Journal of
Sensor and Actuator Networks

MDPI

Article

Efficient Intrusion Detection Algorithms for Smart Cities-Based Wireless Sensing Technologies

Rabie A. Ramadan [1,2]

[1] College of Computer Science and Engineering, University of Ha'il, Ha'il 81451, Saudi Arabia; rabie@rabieramadan.org
[2] Computer Engineering Department, College of Engineering, Cairo University, Giza 12613, Egypt

Received: 24 June 2020; Accepted: 14 August 2020; Published: 19 August 2020

Abstract: The world is experiencing the new development of smart cities. Smart cities' infrastructure in its core is based on wireless sensor networks (WSNs) and the internet of things (IoT). WSNs consist of tiny smart devices (Motes) that are restricted in terms of memory, storage, processing capabilities, and sensing and communication ranges. Those limitations pose many security issues where regular cryptography algorithms are not suitable to be used. Besides, such capabilities might be degraded in case cheap sensors are deployed with very large numbers in applications, such as smart cities. One of the major security issues in WSNs that affect the overall operation, up to network interruption, in smart cities is the sinkhole routing attack. The paper has three-fold contributions: (1) it utilizes the concept of clustering for energy saving in WSNs, (2) proposing two light and simple algorithms for intrusion detection and prevention in smart cities—threshold-based intrusion detection system (TBIDS) and multipath-based intrusion detection system (MBIDS), and (3) utilizing the cross-layer technique between the application layer and network layer for the purpose of intrusion detection. The proposed methods are evaluated against recent algorithms—S-LEACH, MS-LEACH, and ABC algorithms.

Keywords: smart cities; sensing technologies; intrusion detection; threshold-based algorithm; multipath-based algorithm; LEACH

1. Introduction

The world is now experiencing a huge change in technology, especially sensing technologies. These emerging technologies lead to new opportunities in industry and economy. At the same time, people are connected through their smartphones, laptops, and tablets. In addition, smart devices, meters, and appliances are used in almost every city. Vehicles and social systems, as well as services, are almost connected, forming an "internet of things (IoT)." In addition, researchers and organizations are developing standards and protocols for IoT systems to standardize devices' connections. Consequently, cities are developing their infrastructures, services, control systems, and monitoring systems to accommodate the new changes. Smart transportations, smart traffic, weather, and location services are also connected. Public safety and disaster responders are also integrated. Such connectivity and integrations form a new concept, which is smart cities. However, this uncontrolled growth of the cities brings new situations and issues that need to be taken into consideration by the governments and stockholders. The concept of a smart city depends exclusively on embedded systems, intelligent technology, and sensing technologies. In general, a smart city uses information technology and embedded infrastructure to improve the standard of living. Two of its main issues are security and electronic crime concerns [1,2].

Three factors affect the smart city securities, which are governess, technological, social factors. The focus of this paper is on the technological factors, which is mainly the sensing technologies.

Sensing technologies include different types of sensors and networks where smart sensors are used in many applications. Smart sensors are tiny sensors with limited processing and storage capabilities. They are also equipped with communication and sensing technology, see Figure 1; however, they also remain limited in their sensing and communication ranges because of their extremely limited energy sources. As can be seen in Figure 1, the main components of a sensor node are: (1) the power unit, which is the source of energy, (2) the sensing unit that is used to collect the data from the monitoring environment, (3) the processing unit in which the collected data is processed, (4) the radio unit where the communication takes place, (5) the localization unit that defines the location of the sensor nodes, (6) the mobilize unit is used to help the node move, (7) the actuator unit is used for moving sensors parts, (8) the multimedia unit is used for handling videos and images if needed, and (9) the power generator that is used for energy harvesting, if any.

There are many sources of energy that could be used to recharge the sensor nodes, such as WiFi charging units, and mobile vehicle with removable chargers, as well as the advances on solar cells, makes the self-recharging of sensor nodes possible [3,4]. However, each of those solutions come with a cost in which the size of the node increases as well as its price. Therefore, with the large number of sensor nodes deployed in an application, such as a smart city, sensors budget will not be affordable. One more thing to consider is that wireless sensor networks (WSNs) do not always operate in a safe environment; they might be deployed in a hostile environment that cannot be accessed.

Sensors are deployed manually or randomly in the monitored field to monitor the surroundings [5]; they autonomously form a connected network—wireless sensor network (WSN). They usually use self-discovery techniques [6–8] to discover their neighbors' nodes. Those sensors cooperate to send their sensed information to a powerful node, named sink node. There are various routing and clustering algorithms that are used for such purposes [9,10]. WSNs are considered as the core network of smart city applications. Once deployed, WSN is supposed to live for a long time without recharging the nodes' batteries.

