Computers, Materials & Continua
DOI:10.32604/cmc.2022.020920
Article

Tech Science Press

Intelligent Multilevel Node Authentication in Mobile Computing Using Clone Node

Neha Malhotra[1,2,*] and Manju Bala[3]

[1]I.K.Gujral Punjab Technical University, Kapurthala, 144603, India
[2]Lovely Professional University, Phagwara, 144411, India
[3]Khalsa College of Engineering and Technology, Amritsar, 143001, India
[*]Corresponding Author: Neha Malhotra. Email: mneha8789@gmail.com
Received: 14 June 2021; Accepted: 30 July 2021

Abstract: Nodes in a mobile computing system are vulnerable to clone attacks due to their mobility. In such attacks, an adversary accesses a few network nodes, generates replication, then inserts this replication into the network, potentially resulting in numerous internal network attacks. Most existing techniques use a central base station, which introduces several difficulties into the system due to the network's reliance on a single point, while other ways generate more overhead while jeopardising network lifetime. In this research, an intelligent double hashing-based clone node identification scheme was used, which reduces communication and memory costs while performing the clone detection procedure. The approach works in two stages: in the first, the network is deployed using an intelligent double hashing procedure to avoid any network collisions and then in the second, the clone node identification procedure searches for any clone node in the network. This first phase verifies the node prior to network deployment, and then, whenever a node wants to interact, it executes the second level of authentication. End-to-end delay, which is bound to increase owing to the injection of clone nodes, and packet loss, which is reduced by the double hashing technique, were used to evaluate the performance of the aforementioned approach.

Keywords: Node authentication; clone node; mobile computing; double hashing; fault tolerance

1 Introduction

Mobile computing is extensively used in many industries where a fast and efficient response is essential. Smart schools and workplaces, the health care industry, weather forecasting, the military, and a variety of monitoring environments are all examples of such applications. The use of mobile computing in the system has numerous constraints that must be addressed, such as the nodes in such systems having resource issues such as low cost, battery, bandwidth, memory, and so on, which must be handled in such a way that it does not directly affect network health. Nodes in such a network communicate with one another and with a base station, emphasising the importance of all nodes' legitimacy [1]. Because nodes are deployed in an unmanaged, hostile,

and open environment, adversaries of many types can target them by initiating various attacks, such as denial of service, flooding, replication attacks, and so on [2]. The network's nodes are likely to be compromised [3], as adversaries often acquire the target node's private information and credentials in order to impact a large number of other nodes in the network [4], as well as utilise this information to build a variety of clone nodes under their control. The adversary's clones have the identical node id and other acquired information to do malicious activities in the system.

These clone nodes may act like regular nodes, making them difficult to identify in the network. However, many security problems arise as a result of attacks on vulnerable nodes and inadequate resources [5]. As a result, much research has been conducted in the literature to improve clone node detection methods in order to protect the system against clone attacks. These techniques have been classified as centralised and distributed, witness-based or not, location-dependent or not. The main contribution of the work is that, when the network has been deployed using the double hashing method, if a collision occurs, which means that the generated index is occupied by another node in the network, it can be easily resolved by calculating the index using the second hash function. A clone node can be easily located in the network with this hashing searching technique. This solution is not dependent completely on central base station, which resolves numerous issues into the system owing to the network's reliance on a single point, while improving additional overhead by not compromising network lifespan.

Due to the increase in memory and space complexity in the existing methods of node authentication, the majority of techniques raise the overhead while compromising the network lifespan. The suggested node authentication method seeks to reduce communication and memory costs. These predicted changes resulted in the usage of the suggested protocol DHFNA, which shows that as mobility support improves, the number of mobile devices grows over time in the form of mobile nodes linked in a network. Also, each mobile node uses a double hashing technique to prove its sustainability, which is the most important need of any mobile network, that all nodes in the network be verified to avoid malicious actions.

The remaining work is organised as follows: Section 2 represents related work in this field of clone node detection. Section 3 represents the proposed work's system model and network model. Section 4 represents the algorithm used for network deployment and clone node detection. Section 5 represents the proposed work's simulation, and section 6 represents the proposed work's comparison.

2 Related Work

Several clone node detection techniques and fault detection techniques have been proposed in the literature, with the majority of them focusing on the node's location, because each node in the network has a unique node id and spatial coordinates, and if two different nodes in the network claim the same node id but with different spatial coordinates, it clearly indicates the presence of a clone node and its attack. In [6] replication attacks were detected using nearby social signatures. Witness nodes [7] need more battery power, memory, energy, and other resources than normal nodes. Furthermore, the witness node is the most vulnerable in the network and may be readily captured by the attacker [8]. The majority of suggested methods in the literature use witness finding methodology, such as RM and LSM [9,10], RED, Random walk-based protocols, and so on. This approach is based on nodes signing in and transferring geographical coordinates [11] introduces a centralised detection technique based on a random key. The technique focuses on the intended pattern in the pre-distribution of a random key to all accessible nodes, and if the

