Indian Journal of Science and Technology, *Vol 9(31), DOI: 10.17485/ijst/2016/v9i31/86364, August 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

CBCD: Cluster based Clone Detection in Mobile Wireless Sensor Networks

J. Anthoniraj[1*] and T. Abdul Razak[2]

[1]Bharathidasan University, Trichy - 620024, Tamil Nadu, India; antonyrajmiet@gmail.com
[2]Jamal Mohamed College, Trichy - 620020, Tamil Nadu, India

Abstract

Wireless Sensor Network consists of independent sensor nodes that are responsible for monitoring physical conditions. WSN is deployed in hostile environments. So it is vulnerable to capture and compromise by an attacker. An adversary can launch various types of attacks on WSN that can be classified as layer-dependent attack and layer-independent attacks. The layer-independent attacks are Sybil attack and Clone attack. In Clone attack an adversary can capture a sensor and creates clone of a captured node. These clone nodes are deployed in network area. It is difficult to detect clone nodes in the sensor network. There are so many clone detection protocols are available in static and mobile sensor networks. It is more challenging to detect clone attack in mobile WSN compared to the static WSN. The proposed clone attack detection protocol called as Cluster Based Clone Attack Detection (CBCD) discovers the clone nodes available in Mobile WSN. In this protocol sensor network divided into clusters. All the clusters have a cluster head and random number of sensors nodes. This protocol detect clone node according to the movement of sensor nodes. The clone is identified when sensor node move within the cluster or other clusters in the sensor network. Theoretical analysis and simulations have been conducted to evaluate the protocol in terms of clone detection time, clone detection ratio, memory consumption.

Keywords: Clone Attack, Cluster based Clone Detection, Wireless Sensor Network

1. Introduction

Wireless Sensor Network (WSN) has been constructed with several number of sensor nodes. Each sensor node of the network can operate independently. It can be used to monitor various types of physical conditions[1]. In WSN sensor nodes are deployed in an unattended and insecure environment such as battle field or dense forest. The components of a sensor node are sensing unit, processing unit, communication unit and power unit. The sensing unit is composed of a collection of different types of sensors which is needed for measurement of different conditions of the physical environment, the processing unit consists of a processor and a memory, the communication unit consists of a transmitter and a receiver and the power unit provides the energy to the sensor node[2]. In WSN data collected through sensors and transmitted to the base station through wireless media[3]. WSN has various applications that include traffic control, battle field management, disaster management, environmental monitoring, building monitoring, monitoring animal populations, home automation, medical applications etc[1,4]. WSN suffers from many constraints including lack of hardware support for tamper resistance, low computation capability, very small memory, and insufficient power resources, use of insecure wireless communication channels and deployment of sensor nodes in an unattended environment and so on[2,5].

An adversary can launch various types of attacks on WSN that can be classified as layer-dependent attacks and layer-independent attacks. The layer-dependent attacks are Physical layer attacks (jamming, tampering), Data link layer attacks (collision, exhaustion), Network layer attacks (spoofing, wormhole attack, sinkhole attack, selective forwarding, hello flood attack), Transport layer attacks (flooding attack, synchronization), Application layer attacks (altering routing attack, false data injection).

**Author for correspondence*

The layer dependent attacks are Sybil attacks and Clone attacks[1,3,6].

The most dangerous attack is the clone attack. In this attack, the existing sensor node of the sensor network is captured by the adversary. All the secret keys are extracted from the captured node. The attacker reprograms the node and creates a clone of a captured node. The clone nodes are deployed in the sensor network along with original nodes. The adversary can generates any number of clones from a single captured sensor node. The clone nodes are controlled by the attacker and look like a original node. So it is not a easy job to find the clone nodes of the sensor network[7,8,9]. All the notations used in the protocols are described in Table 1.

2. Related Work

There are many clone attack detection protocols available in wireless sensor networks. In the static WSN sensor nodes do not change their location after deployment. But in the mobile WSN sensor nodes change their locations frequently after deployment. The clone attack detection in Mobile WSN is very much difficult compared to the Static WSN[10]. The various clone attack detection protocols of mobile sensor network are given below.

Table 1. Notations and significance

x	Total number of nodes in the network
y	Total number of clusters in the network
ngh	Neighbor node
BS	Base Station
n_i	sensor node
C_j	Cluster
Cj_H	Cluster Head
ID_{ni}	Identification of node ni
LC_{ni}	Location Claim of node ni
H()	Hash function
FP_{ni}	Finger Print of node ni
KUnji	Public key of node ni of cluster Cj
KRnji	Private key of node ni of cluster Cj
$KUCj_H$	Public key of Cluster Head Cj_H
$KRCj_H$	Private key of Cluster Head Cj_H
KUBS,	Public key of Base Station
KRBS,	Private key of Base Station
$KSnjiCj_H$	Session key for node nji and cluster head Cj_H
KS_{CjHBS}	Session key for cluster head Cj_H and base station
$KSn_{10}n_{11}$	Session key for node n_{10} and node n_{11}
m	Total number of keys in a node's key Ring
S	Key Pool

2.1 Centralized Protocols

In[11] proposed a technique called Fast protocol. In this protocol, when a sensor node changes its existing location, it will give location and time information to its neighbor nodes. Then the neighbor nodes send all these information to the base station. Base station detects the clone node with help of the speed of the mobile nodes.