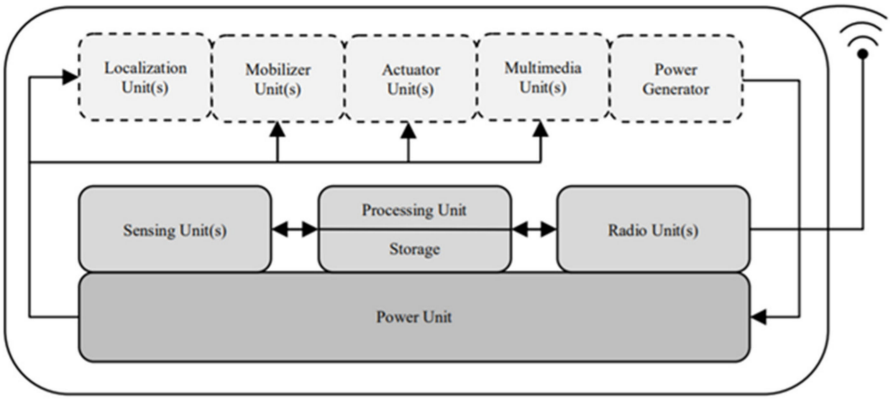


Figure 1. Sensor node architecture [11].

Due to a large number of sensor nodes deployed in the monitored field, especially in smart cities applications, sensors are classified into clusters to reduce the network traffic; consequently, it saves the energy of the sensors. Each cluster head (CH) is responsible for data collection and aggregation and then sends the data to the sink node. Sink node is responsible for data analysis and decision making. It is also possible that the sink node could be as a gateway to another network level where data is collected from different sink nodes to be analyzed.

Security is a challenge in WSNs where nodes could be easily compromised. There are different types of attacks that might affect the integrity, authenticity, and availability of sensor-sensed information, including, denial of service (DoS), jamming, eavesdropping, back-ole, sybil, clone, selective forwarding, black hole, and wormhole attacks [12,13]. The authentication process is also a challenging task due to the dynamicity of the WSNs and their self-configuration process. There are also some routing,

data eavesdropping, and man-in-middle attacks in WSNs, where the attacker uses the transmission medium to capture network traffic and hijack network security. Similarly, a jamming attack is also a type of transmission attack where the attacker uses a large number of transmission signals of the same frequency to disrupt network operations. Jamming attacks could result from other unintentionally wireless interferences, such as noise, crash, etc. [14]. DoS attacks in a wireless network, on the other hand, are designed to transmit an unintended transmitting signal to the sensor node in order to interrupt the communication channel, bandwidth, battery life, storage, and destination route. It disturbs not only the networking cycle, but it also minimizes network lifespan [15].

One of the famous attacks in WSNs is the sinkhole attack, where an intruder compromises a node and lunches an attack [16]. The intruder tries to attract all of the traffic from its neighbor nodes according to the used routing metric. Different solutions are introduced in the literature to solve the sinkhole attacks problem [17]. Those solutions are classified in Figure 2 as follows. The rule-based solutions are based on forcing some rules to detect the sinkhole attacks, while anomaly-bead detection methods depend mainly on monitoring the network traffic. The key management techniques use the different cryptography algorithms to detect and mitigate the sinkhole attacks, while the non-crypto-based approaches try to avoid using any encryption/decryption methods due to their computation overhead. Finally, the hybrid approaches are a combination of one or more of those approaches.

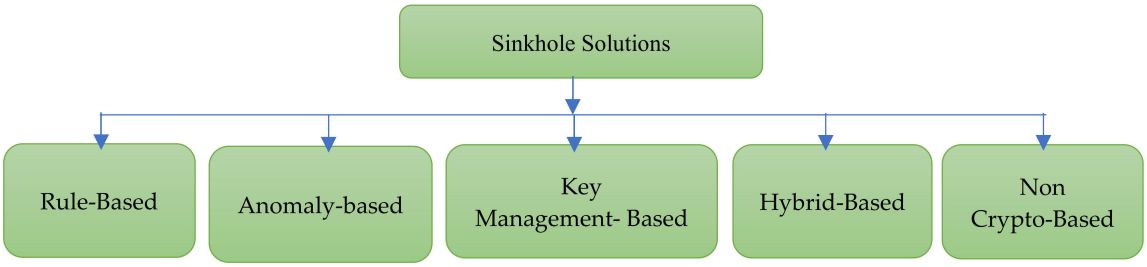


Figure 2. Sinkhole attack solutions classifications [18–28].

This paper introduces two solutions to the sinkhole problem in smart cities based on intrusion detection systems (IDS), namely, multipath-based IDS (MBIDS) and threshold-based IDS (TBIDS). Those two algorithms are designed to be simple enough for limited resources devices, such as sensors. In addition, they put the heavy load on the sink, or sometimes it is called a base station where the energy source is not a problem. For comparison reasons, S-LEACH [22], MS-LEACH [23], and ABC algorithms [28] are implemented and evaluated against the proposed algorithms. The proposed algorithms consider ad-hoc on-demand vector (AODV) [29] routing protocol as the base for communication between the nodes and the CHs, as well as between the CHs and the sink node. AODV has been selected due to its simplicity and because it is a well-known routing protocol in WSNs. AODV modified version [30] proposed for WSNs is implemented in this paper due to its simplicity and its lower routing load. However, the proposed algorithms are applicable to be used with any routing protocol.