pre-set key count exceeds a certain threshold, that node is identified as a clone node. The suggested approach includes a bloom filter for collecting important use data. The authors of [12] use a method based on intersection and union on a network subset that gathers adjacent information in a distributed manner and executes a SET operation on base station nodes. This method may identify internal assaults resulting from evidential analysis. This may be done by combining several pieces of evidence gathered from suffering nodes, i.e., nodes that were attacked. The authors of [13] provide two methods for detecting replica nodes. Whereas in the first phase, a node-to-network broadcast occurs and transmits the position of the node in the network. The authors utilised the localization method in [14], where clone node identification occurs in clusters utilising RFID and the localization technique. The authors propose two approaches for detecting cloned nodes: one based on RFID to authenticate the node and another based on geolocation techniques to identify the cloned node. When compared to previous methods, the suggested approach has shorter transmission latency and improved detection [15] employs a key distribution mechanism in which a node must authenticate itself in order to start any communication or create a connection using a key. The base station monitors the use of the keys [16] in use on a regular and random basis. Various methods for combating clone node attacks have also been proposed, including network-based detection, centralised detection, and distributed detection approaches. Key applications, base stations, social neighbourhood signatures, cluster head, zone, and neighbouring methods were among the sub approaches. The authors provide a concise overview of network security problems such as key management, node authentication, and safe routing [17]. Tab. 1 provides a comprehensive comparison.

Table 1: Comparison of the existing clone detection schemes

Reference	Technique used	Detection Approach		Communication cost	Memory
		Distributed	Centralized		
[6]	Deployment order based	Yes		$< O(n\sqrt{n})$	$< O(\sqrt{n})$
[7]	RED		Yes	$O(c\sqrt{n})$	O(c)
[8]	Distributed	Yes		$O(n^2)$	$O(\sqrt{n})$
[9]	Randomized Multicast	Yes		$O(n^2)$	$O(\sqrt{n})$
[10]	Random key	Yes		$O(n^2)$	O(d)
[11]	Group Distributed	Yes		$O(n\sqrt{n})$	$O(\sqrt{n})$
[12]	SET		Yes	O(n)	O(d)
[13]	Node to network broadcast		Yes	$O(n^2)$	O(d)
[14]	Deterministic multicast	Yes		$(O\ w\ ln\ w\sqrt{n}/d)$	O(w)
[15]	LSM		Yes	$O(n\sqrt{n})$	$O(\sqrt{n})$
[16]	TBNCD	Yes		O(n)	O(n)

Where, n- number of nodes, d-degree of neighboring nodes, w- number of witness nodes and c- communication radius. The proposed technique is completely distributed in nature where all the nodes in the network participate so that the workload cannot be on one central node. This scheme will work in two phases network expansion phase and a clone detection phase.

3 System Model and Assumptions

The base station is considered to be at the network's core in the provided architecture. The network nodes have been deployed at random around the base station. Each node in the network has a unique id and a list of its one-hop neighbours [18]. Moreover, all nodes in the network are aware of their geographical locations. Also, once the deployment phase is completed, nodes with new ids are not allowed to join the network [19,20]. Communication between a base station and other network nodes takes place through intermediary nodes. All nodes transmit their observation data to the base station on a regular basis [21,22]. The network model is shown in Fig. 1. For the adversary paradigm, the primary goal is to destroy or control the network area. This technique is less expensive than inserting duplicate nodes into the network for testing. The attacker compromises a few nodes before replicating its data in mass. Adversaries in the suggested system tend to control the network partly by attacking its few accessible nodes. In the example, the attacker has built a duplicate node and put it in the desired position. An attacker may launch a variety of attacks on the network using these newly generated clone nodes [23]. The clone nodes that have been deployed in various places seem to be normal, with legitimate node ids.

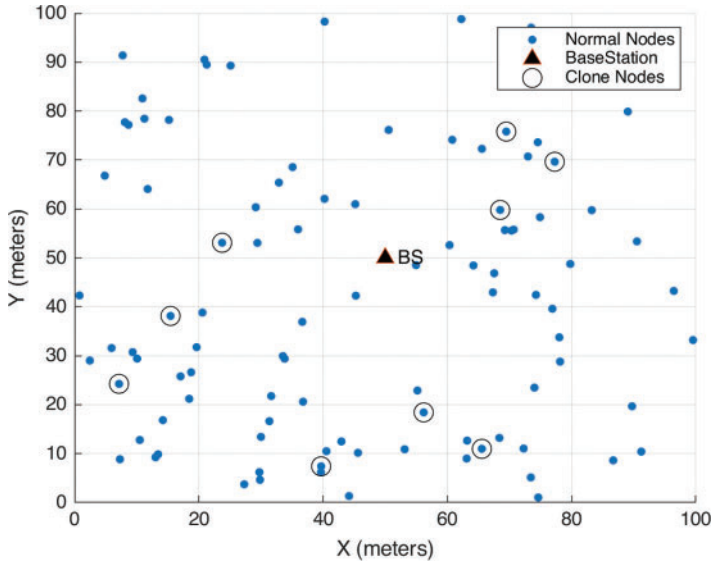

Figure 1: Network model

Assumptions for the network and adversary model have been given below:

- Processes/nodes in the mobile computing environment are uniformly distributed.
- Each process/node has been identified by its unique id which is static.
- An adversary cannot create an ID for the clone nodes
- The adversary could capture a limited amount of normal nodes.

4 Proposed Work

In the proposed method, the node-id is utilised as a key index to build the hash table of a network node. A unique hash function was used to associate a node id with a specific key in order to access the nodes in the network as quickly as possible. The efficiency of node access is

completely dependent on the type of hash function employed, which does not return zero. The suggested approach accomplishes the following tasks:

a. Creates a hash table, which is an array that stores references to nodes. If no such node id exists for the evaluated hash function value, this hash table returns a null entry.
b. A hash function is a function that converts arbitrary-sized data to fixed-size values. In a hash table, these values serve as network node indexes.
c. If the hash function computes a value that is already held by another node in the network, this is referred to as a collision, and it will be addressed by computing the second hash function to create a new address for the collided node.