In[12] proposed a method called new mobile protocol. This protocol performs the detection of clone node in three phases. In the first phase, a symmetric polynomial can be randomly created by the key server before nodes are deployed. In the second phase node deployment and pair wise key establishment can be done. In the third phase, the bloom filter of the base station collects the total number of pair wise key generated by each node. If node's pair wise keys exceed the threshold then it is called as clone node.

2.2 Distributed Protocols

In[13] proposed a mechanism called eXtremely Efficient Detection (XED) protocol. Here, if two nodes exist in the same communication range, then each node creates a random number and exchanges it with another node. The node ID and random number are stored in the table of each node. If these nodes meet again they exchange the previously stored random numbers. If the random number does not match with the existing number, the node is confirmed as a clone node.

In[14] proposed a mechanism called Neighbor Based Detection Scheme (NBDS) protocol. In this protocol, the sensor nodes move to the new locations from their existing locations. After movement the sensor node wants to rejoin the network from its new location. It sends the rejoin claim to the nearest new neighbors. They check the authentication of the claim. If it is correct, forwards the claim to selected nodes. The received nodes verify the signature and ID of the claim. The signature and ID does not match with the existing information of the neighbor table, forwards the failed claim to the base station.

In[15] proposed a scheme called Efficient and Distributed Detection (EDD) protocol. In this protocol sensor node moves according to the random way point. The sensor

node selects a location in the network for its movement. Then it moves to the destination point in the sensor network. There it stays inactive for a random amount of time. After that the node moves according to the previous method.

In[16] proposed a technique called Storage Efficient Distributed Detection (SEDD) protocol. In this protocol, for the given time interval every node of the network monitors only a subset of nodes.

In[17] proposed a scheme called Simple Distributed Detection (SDD) protocol. In this protocol, for a given period of time two sensor nodes do not meet twice. Any one the sensor node may be cloned by the adversary.

In[18] proposed a technique called Unary Time Location Storage and Exchange (UTLSE) protocol. In this protocol every node belongs to the unique tracking set. Every node must be the witness of the particular tracking set only. The node enter in to the particular tracking set, existing node ask the time location claim of the new node.

In[19] proposed a method called Single Hop Detection (SHD) protocol. In this each node collects the neighbor nodes. This neighbor node list sent to the one hop neighbors of the node. The received node becomes a witness node, after that it stores the neighbor node list for future verification.

In[20] proposed a mechanism called Patrol Detection for Replica Attack (PDRA) protocol. In this protocol, patrollers detect clones distributed in different zones of the network. If a mobile node moves with a speed higher than the denoted maximum speed, it will be regarded as clone node.

2.3 Comparison of Mobile Protocols

The node deployment and localization are important challenges of mobile wireless sensor networks. The node position can be determined only one time for the static wireless sensor, when sensor nodes are mobile then node position can be obtained continuously. So the mobile wireless sensor network localization needs extra time and energy[20]. The XED protocol detects the clone nodes with minimum time compare to other protocols. This protocol disregards the location information criteria to detect the clone node[13]. The Fast protocol use GPS protocol for routing and it has lower energy overhead due to its centralized approach[11]. The EDD protocol is susceptible to smart adversary and has high memory over head[15,21]. The comparison of various mobile protocols described in Table 2.

Table 2. Comparison of mobile protocols

Protocol	Type of Approach used	Type of Scheme used	Comm. cost	Mem. cost
Fast	Centralized	Based on Speed	O(n √n)	O(n)
New	Centralized	Based on Key	O(n log n)	---
XED	Distributed	Based on Conflict	O(1)	--
NBDS	Distributed	Based on mobility	O(r√n)	O(r)
EDD	Distributed	Based on node meeting	O(1)	O(n)
SEDD	Distributed	Based on node meeting	O(n)	O(£)
UTLSE	Distributed	Based on Time-location	O(n)	O(√n)
PDRA	Distributed	Based Patroller	O(n)	--

3. Protocol Frame Work

3.1 System and Network Model

The following methodologies have been used during the development and analysis of CBCD protocol[22–25].

- **Hardware Configuration**: MICAz mote sensor node which has ATmega128L 8 bit processor running at 8 MHz speed with RAM size of 4KB and power is supplied via two AA batteries to provide a current capacity of 2000 mAh. The radio transceiver TICC240 transmits and receives the data at the rate of 250kbps.
- **Node Deployment**: Uniformly Random deployment is used for placing the sensor nodes. There is no need for prior knowledge of optimal placement.
- **Localization**: The position of the node is identified through Global Positioning System (GPS). It is a popular technology and reduces the energy consumption.
- **Communication Standard:** IEEE 802.15.4 LR-WPAN Zigbee technology has been used for transmission. There are 27 channels available with a 20 kpbs transmission bandwidth.
- **Routing Protocol:** Greedy Perimeter Stateless Routing (GPSR) location based routing protocol is used for our implementation. It is a geographic routing protocol and it is used to route packets between any pair of nodes.
- **Encryption:** The Elliptic Curve Cryptography (ECC) algorithm is used to encrypt the various keys used in our work.
- **Key Management:** The Diffee Helman public key management system is used for all the key management process. The sensor nodes are assigned a random

subset of keys from a large key pool before the deployment of the network.