The Motivation: In AODV, when one of the nodes needs to send a message, it creates an RREQ (route message) packet to its neighbors. In this case, one of its neighbors close to the destination will reply by RREP (route reply) packet. The source receives a number of RREP packets from its neighbors and selects one node with the least number of hops to send the message. In a sinkhole scenario, the compromised node sends RREP with the minimum number of hops to the sink node. This scenario is repeated with every node that sends the RREQ message. In this case, the compromised node blocks all of the traffic to the sink node. It might perform a denial of service (DoS) attack or reply attack, changing the content of received messages and sending them back to the sink node. As can be seen, such a scenario either interrupts the operation of the smart city networks or provides false information to the decision-makers. Therefore, there is a need for a simple and efficient solution to the sinkhole

problem. However, by studying the previously mentioned solutions, the following common issues have been found, and the proposed algorithms in this paper try to solve some of them:

- Memory network overhead,
- Network communication cost is high,
- The message loss due to sinkhole attack is not considered,
- Collision may impact the proposed solutions,
- Dummy messages are used to detect intrusion.

This paper is motivated to find elegant solutions to the sinkhole problem. The focus of this paper is on the sinkhole attack and its effect on network performance. For instance, collision and dummy messages are not considered in this paper. However, it tries to reduce the communication cost, giving that it is a tradeoff between the network communication and security. Besides, the proposed algorithms try to free the sensor nodes from excessive computation and offload most of the overall computation to the sink node.

The paper is organized as follows: The next section presents the most related work to the research proposed in this paper. The system design is described in Section 3. The proposed intrusion detection techniques are described in Section 4. Section 5 presents the conducted experiments and analysis, while Section 6 contains the discussion. The paper has the conclusion and future work in Section 7.

2. Related Work

Sensor networks are one of the bases either of smart cities or IoT networks. Therefore, many of the algorithms and techniques are proposed for better operation of the network, including clustering algorithms. The most related clustering algorithm for our work in this paper is LEACH [31,32]. At the same time, there are many of the routing algorithms that are proposed for WSNs. However, one of the most used routing protocols is AODV. Based on our research, the LEACH algorithm makes the WSNs vulnerable to the different types of attacks, including the sinkhole. Therefore, this section describes LEACH as a clustering algorithm, S-LEACH, MS-LEACH, and ABC as sinkhole detection algorithms, and AODV as routing technique.

LEACH Algorithm [31,32]: It is a clustering algorithm designed especially for WSNs that make the routing process. Its main idea is based on the concept of energy consumption of each node. It is also a round-based technique, where cluster heads (CHs) are periodically changing to balance the network energy. Previously selected CHs cannot join the next round of selection. The algorithm divides the nodes into groups with a cluster head for each group. It works in two phases—setup and creation. In the setup phase, each node generates a number between 0 and 1. If a node selects a number less than a certain threshold (*T(i)*), it announces itself as a cluster head for the current round. *T(i)* is computed as follows:

$$T(i) = \begin{cases} \left(\frac{p}{(1-p)\left(r \bmod \frac{1}{p}\right)}\right) if\ i \in G \\ 0\ otherwise \end{cases} \quad (1)$$

where p is the desired percentage to be a cluster head, G is the node that is not selected in the $1/p$ round, and r is the sensor communication range. The nodes' desires could be generated randomly or based on the node's residual energy.

The second phase is called steady-state, where nodes join the nearest cluster heads and start aggregating and transmitting data to their CHs. The CHs are responsible for transmitting their collected data to the sink node. After a certain period of time, the network starts another round of selection to CHs. There are many variations of LEACH, including the ones reported in [33].

S-LEACH Algorithm [23]: Since LEACH in its current format is vulnerable to different attacks. S-LEACH adds security to protect the network from insiders and outsiders. S-LEACH protects the network from outsider attackers, guaranteeing data authentication and data freshness. It borrows some of the building blocks from the SPIN [34] algorithm, where symmetric keys are used for authentication.

In the pre-deployment phase, each node is equipped with a symmetric key that is shared with the sink node. Therefore, each node authenticates itself through the "Adv" message with the message authentication code (MAC) generated using the shared key. The same process occurs to authenticate the cluster heads.

LEACH Sinkhole [35]: In the original form of LEACH, the selection of the CH depends on the signal strength; others consider the distance from the cluster heads and the CH energy. Therefore, a powerful adversary can send a message to all of the nodes on the network, announcing itself as a CH, see Figure 3. The adversary receives all of the nodes' messages and drops them all, interrupting the network's operation.

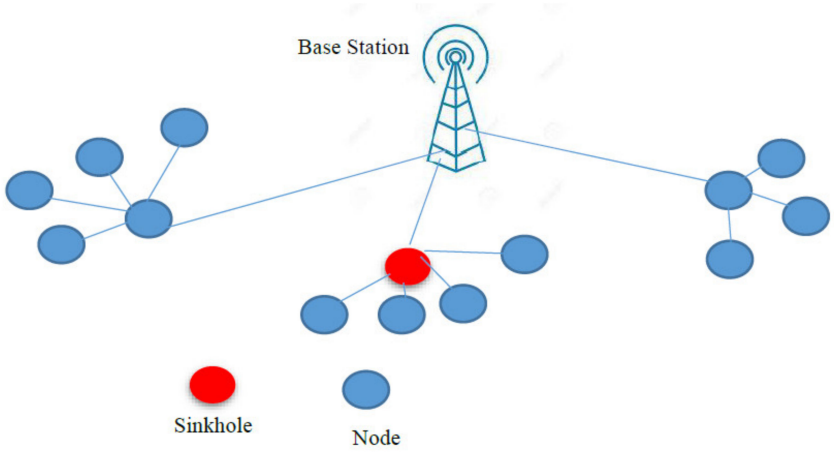


Figure 3. Sinkhole attack [35].