4.1 Network Deployment Phase

Nodes available in the network have their unique node id which is used for indexing in the hash table. With the given list of nodes in the network, its index in hash table has been computed using (1) and methodology has been explained in Fig. 2:

$$H_1(Key) = key \,\%\, m \tag{1}$$

where 'm' is the size of the hash table, 'key' is the node and 'H_1' is the hash function. In case if the value computed in Eq. (1) is already occupied by another node then it results in the collision which can be further dealt with using (2):

$$H_2(Key) = Prime_{num} - (key \,\%\, Prime_{num}) \tag{2}$$

where H_2 is the second hash function computed to resolve collisions, in case the generated index position in Eq. (1) is already occupied by any other value. Whenever there is a collision, the node will be inserted at the index calculated using the below mentioned for (3):

$$H_1(Key) + i \times H_2(Key) \,\%\, Tbl\_size \tag{3}$$

Algorithm Phase 1: Network Deployment Phase

Let 'K' is the node to be inserted in the network, 'M' is the size of the hashTable, 'p' is the Prime number less than table size number, the probe is the value returned by the first hash function, offset is the value returned by the second hash function. A probe is used to get the index location first hash function is called.

```
1. HashTable = containers. Map ('KeyType','int32','ValueType','char');
2. M = length (nodes. Node);
3. p = primes(M);
4. Prime = p (end);
5. for K = 1:M
6.       probe = mod(K,M);
7.       if (~HashTable.isKey(probe))
8.             HashTable(probe) = ('Node id = %i', K);
9.       else
10.       offset = Prime-mod(K, Prime);
11.       while(~HashTable.isKey(probe))
12.             probe = mod((probe + offset), M);
```

(Continued)

```
13.        end
14.        HashTable(probe) = ('Node id = %i', K);
15.     end
16. end
17. secret = Prime;
18. end
```

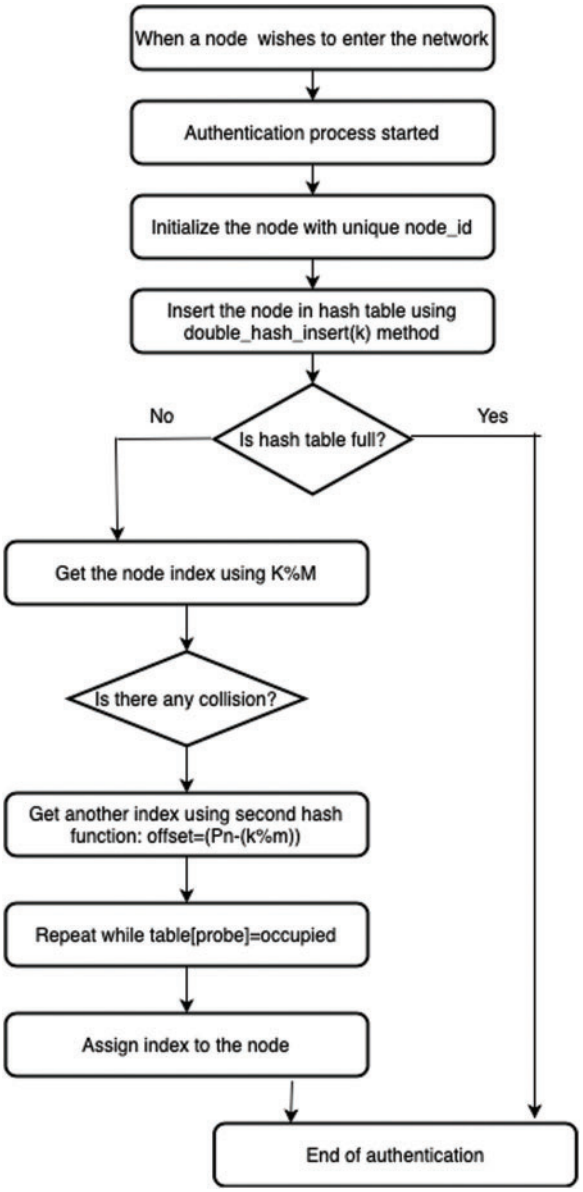

Figure 2: Step by step process for network deployment

Once all the nodes in the network have been deployed, then each node exchanges its node id and spatial coordinates with each other to store the value of its neighboring nodes.

4.2 Clone Node Detection Phase

The suggested work's second step focuses on identifying the clone in the network. This second phase identifies the existence of duplicate/clone nodes in the network as soon as the network is installed in the first phase. If the cloned node is discovered, this step removes it from the network as soon as it is discovered. Nodes in the mobile computing environment interact with one another in a distributed fashion [24–26]. If a node wishes to interact with the base station by transmitting the required data, the base station verifies the node's authenticity by calculating its index using the hash function. If the calculated value of the node is the same as the original index value, the node is authenticated [27,28]. If the value does not match, the node is not genuine and may be a clone node. In such a scenario, the receiver generates a warning message indicating that the sender node may be a clone node, followed by the termination of the cloned node [29,30] from the network, as shown in Fig. 3:

Algorithm Phase 2: Clone Node Detection Phase

In this phase, when any node (Si) wants to send data to the base station (Ti), the following steps will be repeated until the message does not reach the destination node. Also, a warning message is triggered pointing out that the node is a clone node so that the termination procedure can be triggered.

```
1. M = length(nodes.node);
2. for K = 1:M
3.       probe = mod(K,M);
4.       offset = Prime - mod(K, Prime);
5.       Si = 0;
6.       while(HashTable(mod((probe + Si × offset),M)) ~= K )
7.             break
8.             if(HashTable(mod((probe + Si × offset),M)) == −1)
9.                break
10.            end
11.            Si = Si + 1;
12.      end
13. End
```

5 Simulation Parameters

The proposed distributed technique DHFNA has been simulated on MATLAB using the parameters depicted in Tab. 2 under different scenarios like, scenario 1, when the network has been attacked by the network with no detection technique implemented, scenario 2, when the proposed DHFNA has been implemented but there are no attacks in the network, scenario 3, when the proposed DHFNA has been implemented in the presence of attacks in the network. Following parameters have been considered in Tab. 2 to carry out the same:

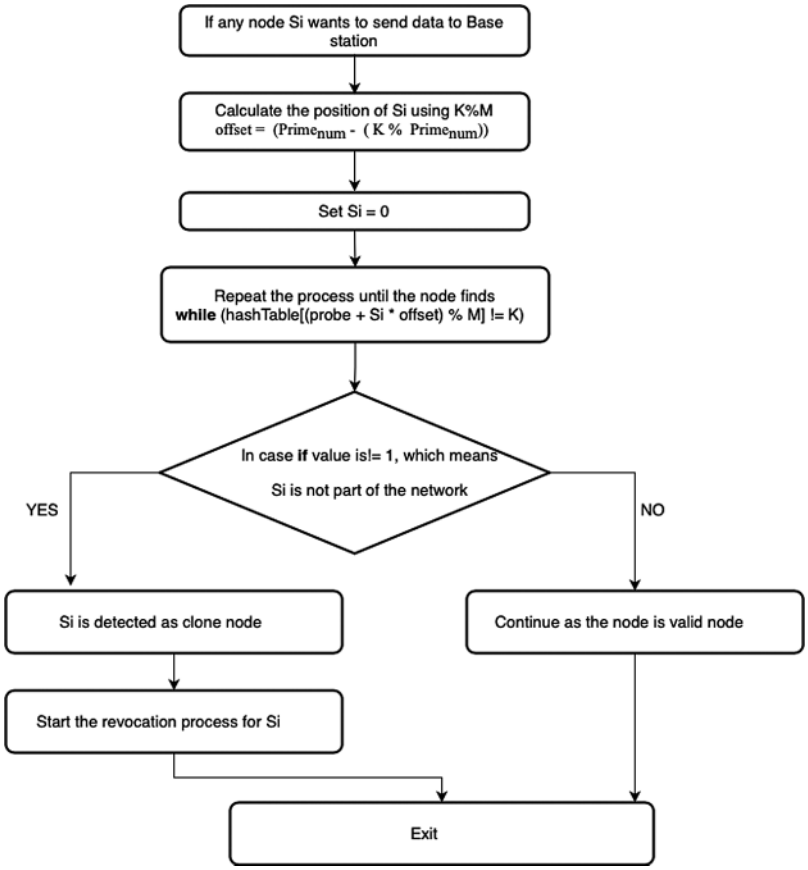

Figure 3: Steps for clone detection phase

6 Performance Evaluation

The performance of the DHFNA has been evaluated using the following parameters:

a. End to end delay: It refers to the transmission time from sender to receiver and is defined in below (4):

$$Delay = \frac{Sum\ of\ delayed\ packets}{Total\ received\ packets} \tag{4}$$

Table 2: Simulation parameters

Parameters	Values
Size of packet	512 B
Total number of nodes	100
Range of transmission	150 m
Time for simulation	10 s
Energy (at the initial point)	1 J

Fig. 4 depicts how the transmission rate changes in the three scenarios described above. It clearly shows that when the network has been attacked in the absence of any detection protocol, the average end-to-end delay is 0.8391, which is less than the end-to-end delay when the network has been attacked in the presence of DHFNA, which is equivalent to 3.92, and when the network has not been attacked but the protocol has been implemented, end-to-end delay comes out to be 3.75. When clone nodes are introduced into the network, they delay the real time of message delivery from sender to recipient while also causing additional packet losses. As a result, end-to-end delays are unavoidable.

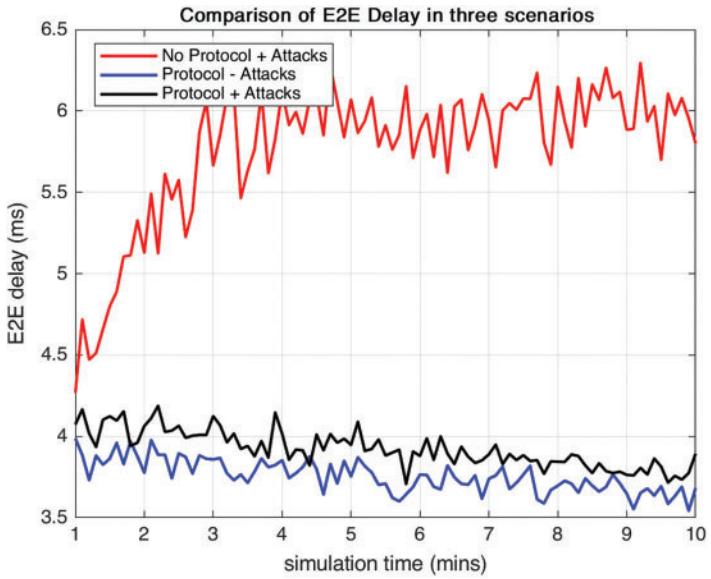

Figure 4: End to end delay

b. Packet loss:

It signifies the number of packets lost before reaching out to the target is the packet loss ratio depicted in Fig. 5. The said result signifies that there are on an average 1949 packets lost in the case of a protocol implemented without attacks, on an average 331 packets are lost when the network has been attacked in the presence of DHFNA, and in the third scenario, where the network has been attacked in the absence of the proposed protocol, the average number of

packets lost is 440. When clone nodes are available in the network, these nodes tend to behave as legitimate nodes and restrict sending data to the desired receiver, which eventually results in packet loss.

