sink model is increased when compared to single sink model.

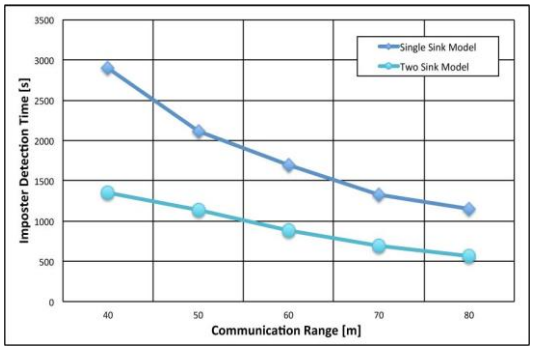

Fig. 4. Plot showing variation of imposter detection time with communication range for the single sink and two-sink models.

From the plot in Fig. 6, we can see that, as the sink speed in-creases, the time required by the sink to detect an imposter drastically reduces. This is due to the fact that, the faster the sink moves, the sooner it comes in contact with imposter nodes which it detects as instantly because of the nonce-value check. A similar trend is observed for the two sink model where the time to detect an imposter decreases as the speed of the individual sinks is increased. We can also see that the two sink model performs better in terms of the time taken to detect an imposter. This is as expected because the presence of a second sink and frequent communication between the sinks helps to identify imposters at a faster rate.

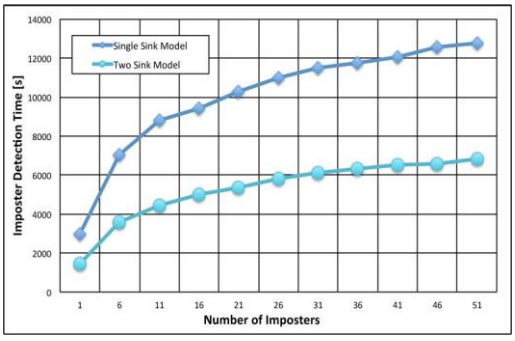

Fig. 5. Plot showing variation of imposter detection time with no of imposters detected for the single sink and two-sink models.

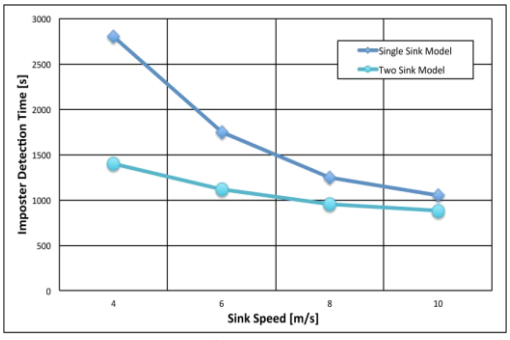

Fig. 6. Plot showing variation of imposter detection time with maxi-mum number of imposters detected in a single sink and two-sink models.

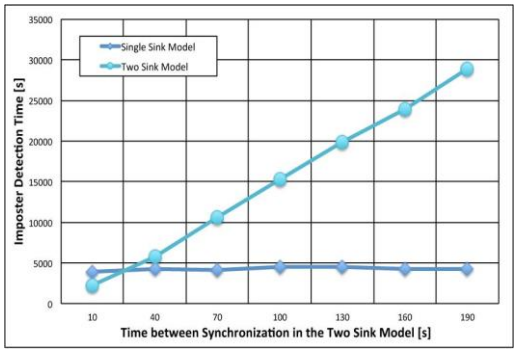

Fig. 7. Plot showing variation of imposter detection time with frequency of synchronization of sinks in two sink model.

Fig. 7 demonstrates the effect of time between two synchronizations of the sinks in the Two Sink Model. Node speed is fixed at 3.0m/s and sink speed is varied from is fixed at 4.0m/s. A single imposter node is introduced into the network at the time of installation; this node impersonates a random sensor node by using the nonce-value of a legitimate node. When the sink is within the coverage area of the imposter, it communicates with the sink whereby it is detected by the sink as an imposter. The two sinks communicate with each other at predetermined intervals in order to exchange network data such as the most recent nonce-value issued to a sensor node and node IDs of sensor nodes which have been identified as imposters. The synchronization time is varied from 10 s between two consecutive synchronizations to 100 s. For each value of synchronization time, the time to detection of the imposter node is logged and plotted, resulting in the graph given in Fig. 6 From the plot in Fig. 6, we can see that, as the time between two consecutive synchronizations increases, the time required to detect an imposter decreases.