MS-LEACH Algorithm [22]: It tries to enhance the security of S-LEACH, adding data confidentiality, and improving nodes to cluster head authentication. It adds another security level where a shared pairwise key between the CH and its member is established. It employs Blowfish as its cryptographic function. Therefore, MS_LEACH assumes that each node is equipped with two symmetric keys shared with the sink node, and the sink node holds the last keychain. As can be seen in the S-LEACH and MS-LEACH, they are secure in terms of authentication and confidentiality but still vulnerable to sinkhole attacks. The following is the sinkhole attack on LEACH.

ABC Algorithm [28]: The idea of ABC is based on the Bees algorithm, where Bees work in a colonial manner to get better food sources using what is called the waggle dance. This waggle dance intimates the communication between nodes. Each bee is assigned energy value that could be positive or negative, indicating the amount of search it did. This value could identify the sinkhole node based on a rule set table. The algorithm assumes that some of the nodes can be selected to examine a suspicious sinkhole node through monitoring its traffic and route update messages. The algorithm seems useful in discovering the sinkhole node. Simultaneously, the bees' algorithm takes too much time to stabilize and consume much energy from the sensor network.

Other algorithms are proposed in the literature, such as the one proposed in [36]. The authors have proposed a mechanism to detect the sinkhole attack by examining the high traffic area. The authors have divided the monitored area into regions, and they monitor the high traffic in each region. However, such a mechanism leads to a high false-positive rate. A similar approach is proposed in [37], where it tries to detect the misuse of the nodes. However, the proposed mechanism has not shown many improvements. A mobile agent-based approach is also proposed in [38] to mitigate the sinkhole in WSNs. One of the disadvantages of such a mechanism is the large overhead of the mobile agents that have to run on the sensor nodes. A trust-aware routing framework (TARF) is also introduced in [39], where dynamic WSNs are considered. The framework tries to avoid sensor time synchronization and geographic location. The authors claim that their framework is capable of saving a reasonable amount of energy-based routing. Besides, the authors of [40] have proposed a sinkhole mitigation system for

the internet of things (IoT) communication. The system tries to detect and avoid bypass the area that has a sinkhole node. However, the proposed system's computation is very high.

3. System Design

In this section, the details of the proposed algorithms for sinkhole detection and mitigation in smart cities are explained. The algorithms assume the integration between the application layer and the routing layer of the WSNs. The routing layer provides the routing information to the application layer for intrusion monitoring and detection. Here, two different approaches are proposed for sinkhole attack detection. However, the same algorithms could be used for monitoring other attacks as well.

Figure 4 shows a sample WSN with 70 nodes and one sink node used for smart cities. The sink node is marked with a red circle on the figure. Certainly, the network could have multiple sink nodes due to the large monitoring field. Other possible scenarios in smart cities are to have more than one WSN with more sink nodes. Our proposed approach is feasible for all of the scenarios.

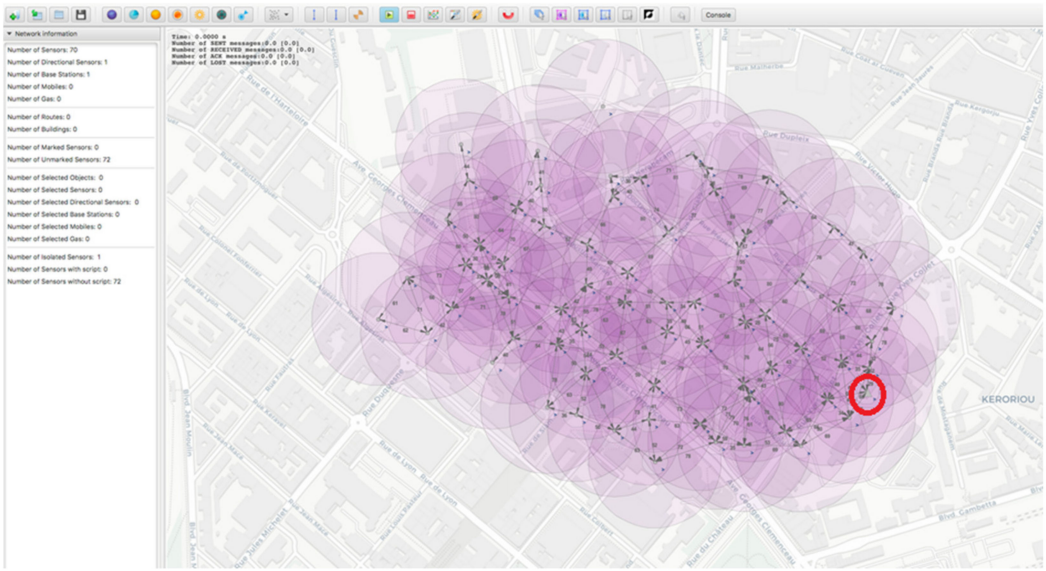


Figure 4. Sample wireless sensor network (WSN) for smart cities (generated by [41]).