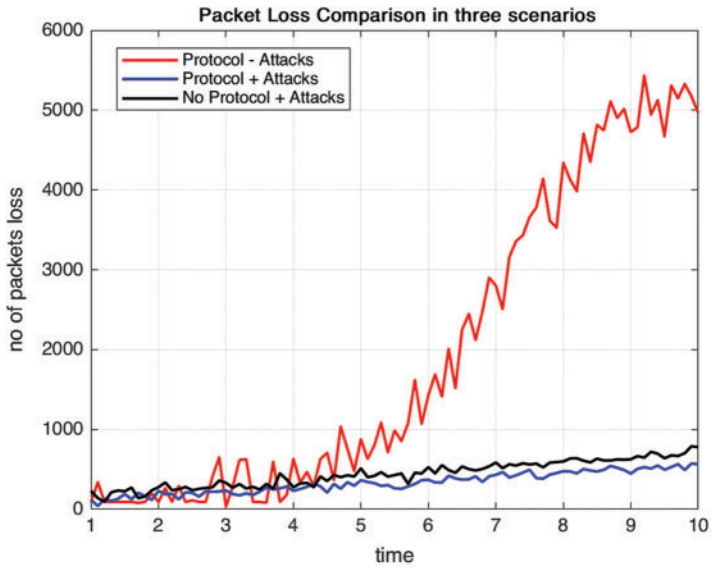

Figure 5: Packet loss in three scenarios

c. Packet Delivery Ratio:

It is the ratio of the number of packets sent by the source node to the number of packets received by the target node [31] as depicted in Fig. 6. It has been calculated using (5):

$$Packet\ Delivery\ Ratio = \frac{Packets\ Received}{Packets\ Sent} \times 100 \tag{5}$$

The packet delivery ratio in scenario 1 when the network has been implemented with a protocol but attacks are not present, is 63% on an average, and this ratio is 83% in the case when the network has not been deployed with protocol but has been attacked, which is increased to 88% in case of implemented protocol along with attacks. The results signify that the delivery ratio improves with time when the DHFNA protocol has been implemented.

d. Energy consumption:

It refers to the amount of energy consumed during the entire process and is represented in Fig. 7. The consumption of energy of a node in a mobile network is defined as the sum of the processor energy (E_P), mobility energy (EMob), and transceiver energy (E_{Tr}) [16] which is represented in (6), where E_C is defined as the total energy consumed and the energy of the processor is defined by (7) and where $E_{Pzstate}$ is the processor state energy consumption and $E_{Pchange}$ is the energy consumed in state transition. Transceiver energy consumption is defined by (8):