V. Conclusion

In this work, we proposed a novel imposter detection scheme to detect and quarantine imposters in MWSNs. Our scheme uses nonce values and black-out time mechanisms to enable sink to detect imposters in the network. We describe the mobility model for the sink such that the sink is able to cover the entire network area and detect imposters based on the veracity of the nonce value and the blackout period parameters. Imposters are detected by the sink which uses symmetric cryptography for all its communications with other nodes. If the sink identifies an imposter node it prevents the imposter from communicating with other sensor nodes by using quarantine mechanism. The quarantine mechanism pre-vents the imposters and the legitimate nodes from communicating within the network. Hence, we propose node restoration mechanism that uses asymmetric cryptography to add the legitimate node back into the network. In order to facilitate faster detection of imposters we introduce a two-sink model which helps significantly reduce the time to detect an imposter.

We evaluated the proposed scheme and our simulation results shows that as the sink speed increases the time required by the sink to detect the imposter drastically decreases. Similarly, we noticed that for the two sink model the time to detect an imposter decreases as the speed of

individual sinks is increased. We also conclude that the two sink model performs better in terms of time taken to detect imposters. Hence, increasing the frequency of synchronization between the sinks reduces the time to detect an imposter.

REFERENCES

[1] S. A. Munir *et al*. "Mobile wireless sensor net-work: Architecture and enabling technologies for ubiquitous computing," in *Proc. the 21st International Conference on Advanced Information Networking and Applications Workshops*, 2007, vol. 2, pp. 113–120.

[2] N. A. B. A. Aziz, A. W. Mohemmed, and B. S. D. Sagar, "Particle swarm optimization and voronoi diagram for wireless sensor networks coverage optimization," in *Proc. the International Conference on Intelligent and Advanced Systems*, 2007, pp. 961–965.

[3] B. Liu *et al*., "Dynamic coverage of mobile sensor net-works," in *ArXiv E-prints*, Jan. 2011.

[4] G. Wang, G. Cao, and T. L. Porta, "Proxy-based sensor deployment for mobile sensor networks," in *Proc. 2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2004.

[5] V. Bhuse and A. Gupta. "Anomaly intrusion detection in wireless sensor networks," *J. High Speed Networks*, vol. 15, no. 1, pp. 33–51, 2006.

[6] T. G. Lupu, "Main types of attacks in wireless sensor networks," in *Proc. the 9th WSEAS International Conference on Signal, Speech and Image Processing*, 2009, pp. 180–185.

[7] T. Roosta, S. Shieh, and S. Sastry, "Taxonomy of security attacks in sensor networks and countermeasures," in *Proc. the First IEEE International Conference on System Integration and Reliability Improvements*, vol. 25, 2006.

[8] J. Hall, M. Barbeau, and E. Kranakis, "Detection of transient in radio frequency fingerprinting using signal phase," *Wireless and Optical Communications*, pp. 13-18, 2003.

[9] S. Hussain and M. S. Rahman, "Using received signal strength indicator to detect node replacement and replication attacks in wireless sensor networks," in *Proc. SPIE*, vol. 7344, 2009.

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

[11] J. W. Ho, M. Wright, and S. K. Das, "Fast detection of replica node attacks in mobile sensor networks using sequential analysis," in *Proc. IEEE INFOCOM 2009*, pp. 1773–1781.

[12] M. Conti *et al*., "Emergent properties: detection of the node-capture attack in mobile wireless sensor networks," in *Proc. the First ACM Conference on Wireless Network Security*, 2008, pp. 214–219.

Ebrahim Al Rashed received his PhD and MS degrees in computer engineering from the University of Southern California, Los Angeles, CA, in 1997 and 1993, respectively.

He is currently working as an assistant professor in Department of Computer Engineering at Kuwait University. His research interests includes network security, cloud computing and sensor networks.

Dr. Al Rashed has membership in professional organizations such as ACM Association of Computing Machinery, IEEE Computer Society and IEEE Communications Society.

Mehmet Hakan Karaata received his PhD and MS degrees in computer science from The University of Iowa, Iowa City, Iowa in 1995 and 1990 and did his BS degree in Computer Science from The University of Hacettepe, Ankara, Turkey in 1987.

He joined Bilkent University, Ankara, Turkey as an assistant professor in 1995. He is currently working as a professor in Department of Computer Engineering at Kuwait University.

His research interests include mobile computing, distributed systems, fault tolerant routing and self-stabilization.

Electrical and Electronics