If the sensing device is mandated to send its sensed data, it must send a request for connection establishment to its CH first. If there is a route in the routing table of the CH, it puts it on the link to be captured by the next CH device. The CH will otherwise discover an appropriate route. A CH device transmits a route request message (RRM) to other connected CHs to offer the next step route. If the adjacent CH has the route information to the sink node, it returns the route information message (RIM) to the source CH. If not, it will give RRM to the linked CHs.

Figure 5 shows a typical message routing scenario in which a network of sensors and CHs are deployed in the monitored field and a sink node. If the sensor number 1 wants to send its sensed data, it notifies the CH number 6. CH number 6 receives the message and tries to send it to the sink node. If it has the routing table, it uses it to send the message directly; if not, it sends RREQ to all of its neighbors. One of the neighbors will reply to it, assuming the reply comes based on the CH proximity. Therefore, CH number 5 will reply to CH number 6 with route information message, RREP. After identifying the routes for the sink node to get the routes, it broadcasts a message to all of its neighboring CHs. CHs will continue broadcasting the message to their neighbors by increasing the hop count until it reaches all CHs. The routing information will go back to the base station through the same routes.

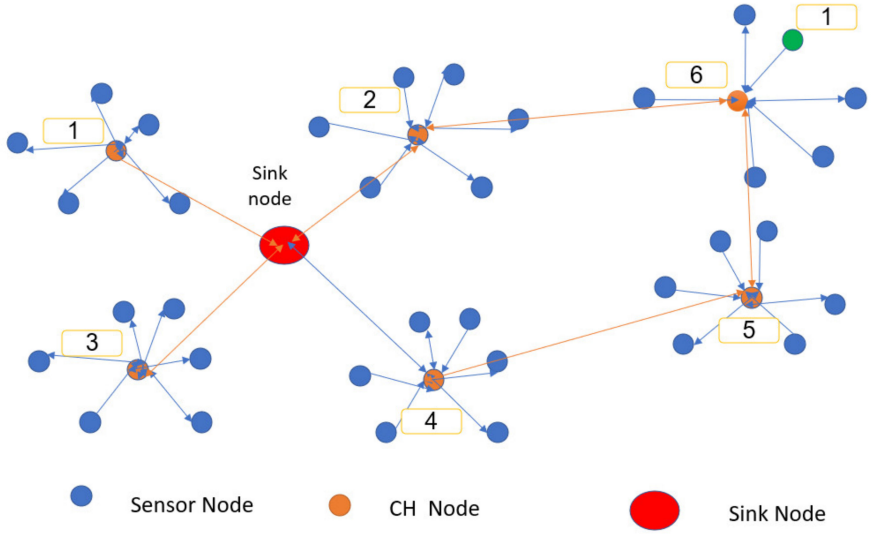


Figure 5. Typical routing scenario.

Launching Sinkhole Attack

One of the attack problems in smart cities is the sinkhole, where intruders try to forward the traffic to a particular compromised node, in our case, a sink node. The attack could be due to a single compromised node or could be due to multiple compromised nodes. In AODV, the advisory will send RREQ to get the neighboring CHs, asking for a route to the base station. The neighboring CHs reply with RREP information. The compromised CH sends RREP with the routes to the sink node. At the same time, it sends to its neighbors that it has the optimum route to the sink node, in terms of either energy or distance. Now, all of the intruder's neighbors forward their data to the compromised CH.

4. Proposed Intrusion Detection Algorithms for Smart Cities Network

In this section, two different algorithms for intrusion detection are proposed. The first one is based on the multipath routing to a control message, while the second one is a threshold base. Both of them utilize the cooperation of the CHs. In addition, both algorithms work on different types of WSNs, including single sink and multi-sink networks (base stations). The proposed IDSs work fine with multi-sink nodes, where the IDS could be deployed on all of the sink nodes.

4.1. Multipath-Based IDS (MBIDS)

To solve the sinkhole problem, an MBIDS algorithm is assumed to be uploaded to all of the CHs. However, before describing the details of the algorithm, a modified version of the AODV protocol is introduced to guarantee disjoint routes as follows:

To guarantee disjoint routes for the control messages, AODV is modified as follows:

Once the RREQ is broadcasted by the source, each intermediate node receiving the request attaches its address to the request and broadcasts it again to its neighbors. This process is repeated until the request reaches the sink node. The sink node forms disjoint paths as a reply to the source node request through the selected paths to send the data through them.

The proposed attack identification and prevention system, MBIDS, is described as follows:

Phase 1: The network is divided into clusters, where each cluster has a CH.

Phase 2: Once the data Msg is ready, an RREQ is instantiated by the CH as a source node. The reply Msg includes two different paths to the sink node, as stated previously, in the modified version of AODV protocol. The CH sends the data Msg through one of the received routes and generates Ctrl Msg through the other route. The Ctrl message consists of sender ID, size, control route information, packet information route, and checksum.

Phase 3: Once the sink node receives the data message and both control messages, it starts to compare the control message with the received message. By analyzing the control messages, the sink node will realize if the received message is changed or it is received without change. In addition, if the sinkhole drops the received messages, the sink node would realize that by receiving the control messages.