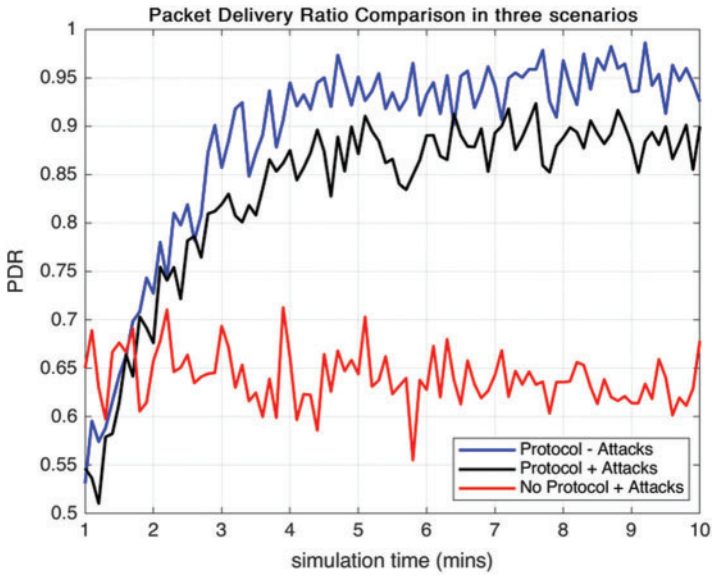

Figure 6: : Packet delivery ratio

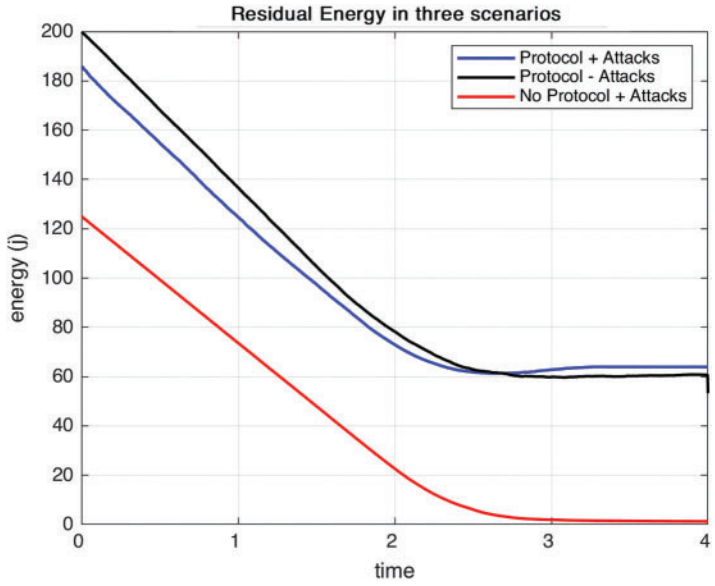

Figure 7: Residual energy in three scenarios

$$E_{\mathrm{C}} = E_{\mathrm{P}} + E_{\mathrm{Tr}} + E_{\mathrm{Mob}} \tag{6}$$

$$E_{P} = E_{\mathrm{Pzstate}} + E_{\mathrm{Pchange}} \tag{7}$$

$$E_{\mathrm{Tr}} = E_{\mathrm{Ts}} + E_{\mathrm{Tc}} \tag{8}$$

where E_{Ts} is the energy of transition state and E_{Tc} is the transition change energy consumed [32]. Energy consumed while the nodes are in move i.e., in mobility is represented by (9) and where ec (Ds_i,Tm_i) is defined as the amount of energy consumed while the movement for a distance (Ds_i) and in time (Tm_i). Nodes in the mobile network tend to consume energy to perform all their operations like data transmission and computation, eventually which results in a decrease in the residual energy with time.

$$E_{\text{Mob}} = \sum_{i=1}^{m} ec(Ds_i, Tm_i) \tag{9}$$

Values of Eqs. (6)–(8) have been merged and residual energy values have been obtained which depicts the residual energy in millijoules (mJ) left with each node after the end of simulation time.

The proposed work has been compared with the existing techniques in terms of memory and communication expenditure. To carry out the comparison, there is a need to calculate the consumption in the case of DHFNA. The comparison of all the available techniques has been tabulated in Tab. 3. The cost required to carry out the node finding and saving operation using the double hashing technique gives an O(1) complexity in its best cases and average cases and an O (n) complexity in its worst case scenario. The best case is the scenario in which all the operations like searching for a node, removing and inserting a new node takes place with no collisions at all. The average case is the scenario in which all such operations are done with few collisions or probes. The worst case is the scenario in which all such operations are done with collisions in each step, mean probing is required for all the nodes.

Table 3: Comparison of DHFNA with other methods

Method	Communication	Memory
RAWL [17]	$O(\sqrt{n}\ \log n)$	$O(\sqrt{n}\ \log n)$
TRAWL [17]	$O(\sqrt{n}\ \log n)$	O(1)
TBCND [18]	O(n)	O(n)
DHFNA	O(1)	O(1)
LSM [13]	$O(n\sqrt{n}\ )$	$O(\sqrt{n})$
RSM [8]	O(log n)	O(log n)

Communication expenditure: it can be calculated using the summation of communication from source to reporter and from reporter to witness [33]. This communication cost from source to reporting node can be calculated using the below mentioned (10) and communication cost from reporting node to witness node (W_N)can be calculated using (11). Using (8) and (9), the total communication cost of one node can be calculated in (12). The total communication cost of the network will be calculated using (13) which is generated by multiplying the number of nodes (n) to (10) which gets reduced and the same has been depicted in Fig. 8:

$$C_1 = M_S \times E \tag{10}$$

$$C_2 = W_N \times M_S \times E \tag{11}$$

$$C_T = M_S \times E + W_N \times M_S \times E \tag{12}$$

$$C_T = n \times (M_S \times E + W_N \times M_S \times E) \tag{13}$$

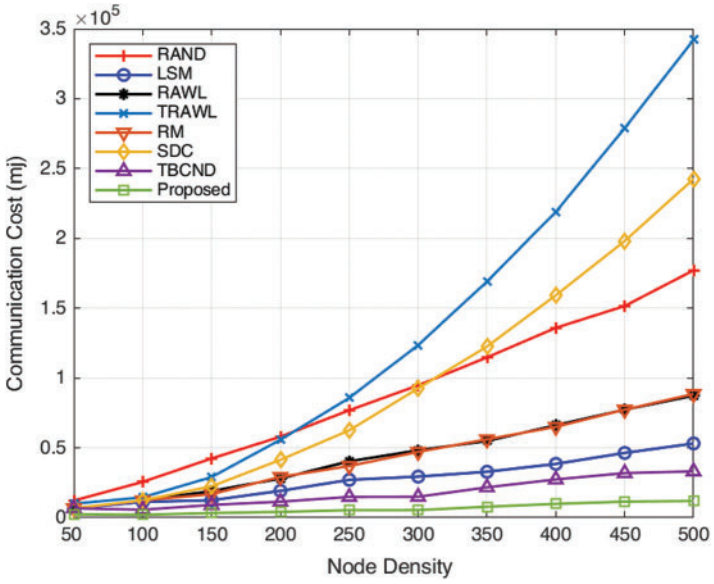

Figure 8: Communication cost

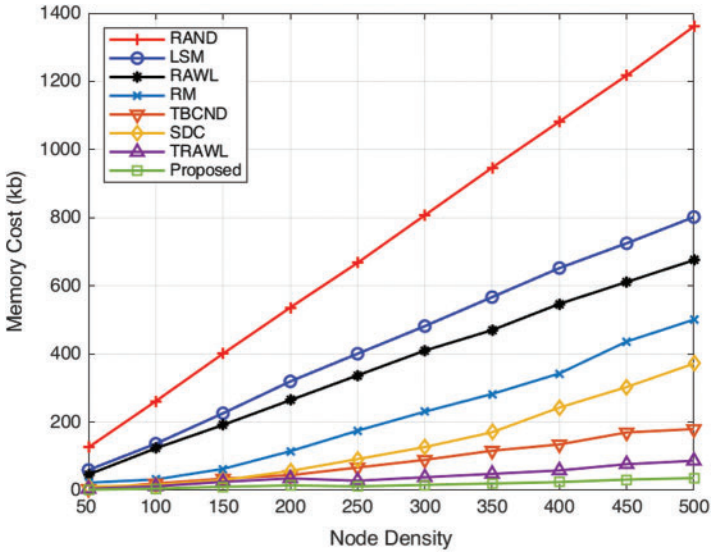

Figure 9: Memory cost comparison

Cost of memory consumed: Using the double hashing method, the memory cost per node is given and represented in Fig. 9. The values in the table signify that the proposed DHFNA method gives on an average 17.56 memory cost which is far less than the existing methods. Detection rate of attacker nodes and detected nodes has been given as per the values depicted in Tab. 4 and the details of the values used for the simulation of memory cost checking in all the existing methods have been tabulated in Tab. 5.

Table 4: Detection rate of nodes

Attacker nodes	Detected nodes	Detection rate
5	5	100%
10	10	100%
15	14	93%
20	19	95%
25	23	92%
30	30	100%
35	33	94%
40	38	95%
Mean Accuracy		96.54%

Table 5: Energy comparison

Nodes	RAND	LSM	RAWL	RM	TBCND	SDC	TRAWL	Proposed
50	125.98	59.549	46.248	22.14	4.0515	10.45	4.3827	1.873
100	261.41	136.48	123.15	31.775	19.85	15.45	11.54	4.9316
200	536.35	319.51	264.42	114.59	44.158	56.466	34.807	14.875
300	807.26	481.77	409.91	230.7	89.183	126.69	38.145	16.301
400	1082.2	652.18	546.9	342.67	134.44	242.76	58.198	24.871
500	1361.3	801.91	675.58	500.8	179.64	371.61	86.559	36.991

7 Conclusion

Mobile computing systems are susceptible to a wide range of attacks from attackers. The deployment of the network in an authentic manner is the most essential requirement for making the system fault-tolerant. Each node in this study's network has been installed using the DHFNA method, which is responsible for authentication of the nodes in the network. Using double hashing, the first phase of this protocol is used to deploy nodes in the network, and the second phase is used to detect the existence of clone nodes in the network. If the cloned node is discovered, it is not permitted to stay on the network, and the base station and other nodes are alerted to its existence via the network's security system. The clone node search is also carried out by the suggested work in the next phase. Evaluation results proves the performance of this approach is better in terms of communication and memory cost.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no interest in reporting regarding the present study.

References

[1] H. Radhappa, L. Pan, J. Xi Zheng and S. Wen, "Practical overview of security issues in wireless sensor network applications," *International Journal of Computers and Applications*, vol. 40, no. 4, pp. 202–213, 2018.

[2] X. Huang, R. F. Rojas, A. C. Madoc and D. Ahmad, "Evidentiary assessment for protecting WSNs from internal attacks in real-time," *International Journal of Computers and Applications*, vol. 39, no. 1, pp. 1–8, 2017.

[3] W. Ametepe, C. Wang, S. K. Ocansey, X. Li and F. Hussain, "Data provenance collection and security in a distributed environment: A survey," *International Journal of Computers and Applications*, vol. 43, no. 1, pp. 11–25, 2021.

[4] W. Z. Khan, M. Y. Aalsalem, M. N. B. M. Saad and Y. Xiang, "Detection and mitigation of node replication attacks in wireless sensor networks: A survey," *International Journal of Distributed Sensor Networks*, vol. 9, no. 5, p. 149023, pp. 1–22, 2013.

[5] L. S. Sindhuja and G. Padmavathi, "Replica node detection using enhanced single hop detection with clonal selection algorithm in mobile wireless sensor networks," *Journal of Computer Networks and Communications*, vol. 2016, article. 1620343, pp. 1–13, 2016.

[6] Z. Zhang, S. Luo, H. Zhu and Y. Xin, "A clone detection algorithm with low resource expenditure for wireless sensor networks," *Journal of Sensors*, vol. 2018, article. 4396381, pp. 1–16, 2018.

[7] M. Conti, R. Di Pietro and A. Spognardi, "Clone wars: Distributed detection of clone attacks in mobile WSNs," *Journal of Computer and System Sciences*, vol. 80, no. 3, pp. 654–669, 2014.

[8] S. Anitha, P. Jayanthi and V. Chandrasekaran, "An intelligent based healthcare security monitoring schemes for detection of node replication attack in wireless sensor networks," *Measurement*, vol. 167, p. 108272, 2021.

[9] V. Manjula and C. Chellappan, "The replication attack in wireless sensor networks: analysis and defenses," in *Int. Conf. on Computer Science and Information Technology*, vol. 132 CCIS, no. 2, pp. 169–178, 2011.

[10] Z. Wang, C. Zhou and Y. Liu, "Efficient hybrid detection of node replication attacks in mobile sensor networks," *Mobile Information Systems*, vol. 2017, article. 8636379, pp. 1–13, 2017.