- **Network Architecture:** Hierarchical architecture is used in our proposed system. Here, a group of sensor nodes form a cluster. Each cluster has a cluster head, which is responsible for sending data from the cluster members to the base station.
- **Cluster Head Selection:** The Legacy algorithm is used to select the cluster head of the cluster randomly for every process.
- **Adversary Model:** The adversary has the ability to capture any number of sensor nodes. Once a node is compromised, the adversary gains full control over the node. An adversary can create as many clones of the captured node as required and deploy in the network. In our experiment 50 clone nodes are deployed in the sensor field.

4. Clone Detection in Mobile WSN

4.1 Public Key Management

Before deployment, each cluster has set of nodes, the node ID (*IDnji*) and location (*LCnji*) of these nodes are stored in the corresponding cluster head. In the same way cluster head ID (ID_{CJH}) and location (LC_{CJ}) of all the clusters are stored in the base station. Public and private keys are distributed by base station to the cluster heads and sensor nodes. All the nodes discover its neighbor nodes by transmitting the HELLO message. After the discovery, it distributes the public keys to the neighbor nodes.

4.2 Session Key Establishment

Every node distributes the session key and creates symmetric communication with its neighbor nodes. All the nodes in the cluster must seek admission from cluster head for their data transfer. The cluster head verifies the admission request and provides the session key to that node. In the same way all cluster heads seek admission with base station. The base station verifies the admission request and provides the session key to that cluster heads.

4.3 Finger Print Computation

Every node collects the location information from all its neighbor nodes. The node stores the node ID and location of the neighbor nodes in the neighbor table as described in Table 3. It sends the neighbor node list to the cluster head for verification. After the verification, node computes the finger print with Boolean sum of neighbor node Ids [FP = ngh1 (ID) V ngh2 (ID) V ngh3 (ID)]. The node must attach the Finger print along with the message content which send to the cluster head

Table 3. Neighbor table

NodeID (ID_{ni})	Location Claim (LC_{ni})
….	….
….	….

5. Clone Detection using Node Movement

5.1 Movement of the Sensor Node within the Cluster

The node n_{10} belongs to Cluster C_1 moves from its location to some other location within the cluster.

5.1.1 Before Movement

5.1.1.1 Remove its Location Information from the Neighbor Nodes

Node n_{10} may be neighbor of many nodes. So it informs to its neighbor nodes to remove the location information from its neighbor table. Node n_{10} send the message to remove the location information from the neighbor table to all its neighbors. According to the instruction given by the node n_{10}, node n_{11} remove the location information of n_{10} from its neighbor table. After remove the location information from the neighbor table the node n_{11} send the ack to the node n_{10}. Likewise node n_{10} send remove message to all its neighbor nodes. The neighbor nodes remove location information of n_{10} from its neighbor table. After it receives remove Ack from all its neighbors, node n_{10} go for the movement.

5.1.1.2 Seeking Possible Locations for Movement

Node n_{10} does not move to the location where ever it wants, but it can move to the location given by the cluster head. Node n_{10} sends a request message to Cluster Head to give the possible locations for its movement. Cluster Head C_{1H} receive the information and verify the node belongs to its cluster (or) not. If it is not belongs to its cluster, node is identified as clone. Otherwise the Cluster

Head C_{1H} sends various locations to be move to the node n_{10}. Now node n10 choose any one location given by the C_{1H} for its movement. Before movement node n_{10} inform its new location to be move to the Cluster Head C_{1H}. Now Cluster Head C_{1H} changes the status of node n_{10} from existing to moving. Cluster table of C_1 before movement is described in Table 4.

Algorithm 1

Remove its location information from the neighbor nodes

Step 1: Remove my entry in the neighbor table

$$n_{10} \rightarrow n_{11}$$

$$EKS_{n10n11}(remove(ID_{n10}),ID_{n10,}\ ID_{n11}).$$

Step 2: Acknowledgement after remove the entry

$$n_{11} \rightarrow n_{10}$$

$$EKS_{n10n11}(removeAck(ID_{n10}),ID_{n10,}ID_{n11}).$$

Step 3: Send remove my entry msg to all its neighbors.

Step 4: Collect the Ack from all the neighbors.

Seeking locations for movement

Step 5: Seeking locations for movement from cluster $HeadC1_H$

$$n_{10} \rightarrow C_{1H}$$

$$EKS_{C1Hn10}(moveLocReq,ID_{n10,}LC_{n10},FP_{n10}).$$

Step 6: C_{1H} verifies the request

If existing $(ID_{n10},LC_{n10},FP_{n10}) \neq$ received$(ID_{n10},LC_{n10,}FP_{n10})$

Clone node detected .

If existing$(ID_{n10},LC_{n10},FP_{n10}) =$ received$(ID_{n10},LC_{n10},FP_{n10})$

Request accepted.