Phase 4: Since the sink node records all of the routes, it analyzes the history of the same route to discover which CH is attacking the traffic.

Phase 5: If the sink node detects an intrusion, it broadcasts an alert to all of the CHs to avoid the compromised one.

As shown in Algorithm 1, it is assumed to have S number of nodes, a set of cluster heads H, and one sink node or more, SN. When a sensor s_i is ready to send its sensed data, it sends the RREQ message to the CH_j connected to it. CH_j broadcasts a message to its neighbors following the procedure of phase 2, stated previously, $Neigh\left(CH_j\right)$. Once it receives the two disjoint paths, it replies by RREP message to s_i with permission to send. After receiving the message, CH_j sends the received message (data msg) through one of the routes it received—Send (Msg)—and forms a Ctrl message to be sent through the other route Send (Ctrl). At the sink node, both data and the control messages are received from different routes. The sink compares the received data with the information contained in the Ctrl message. Basically, it compares the sender ID, Msg size, packet information route, and checksum. The packet information route is checked once more to guarantee the route is not tampered with, and the routes are disjoint routes. If the data do not match the information in the Ctrl message, then the route and its history are investigated. If a certain number of messages are directed to the same CH, such CH is considered as sinkhole node. Based on our experiments, 15 rounds are enough to detect the sinkhole.

Algorithm 1. Multipath-Based IDS (MBIDS)

$s_1 \ldots s_n \in S$
$CH_1 \ldots . CH_h \in H$
$sn_1 \ldots . sn_n \epsilon SN$
Repeat
  If (s_i ready to send)
    $s_i \overset{\text{RREQ}}{\rightarrow} CH_j$
    $CH_j \overset{\text{RREQ}}{\rightarrow} Neigh\left(CH_j\right)$
    $CH_j \overset{RREP}{\leftarrow} Neigh\left(CH_j\right)$
    $s_i \overset{\text{Msg(Data)}}{\rightarrow} CH_j$
    Send (Msg)
    Send (Ctrl)
  End if
  $\forall\ \boldsymbol{sn_i} \in \boldsymbol{SN}$
    Receive (Msg(data))
    Receive (Msg (Ctrl))
    If (*Compare* (Msg (data), Msg (Ctrl)) = true)
      Alert()
    End if
  Until the transmission process is completed
End

As can be seen in Algorithm 1, MBIDS involves certain overhead, which is the control message that has to be sent along with the message and through another route. The control message, in our experiments that will be shown in later sections, contains a small number of fields, which are sender

ID, size, control route information, packet information route, and checksum. The control message has no payload information, and it is also a default message with any data message to be sent from any of the sensor nodes; therefore, the overhead is the only control message that will be sent through another route. To easily recognize the overhead of the MBIDS algorithm, the payload of any WSN could take the values of 120, 60, 40, 30, 24, and 20 bytes, and the control message is only 8 bytes [3]. Therefore, the overhead varies with the payload size of the message to be sent to the sink node; while in the worst case, the overhead is 40% when the payload is 20 bytes only, in the best case, the overhead is almost 6% when 120 bytes are used as message payload. In our case, the best case has been utilized where the 20 bytes payloads are not practically used in real network scenarios. However, security always comes with a cost, and it is a tradeoff between network overhead and network disruption. There are other overheads that have been noticed during simulating the algorithms, which are the packet loss and network energy consumption. These overheads have been described later in the experiment section.

4.2. Threshold-Based IDS (TBIDS)

Here is another method in detecting the sinkhole intrusion attack. The idea behind the TBIDS is its simplicity, and it works only on the sink node, which saves sensors' energy, and it does not affect the network packet loss. In this algorithm, the sink node plays the major rule in detecting the attack. It is assumed that the sink node is aware of all of the messages' routes. Therefore, it can determine the path of any of the messages and recognize the cluster heads that forwarded them. At the same time, the sink node is aware of the cluster head nodes' positions. Thus, the sink node would exactly know the number of messages transmitted to certain *CH*, as well as the number of messages transmitted to the sink node from that particular *CH*. A sinkhole node attracts other *CHs* to forward their messages to it. It either drops the received messages or sends them to the nearest *CH* or to the sink node, based on the sinkhole node position in the network. In both cases, the sink node is now aware of the situation and can determine if there is a sinkhole or not. The sink node sets threshold values for each *CH*, N_i, α_1, and α_2, based on the number of received and transmitted messages compared to other CHs. Those values need to be learned by the sink node; however, for the purpose of the performance measure to the TBIDS, those values are manually set based on trial and error experiments, where the learning process is out of the scope of this paper, and it is part of my future work.

The sink node computes certain ratios for each *CH* named sinkhole ratios, $R\,{}^{t}_{j}$*and* $SR\,{}^{b}_{j}$, in which $SR\,{}^{t}_{j}$ is the transmission sinkhole ratio, and $SR\,{}^{b}_{j}$ is the blocked sinkhole ratio for CH_j. Those ratios are compared to α_1 and α_2, respectively, to determine if the *CH* is a sinkhole or not. The detailed algorithm is shown in Algorithm 2. As can be seen, the sink node computes $SR\,{}^{t}_{j}$ *and* $SR\,{}^{b}_{j}$ for each cluster head CH_j and compares them to the thresholds α_1 and α_2, respectively, to determine if CH_j is a sinkhole node or not. If it is a sinkhole, it will be isolated, and CHs, as well as nodes, will be notified; otherwise, no action is determined.