[11] R. Brooks, P. Y. Govindaraju, M. Pirretti, N. Vijaykrishnan and M. T. Kandemir, "On the detection of clones in sensor networks using random key predistribution," in *IEEE transactions on systems, man, and cybernetics*," *Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1246–1258, 2007.

[12] W. T. Zhu, J. Zhou, R. H. Deng and F. Bao, "Detecting node replication attacks in wireless sensor networks: A survey," *Journal of Network and Computer Applications*, vol. 35, no. 3, pp. 1022–1034, 2012.

[13] B. Parno, A. Perrig and V. Gligor, "Distributed detection of node replication attacks in sensor networks," in *IEEE Symposium on Security and Privacy (S&P'05)*, Oakland, CA, USA, pp. 49–63, 2005.

[14] T. P. Rani and C. Jayakumar, "Unique identity and localization based replica node detection in hierarchical wireless sensor networks," *Computers and Electrical Engineering*, vol. 64, pp. 148–162, 2017.

[15] Q. Yang, X. Zhu, H. Fu and X. Che, "Survey of security technologies on wireless sensor networks," *Journal of Sensors*, vol. 2015, article. 842392, pp. 1–9, 2015.

[16] A. K. Mishra and A. K. Turuk, "Residual energy-based replica detection scheme for mobile wireless sensor networks," *Security and Communication Networks*, vol. 38, pp. 637–648, 2015.

[17] Numan, M. *et al.* "A systematic review on clone node detection in static wireless sensor networks," *IEEE Access*, vol. 8, no. 2020, pp. 65450–65461. 2020.

[18] Y. Zeng, J. Cao, S. Zhang, S. Guo and L. Xie, "Random-walk based approach to detect clone attacks in wireless sensor networks," *IEEE Journal on Selected Areas in Communication*, vol. 28, no. 5, pp. 677–691, 2010.

[19] S. Lalar, S. Bhushan and Surender, "An efficient tree-based clone detection scheme in wireless sensor network," *Journal of Information and Optimization Sciences*, vol. 40, no. 5, pp. 1003–1023, 2019.

[20] C. Roy, K. Cordy and J. R. Koschke,"Comparison and evaluation of code clone detection techniques and tools: A qualitative approach,", *Science of Computer Programming*, vol. 74, no. 7, pp. 470–495, 2009.

[21] M. White, M. Tufano, C. Vendome and D. Poshyvanyk, "Deep learning code fragments for code clone detection,", in *Proc- 31st IEEE/ACM Int. Conf. on Automated Software Engineering (ASE)*, pp. 87–98, 2016.

[22] B. Hummel, E. Juergens, L. Heinemann and M. Conradt, "Index-based code clone detection: incremental, distributed, scalable", in *IEEE Int. Conf. on Software Maintenance*, pp. 1–9, 2010.

[23] I. D. Baxter, A. Yahin, L. Moura and M. Sant'Anna, "Clone detection using abstract syntax trees," in *Int. Conf. (Ed.). on Software Maintenance (Cat. No. 98CB36272)*, Bethesda, MD, USA, pp. 368–377, 1998.

[24] D. Rattan, R. Bhatia and M. Singh, "Software clone detection: A systematic review,", *Information and Software Technology*, vol. 55, no. 7, pp. 1165–99, 2013.

[25] L. Jiang, G. Misherghi, Z. Su and S. Glondu, "Deckard: scalable and accurate tree-based detection of code clones," in *29th Int. Conf. on Software Engineering (ICSE'07)*, pp. 96–105, 2007.

[26] J. Svajlenko and C. K. Roy, "Evaluating clone detection tools with big clone bench," in *IEEE Int. Conf. on Software Maintenance and Evolution (ICSME)*, pp. 131–140, 2015.

[27] C. M. Kamalpriya and P. Singh, "Enhancing program dependency graph based clone detection using approximate subgraph matching," in *IEEE 11th Int. Workshop on Software Clones (IWSC)*, pp. 1–7, 2017.

[28] K. Solanki and S. Kumari, "Comparative study of software clone detection techniques," in *Proc. Management and Innovation Technology Int. Conf. (MITicon)*, pp. MIT–152, 2016.

[29] A. Sheneamer and J. Kalita, "Semantic clone detection using machine learning," in *Proc. 15th IEEE Int. Conf. on Machine Learning and Applications (ICMLA)*, pp. 1024–1028, 2016.

[30] V. Bauer, T. Volke and S. Eder, "Combining clone detection and latent semantic indexing to detect re-implementations," in *IEEE 23rd Int. Conf. on Software Analysis, Evolution, and Reengineering (SANER)*, Osaka, Japan, pp. 23–29, 2016.

[31] M. Z. Chowdhury, M. Shahjalal, S. Ahmed and Y. M. Jang, "6G wireless communication systems: Applications, requirements, technologies, challenges, and research directions," in *IEEE Open Journal of the Communications Society*, vol. 1, pp. 957–975, 2020.

[32] N. Malhotra and M. Bala. "Fault diagnosis in wireless sensor networks-a survey." in *In 4th Int. Conf. on Computing Sciences (ICCS)*, Jalandhar, India, pp. 28–34, 2018.

[33] A. Elkhail, J.Svacina and T.Cerny, "Intelligent token-based code clone detection system for large scale source code," in *Conference on Research in Adaptive and Convergent Systems (RACS '19) Association for Computing Machinery*, pp. 256–260, 2019.