Step 7: Request is accepted by C_{1H}

C_1 send possible locations to node n_{10}

$$C_{1H} \rightarrow n_{10}$$

$$EKS_{C1Hn10}(moveLocList,ID_{n10},ID_{C1H},<Moveable\ locations\ list>).$$

Step 8: Node n_{10} choose any one location given by C_{1H}.

Step 9: Node n_{10} inform its new location to C_{1H}

Table 4. Cluster table of C1

node	Node ID	status	Loc Claim(LC)	Ngh node list	Finger Print(FP)	New Loc Claim(NLC)
n10	….	Moving	…..	…..	…..	…..
n11	….	Existing	…..	…..	…..	…..

$$n_{10} \rightarrow C_{1H}$$

$$EKS_{C1Hn10}(newLocInf,ID_{n10,}LC_{n10},newLC_{n10,}FP_{n10}).$$

Step 10: Cluster Head C_{1H} store the new location in the table and update the status as moving.

5.1.2 After Movement

5.1.2.1 Node Seeking Readmission

The node n_{10} moves to the new location within the cluster. It must get the readmission from Cluster Head to do its regular process in the cluster. Node n_{10} sends the readmit message to the Cluster Head C_{1H}. The Cluster Head C_{1H} receives this readmit request and checks the given information with the existing information in the cluster table. Suppose existing information do not match with the received information, node n_{10} is identified as clone node. If existing information is same as the received information, then Cluster Head C_{1H} select the neighbor node list of node n_{10} from the cluster table and any old neighbor of the node n_{10} is selected. Suppose node n_{11} is the neighbor of node n_{10}, Cluster Head C_{1H} check whether node n_{10} still exist in the neighbor table of n_{11}.

5.1.2.2 Node Available in the Neighbor Table

Now node n_{11} check whether the node n_{10} is available in its neighbor table or not. Suppose the node n_{10} is available in the neighbor table of n_{11}. The node n_{11} send the node available message to Cluster Head C_{1H}. The node n_{10} already available in its old location, so it can not appear in another location, Cluster Head confirm the node n_{10} is a clone node. The Cluster Head inform this clone node identification information to the Base Station.

5.1.2.3 Node not Available in the Neighbor Table

Suppose node n_{10} is not available in the neighbor table of n_{11}. Node n_{11} send node not available message to the cluster head C_{1H}. Now C_{1H} confirm that the node n_{10} is not in its old location. Now C_{1H} updates its location and remove its existing Finger Print. Now the C_{1H} sends the readmit accepted message along with its session key (KS_{C1n10}) to the node n_{10}. Now the node n_{10} made neighbor node discovery and other process as per the protocol.

Algorithm 2

Node seeking readmission

Step 1: $n_{10} \rightarrow C_{1H}$

EKU_{C1H}(readmitReq,ID_{n10},ID_{C1H},$newLC_{n10}$, FP_{n10}).

Step 2: If existing [ID_{n10},FP_{n10},$newLC_{n10}$] ≠ received[$ID_{n10,}$ FP_{n10}, $newLC_{n10}$]
Clone identified.

Step 3: If existing[ID_{n10}, FP_{n10}, $newLC_{n10}$] = received [ID_{n10}, FP_{n10}, $newLC_{n10}$]
Readmission accepted.

Step 4: C_{1H} gives possible locations to be move

$$C_{1H} \rightarrow n_{10}.$$

EKS_{C1Hn10}(moveLocList,ID_{n10},ID_{C1H},<Moveable locations list >).

Step 5: Node n_{10} choose any one location from the list for movement.

Step 6: Before movement node n_{10} inform its new location to C_{1H}.

$$n_{10} \rightarrow C_{1H}$$

EKS_{C1Hn10} (newLocInf, $ID_{n10,}$ LC_{n10}, $newLC_{n10,}$ FP_{n10}).

Step 7: C_{1H}verify the location claim and finger print of node n_{10}
Store the new location claim in the cluster table
Update the status of the node n10 as moving.

Step 8: node n_{10} move to the new location
Node n_{10} send a readmission request to join the cluster.

Step 9: $n_{10} \rightarrow C_{1H}$
EKU_{C1H}(readmitReq,ID_{n10},ID_{C1H},$newLC_{n10}$, FP_{n10}).

Step 10: The cluster head C_{1H} verify the given finger print and new
Location with the existing information
If existing [ID_{n10} ,FP_{n10}, $newLC_{n10}$] ≠ received [$ID_{n10,}$ FP_{n10}, $newLC_{n10}$]
Clone node identified.

Step 11: If existing [ID_{n10} ,FP_{n10}, $newLC_{n10}$]= received [$ID_{n10,}$ FP_{n10}, $newLC_{n10}$]
Readmission accepted.

Step 12: C_{1H} selects any one neighbor from neighbor node list of node n_{10}.
Suppose node n_{11} is the one of the old neighbor of node n_{10}.

$$C_{1H} \rightarrow n_{11}.$$

Step 13: Node n_{11} check whether the node n_{10} is available in its neighbor table or not.

Node available in the neighbor table

Step 14: Node n_{11} send available message to the Cluster Head $C1_{H}$.

$$n_{11} \rightarrow C_{1H}$$

EKS_{n11C1H} (available(ID_{n10}), ID_{n11}, ID_{C1H}).