Algorithm 2. Threshold-Based IDS (TBIDS)

$s_1 \ldots s_n \in S$
$CH_1 \ldots . CH_h \in H$
$sn_1 \ldots . . sn_n \epsilon SN$
$SR\,{}^{t}_{j}$ is the Sinkhole transmission ratio for CH_j
$SR\,{}^{b}_{j}$ is the Sinkhole transmission ratio for CH_j
RM_j Received messages at CH_j
TM_j Transmitted messages at CH_j
Repeat
    $\forall\ \boldsymbol{sn_i} \in \boldsymbol{SN}$

Algorithm 2. Threshold-Based IDS (TBIDS)

If (Msg is received)
$\forall$ $CH_j \in H$
$SR_j^t = \frac{|RMj|}{|TMj|}$
SR_j^b= CH_j blocked messages
If (($SR_j^t \geq \propto_1 \sim \infty$) or ($SR_j^b \leq \propto_2$))
CH_j is Sinkhole node
Alert (CH_j)
else
CH_j is not a Sinkhole node
End if
Until the transmission process is completed
End

The sink node does not base its computation on the *CH* self-report, but it is based on its neighbors' report, as well as on the received messages routing information. Therefore, a *CH* cannot deceive the sink node.

TBIDS Detection Model

In this section, the detection model is described in which two cases are considered, a single sink node station and multiple sink nodes. A sensor network consists of *S* sensing devices that could be homogenous or heterogeneous. The heterogeneous sensor, in this context, is a sensor with different settings or capabilities. For instance, sensors might differ in their sensing, communication, and/or initial energy. In the setup phase, AODV and LEACH models are uploaded to all of the sensor nodes. Sensors are deployed either randomly or according to one of the deployment algorithms [42]. After deployment, the LEACH clustering algorithm is applied to form connected cluster heads (CHs). Once the cluster heads are chosen, the network is now ready to operate.

In smart cities, the concept of using a single sink node might not be applicable. Therefore, in this paper, the detection model for two cases—a single sink node and multiple sink nodes—are considered.

In the case of a single sink node or multi-sink nodes, the sink is responsible for collecting sensors information as well as running the detection model. Besides, the sink node *SN* is supposed to compute the following information for each cluster head (CH_j):

- The packets sent (RMj)
- The packets received (TMj)
- The sinkhole ratios SR_j^t *and* SR_j^b

$$SR_j^t = \frac{|RMj|}{|TMj|} \text{ and } SR_j^b = CH_j \tag{2}$$

Based on the value of ratios SR_j^t *and* SR_j^b, the sink node (*SN*) decides on the malicious CH_j. The value of SR_j^t is a numeric value in which:

$$SR_j^t = \begin{cases} \infty, |CH_j \text{is a malicious node} \\ n, |n \leq N_i \ CH_j \text{ is not a malicious node} \end{cases} \tag{3}$$

N_i is learned from the network activities or decided by the network designer. Therefore, the best value of N_i has been exerted based on the trial and error approach. It would be better to have a learning

algorithm to deduce such value and other thresholds as well. However, it is considered out of the scope of this paper, and it is part of the future work.

5. Simulation and Analysis

5.1. Simulation Environment

To simulate the previously mentioned algorithms and sensor networks, the CupCarbon simulator [34] is extended. CupCarbon provides a good library for analyzing the sensor's energy and providing sensor network performance evaluation. It also provides different types of sensors that can be programmed using simple script language. The simulator has been extended by adding the LEACH clustering, AODV routing protocol, ABC, S-LEACH, MS_LEACH, TBIDS, and MBIDS algorithms. The general network configuration is shown in Table 1.

Table 1. General sensor network configuration.

Sensors	Variable (Based on the Experiment Requirements)
Communication range	100 m
Transmission power	100 mW
Base station	Variable (based on the experiment requirements)
Deployment	Random

It is assumed that the compromised node is the one that has higher energy among the CHs. Besides, in this paper, the deployment process is done randomly; certainly, any type of deployment could be suitable to be used as well. Sensors select their CHs based on the LEACH algorithm. Additionally, *CHs*, as well as the sink node(s), are designed to use one of the IDS monitoring techniques (multi-path or threshold-based).

The evaluation criteria are as follows:

- Detection Accuracy: Based on the used algorithms, the detection accuracy is measured. This criterion has not been used as a measure in the previous algorithms, but it is believed that it will be beneficial to check on the detection accuracy in identifying the hacked node.
- Power Consumption: This is the average sum of the energy consumed by all nodes in transmitting messages at a certain time of operation. This could be measured by the following equation:

$$Avg\ power\ Consumption = \sum_{i=1}^{N} \left(\frac{sn_i\ consumed\ enegry}{N}\right) \quad (4)$$

- Network Lifetime: It is the time from the start of the simulation to the first node dies due to energy depletion.
- Packet Loss: The definition of the packet loss is a little bit different, where it represents the number of lost packets during the existence of the sinkhole attack.