Step 15: Now C_{1H} confirm n_{11} is a clone node.

Step 16: Cluster Head inform this clone node identification to the Base Station.

Node not available in the neighbor table

Step 17: Node n_{10} is not available in the neighbor table of n_{11}.

$$n_{11} \rightarrow C_{1H}$$

EKS_{n11C1H}(not Available(ID_{n10}),ID_{n11}, ID_{C1H}).

Step 18: C_{1H} update location Claim(LCn_{10}) with the-$newLCn_{10}$.

Step 19: C_{1H} remove the existing Finger Print(FPn_{10}) of the node n_{10}.

Step 20: C_{1H} send the readmit accepted message to the node n_{10}.

Step 21: $C_{1H} \rightarrow n_{10}$
EKR_{C1H}(readmitAccept,$ID_{n10,}$ ID_{C1H},KS_{C1Hn10}).

Step 22: Node n_{10} made the neighbor node discovery and find out new neighbors.

Step 23: After that all process continues as in the previous session.

5.2 Movement of the Node to Other Cluster

The node n_{10} belongs to cluster C_{1}. It move from its location to some other location of some other cluster.

5.2.1 Before Movement

5.2.1.1 Remove its Location Information from the Neighbor Nodes

The procedure and algorithm used to remove the location information is same as in the procedure of movement of the node within the cluster.

5.2.1.2 Seeking Locations for Movement

The node n_{10} does not move to the location where ever it wants, but it can move to the location given by the cluster head. Node n_{10} sends a request message to Cluster Head C_{1H} to give the possible locations for its movement. The Cluster Head C_{1H} forwards this request to the base station BS. Ask the BS to return the possible moveable locations

in the cluster C_{2H}.Now BS send the possible locations in the C_{2H} to the Cluster Head C_{1H}. The Cluster Head C_{1H} send various possible locations to be move to the node n_{10}. Now node n_{10} choose any one location given by the C_{1H} for its movement. Before movement node n_{10} inform its new location to be move to the Cluster Head C_{1H}. Now Cluster Head of C_{1H} change the status of node n_{10} from existing to moving. The cluster tables of C_1,C_2, Base Station are described in the Tables 5,6,7.

The node n_{10} moves to the cluster C_2. It must get the readmission from Cluster Head to do its regular process in the cluster. Node n_{10} sends the readmit message to the Cluster Head C_{1H}. The Cluster Head C_{1H} receives this readmit request and checks the given information with the existing information in the cluster table.

Algorithm 3

Seeking locations for movement

Step 1: Seeking locations for movement from Cluster Head $C1_H$

$$n_{10} \rightarrow C_{1H}$$

$EKS_{C1Hn10}(moveLocReq,ID_{n10,}LC_{n10},FP_{n10})$.

Step 2: C_{1H} verifies the request

Table 5. Cluster table of C1

node	Node ID	status	Loc ClaimLC	Ngh node list	Finger Print (FP)	New Loc Claim (N LC)
n10	….	outgoing	…..	…..	…..	…..
n11	….	existing	…..	…..	…..	…..

Table 7. Cluster table of C2

node	Node ID	status	Loc Claim(LC)	ngh node list	Finger Print (FP)	New Loc Claim NLC)
n10	….	incoming	…..	…..	…..	…..

Table 6. Cluster table of base station

cluster	node	Node ID	status	Loc ClaimLC	Ngh node list	Finger Print(FP)	New Loc Claim (NLC)
c1	n10	….	outgoing	…..	…..	…..	…..
c1	n11	….	existing	…..	…..	…..	…..
c2	n10	….	incoming	…..	…..	…..	…..

If existing(ID_{n10}, LC_{n10}, FP_{n10}) ≠ received(ID_{n10}, LC_{n10}, FP_{n10})

Clone node detected.

If existing(ID_{n10}, LC_{n10}, FP_{n10}) = received(ID_{n10}, LC_{n10}, FP_{n10})

Request accepted.

Step 3: $C_{1H} \rightarrow BS$

$EKS_{BSC1H}(moveClusterLoc(C2), ID_{BS},ID_{C1H})$.

Step 4: $BS \rightarrow C_{2H}$

$EKS_{BSC2H}(moveClusterLoc(C2), ID_{BS},ID_{C2H})$.

Step 5: $C_{2H} \rightarrow BS$

EKS_{BSC2H}(moveLocList(C2),ID_{BS},ID_{C2H},<Moveablelocations list >).

Step 6: $BS \rightarrow C_{1H}$

EKS_{BSC1H}(moveLocList(C2),ID_{BS},ID_{C1H},<Moveablelocations list >).

Step 7: $C_{1H} \rightarrow n_{10}$

EKS_{C1Hn10}(moveLocList(C2),ID_{n10},ID_{C1H},<Moveablelocations list >)

Step 8: Node n_{10} Choose any one location given by C_{1H}

Step 9: Node n_{10} inform its new location to C_{1H}

$$n_{10} \rightarrow C_{1H}$$

EKS_{C1Hn10}(newLocInf,$ID_{n10,}LC_{n10}$, $newLC_{n10,}FP_{n10}$).

Step 10: Cluster Head C_{1H} store the new location in the table and update the status as moving

Step 11: $C_{1H} \rightarrow n_1$

EKS_{C1Hn10}((moveAccept(n_{10}),KU_{C2H}, ID_{C2H}) ID_{n10},ID_{C1H}).

5.2.2 *After Movement*

Algorithm 4

Node seeking readmission

Step 1: $n_{10} \rightarrow C_{2H}$

EKU_{C2H}(readmitReq,ID_{n10},$newLC_{n10}$, FP_{n10}, ID_{C2H}).

Step 2: C_{2H} verifies the request

If existing[ID_{n10},FP_{n10},$newLC_{n10}$] ≠ received[$ID_{n10,}$ FP_{n10}, $newLC_{n10}$]

Clone identified

If existing [ID_{n10},FP_{n10},$newLC_{n10}$] = received [ID_{n10},FP_{n10},$newLC_{n10}$]

Readmission accepted.

Step 3: C_{2H} verifies BS for the availability of node n_{10}

$$C_{2H} \rightarrow BS$$

$EKS_{BSC2}(ISavailable(ID_{n10}),ID_{BS},ID_{C2H})$.

Step 4: BS verifies C_{1H} for the availability of node n_{10}

$$BS \rightarrow C_{1H}$$

$EKS_{BSC1H}(ISavailable(ID_{n10}),ID_{BS},ID_{C1H})$.

Step 5: $C1_{H}$ select one of the neighbor of n10 and forward this message

$$C_{1H} \rightarrow n_{11}$$

$EKS_{n11C1H}(ISavailable(ID_{n10}),ID_{n11},ID_{C1H})$.

Node available in the neighbor table

Step 6: $n_{11} \rightarrow C_{1H}$
$EKS_{n11C1H}(available(ID_{n10}),ID_{n11}, ID_{C1H})$.

Step 7: $C_{1H} \rightarrow BS$
$EKS_{BSC1H}(available(ID_{n10}),ID_{BS}, ID_{C1H})$.

Step 8: $BS \rightarrow C_{2H}$
$EKS_{BSC2H}(available(ID_{n10}),ID_{BS}, ID_{C2H})$.

Step 9: Node n_{10} already exist in its old location
Node n_{10} is identified as clone node.
This can be informed to the BS.

Node not available in the neighbor table

Step 10: $n_{11} \rightarrow C_{1H}$
$EKS_{n11C1H}(notAvailable(ID_{n10}),ID_{n11}, ID_{C1H})$.

Step 11: $C_{1H} \rightarrow BS$
$KS_{BSC1H}(notAvailable\ (ID_{n11}),ID_{BS}, ID_{C1H})$.

Step 12: $BS \rightarrow C_{2H}$
$EKS_{BSC2H}(notAvailable(ID_{n11}),ID_{BS}, ID_{C2H})$.

Step 13: $C_{2H} \rightarrow n_{11}$
$EKR_{C2H}(readmitAccept,ID_{n11},ID_{C2H},KS_{C2Hn11})$.

Step 14: Node n_{11} made the neighbor node discovery and Identified new neighbors.

Step 15: After that all process continues as in the previous session.

Suppose existing information do not match with the received information, node n_{10} is identified as clone node. If existing information is same as the received information, then Cluster Head C_{1H} select the neighbor node list of node n_{10} from the cluster table and any old neighbor of the node n_{10} is selected. Suppose node n_{11} is the neighbor of node n_{10}, Cluster Head C_{1H} check whether node n_{10} still exist in the neighbor table of n_{11}.

6. Results and Discussion

The proposed protocol has been tested with Castalia 3.2 simulator that runs on Omnet++. The experiment has been done with the 10,000 sensor nodes deployed in 50 m communication range shown in the Figure 1. The average number clusters in the network may be 100 to 500. Each cluster has 3 to 40 sensor nodes.

The performance of CBCD protocol has been compared with the existing EDD and XED protocols based on the following evaluation parameters.

6.1 Clone Detection Time

The EDD protocol takes 72 seconds to detect a clone node, the XED protocol takes 81 seconds, whereas the CBCD protocol takes only 52 seconds for the 1000 nodes. In the same way for 10,000 nodes EDD takes 121 seconds, XED takes 145 seconds, whereas the CBCD protocols takes only 94 seconds. Compare with other existing protocols CBCD protocol takes very minimum clone detection time. This is described in the following Table 8 and Figure 2.

6.2 Total Number of Clones Detected

The EDD protocol detects 45 clones, the XED protocol detects 42 clone nodes, whereas the proposed CBCD

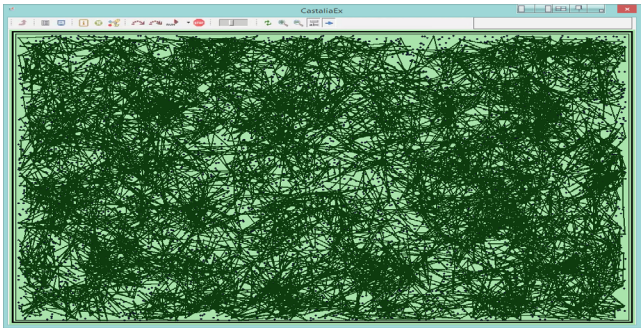

Figure 1. Node deployment and cluster formation of 10,000 sensor nodes.

Table 8. Clone detection time

Nodes	EDD	XED	CBCD
1000	72	81	52
2000	78	92	56
3000	84	96	61
4000	88	102	61
5000	95	112	71
6000	99	117	76
7000	106	127	77
8000	113	135	83
9000	115	136	89
10000	121	145	94

protocol detects the maximum of 50 clone nodes for 1,000 nodes. In the same way 10,000 nodes EDD detects 44 clones, XED detects 42 clones, whereas CBCD protocol detects the maximum of 50 clone nodes. Compare with other existing protocols CBCD protocol detects more number of clones and it achieves 100% clone detection many times. The total number of clones detected for 1,000 to 10,000 nodes are described in the following Table 9 and Figure 3.

6.3 Memory Consumption

The average memory consumption of CBCD protocol is very low, compared with other protocols. While the EDD protocol uses 3606 bytes and the XED protocol uses 3367 bytes, our CBCD protocol uses only 3162 bytes for 1000 nodes. In the same way for 10,000 nodes EDD protocol uses 3853 bytes, XED protocol uses 3578 bytes and CBCD protocol 3395 bytes only. The memory consumption of all the three protocols for 1000 to 10,000 nodes are illustrated in Table 10 and Figure 4.

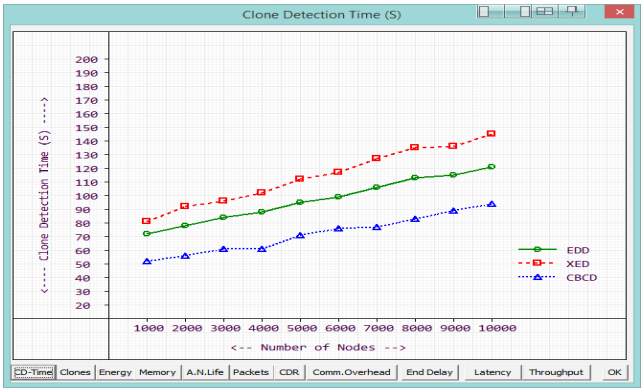

Figure 2. Clone detection time.

Table 9. Number of clones detected

Nodes	EDD	XED	CBCD
1000	45	42	50
2000	44	40	49
3000	44	41	48
4000	45	41	50
5000	45	41	48
6000	44	42	49
7000	44	41	49
8000	44	43	48
9000	45	43	48
10000	44	42	50

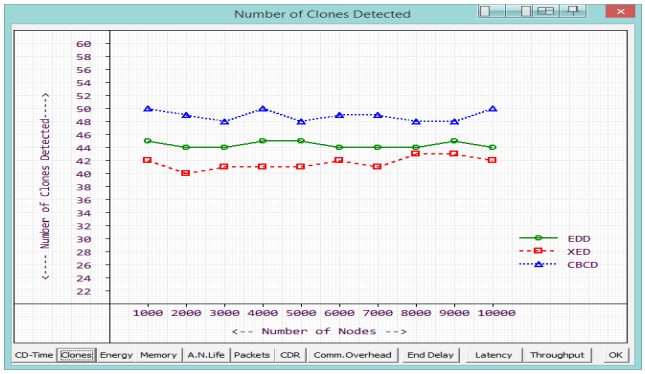

Figure 3. Number of clones detected.

Table 10. Memory consumption

Nodes	EDD	XED	CBCD
1000	**3606**	**3367**	**3162**
2000	**3636**	**3347**	**3165**
3000	**3639**	**3440**	**3208**
4000	**3665**	**3464**	**3216**
5000	**3710**	**3463**	**3255**
6000	**3749**	**3514**	**3253**
7000	**3753**	**3535**	**3333**
8000	**3796**	**3588**	**3375**
9000	**3812**	**3616**	**3367**
10000	**3853**	**3578**	**3395**

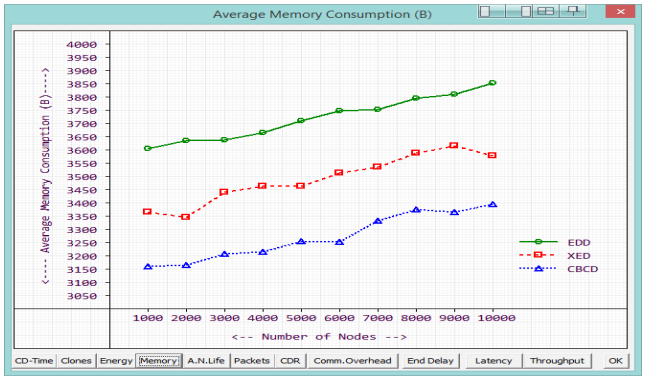

Figure 4. Memory consumption.

7. Conclusion

In this paper we propose a Cluster Based Clone Detection (CBCD) Protocol for detecting clone nodes in Mobile sensor networks. The key management presented here can be used for both public and symmetric key cryptography of cluster based WSN. The Finger Print can be used for secure communication between sensor nodes, sensor node and

cluster head, cluster head and base station. In our experiment 10,000 sensor nodes are deployed in the field and 50 clone nodes tries to enter into the network. Compared with XED, EDD protocols CBCD protocol takes very minimum time to detect a clone node. The proposed protocol has highest percentage of clone detection ratio than other protocols. It has very minimum memory consumption compare to other mobile based clone detection protocols.

8. References

1. Mishra AK. Node replica detection in wireless sensor networks. National Institute of Technology; Rourkela, India. 2014.
2. Wang Y, Attebury G, Ramamurthy B. A survey of security issues in wireless sensor networks. IEEE Communications Survey and Tutorials. 2006; 8(2):1–23.
3. Ahamed MR. Protecting wireless sensor networks from internal attacks. Austrslia: University of Canberra; 2014.
4. Kraub C. Handling insider attacks in wireless sensor networks. Darmstadt: Technische University; 2010.
5. Padmavathi G, Priya DS. A survey of attacks security mechanisms and challenges in wireless sensor networks. International Journal of Computer Science and Information Security. 2009; 4(1,2):1–9.
6. Ahmed MR, Huang X, Cui H. A novel two-stage algorithm protecting internal attack from WSNs. IJCNC. 2013 Jan; 5(1):97–116.
7. Ho JW, Lin D, Wright M, Das SK. Distributed detection of replicas with deployment knowledge in wireless sensor networks. Elsevier; 2009 Mar. p. 1–33.
8. Mohanty P, Panigrahi S, Sarma N, Satapathy SS. Security issues in wireless sensor network data gathering protocols: A survey. Journal of Theoretical and Applied Information Technology. 2010; 13(1/2):14–27.
9. Zhu B, Setia S, Jajodia S, Roy S, Wang L. Localized multicast: Efficient and distributed replica detection in large-scale sensor networks. IEEE Transactions on Mobile Computing. 2010; 9(7):913–26.
10. Shaukat HR, Hashim F, Sali A, Fadlee M. Node replication attacks in mobile wireless sensor network: A survey. International Journal of Distributed Sensor Networks. 2014; 14.
11. Ho JW, Wright M, Das SK. Fast detection of replica node attacks in mobile sensor networks using sequential analysis. Proc IEEE INFOCOM; Rio de Janeiro. 2009 Apr. p. 1773–81.
12. Deng XM, Xiong Y. A new protocol for the detection of node replication attacks in mobile wireless sensor network. Journal of Computer Science and Technology. 2011; 26(4):732–43.
13. Yu CM, Lu CS, Kuo SY. Mobile sensor network resilient against node replication attacks. IEEE 5th Annual Communications Society Conference on Sensor Mesh and Ad Hoc Communications Networks; San Francisco, CA. 2008. p. 597–99.
14. Ko LC, Chen HY, Lin GR. A neighbor-based detection scheme for wireless sensor networks against node replications attacks. IEEE International Conference on Ultra Modern Telecommunications and Workshops; St. Petersburg. 2009. p. 1–6.
15. Yu CM, Lu CS, Kuo SY. Efficient and distributed detection of node replication attacks in mobile sensor networks. 2009 IEEE 70th Vehicular Technology Conference Fall VTC-2009; Anchorage, AK. 2009. p. 1–5.
16. Conti M, DiPietro R, Mancini LV, Mei A. Emergent properties detection of the node captures attack in mobile wireless sensor networks. Proceeding of the 1st ACM Conference on Wireless Network Security; Alexandria, USA. 2008. p. 214–9.
17. Deng X, Xiong Y, Chen D. Mobility assisted detection of the replication attacks in mobile wireless sensor networks. Proceedings of the 6th Annual IEEE international Conference on Wireless and Mobile Computing, Networking and Communications; Niagara Falls, ON. 2010 Oct. p. 225–32.
18. Lou Y, Zhang Y, Liu S. Single hop detection of node clone attacks in mobile wireless sensor networks. Proceedings of the International Workshop on Information and Electronics Engineering (IWIEE); 2012. p. 2798–803.
19. Wang LM, Shi Y. Patrol detection for replica attacks on wireless sensor networks. Sensors. 2011; 11(3):2496–504.
20. Khan WZ, Aalsalem MY, Saad MNBM, Xiang Y. Detection and mitigation of node replication attacks in wireless sensor networks: A survey. International Journal of Distributed Sensor Networks. 2013; 1–22.
21. Kumar AM, Turuk AS. A comparative analysis of node replica detection schemes in wireless sensor networks. Journal of Network and Computer Applications. 2016.
22. Saranya V, Matheswari N, Punidha R, Soundarya M. Tracking dynamic target in wireless sensor networks. Indian Journal of Science and Technology. 2016 Jan; 9(1).
23. Ahmed MR, Huang X, Sharma D, Cui H. Wireless sensor network: characteristics and architectures. World Academy of science, Engineering and Technology. Penang, Malaysia. 2012; 72:660–3.
24. Vijayan K, Raaza A. A novel cluster arrangement energy efficient routing protocol for wireless sensor networks. Indian Journal of science and Technology. 2016 Jan; 9(2).
25. Abdallah W, Boudriga N, Kim D, Sunshin A. An efficient and scalable key management mechanism for wireless sensor networks. 17th International Conference on Advanced Communication Technology (ICACT); 2015.