Experiments are designed to include different numbers of sensors, different number of sink nodes, and various locations of sinkhole nodes. In addition, our proposed methods are compared to three different algorithms—S-LEACH, MS-LEACH, and ABC algorithms—explained previously.

5.2. Simulation Results

Experiments in this section are designed to show the performance of the proposed algorithms compared to other similar algorithms. S-LEACH, MS-LEACH, and ABC algorithms are implemented separately, considering the requirements and parameters of each of them. In addition, sensor deployment and node configuration are considered the same for all for a fair comparison. The experiments in this section are divided into different sets based on the performance measures.

Experiments are conducted with a different number of nodes, a different number of sink nodes, and different sinkholes. Small networks are used to examine the proposed IDS methods, and large-scale networks are used to examine the proposed IDS in smart cities with a large number of sensors.

For a small-scale set of experiments, the average detection accuracy is examined for the two proposed algorithms and compared with the S-LEACH, MS-LEACH, and ABC algorithms. Those sets of experiments consider small-scale networks with a range of sensor nodes between 25 and 500 nodes. The sink node is set at different positions in the network for the accuracy of the results. The monitored field or the deployment area is changed with the number of nodes deployed from 50 m × 50 m to 1000 m × 1000 m. Besides, the results are reported after the first node died in all of the experiments.

For large-scale networks, the network configuration is changed according to the performance criteria, and it has been described accordingly.

5.2.1. Detection Accuracy

For a single sink node and a single sinkhole, Figure 6 shows that the detection accuracy of ABC, MBIDS, and TBIDS are comparable. However, the detection accuracy of S-LEACH and MS-LEACH is zero percent. The reason behind that is S-LEACH and MS_LEACH depend on a secret key shared between the *CH* and the sink node; once the key is compromised, the sinkhole cannot be detected. So, as long as the *CH* is not compromised, S-LEACH and MS-LEACH have a higher percentage not to practice the sinkhole attack. On the other hand, ABC, MBIDS, and TBIDS have higher detection percentage. On average, ABC and MBIDS have a very close percentage of detection accuracy with a small number of nodes. When the number of nodes increases, MBIDS shows better accuracy, up to 95%. At the same time, TBIDS shows excellent detection performance, up to 99%, especially with a small number of nodes.

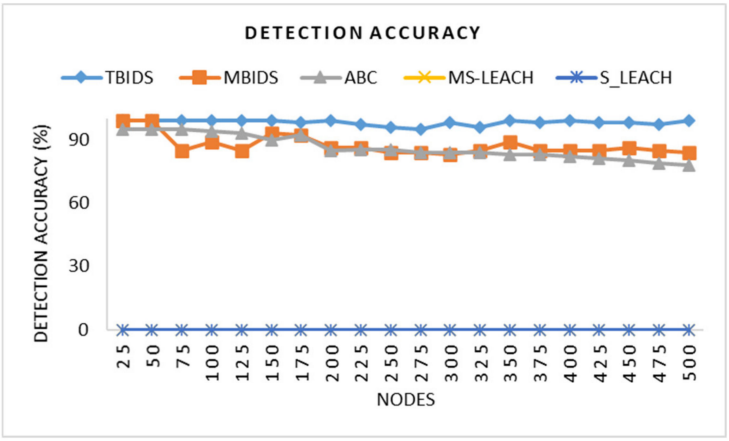


Figure 6. Single sink node and a single sinkhole.

For a single sink node and two sinkholes, again, Figure 7 shows the detection accuracy for the five algorithms. Since there are two sinkholes, the detection accuracy, on average, of TBIDS overperforms the ABC and MBIDS by 3% in the case of a small number of nodes and overperforms MBIDS by 5% and ABC by 7% in case of a large number of nodes are used.

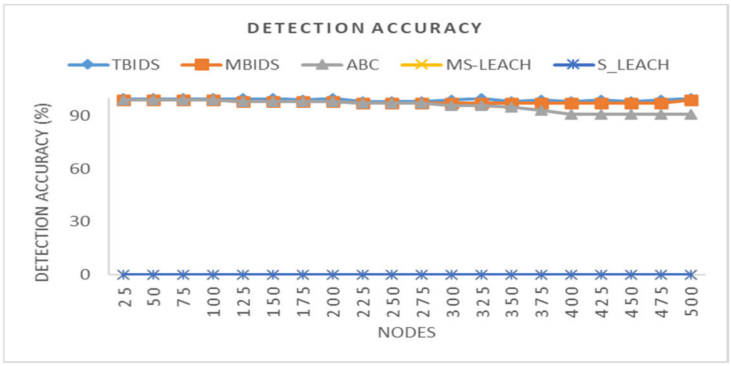


Figure 7. Single sink node and two sinkholes.

Figure 8 shows the detection accuracy for two sink nodes and two sinkholes. As can be seen in the figure, the detection accuracy of the proposed algorithms is slightly increased, almost by 1%. However, the detection percentage of ABC is still the same, and it is getting worse when the number of nodes increases. The difference between ABC and MBIDS and TBIDS is 7% and 9%, respectively. By investigating the experiments, it has been found that ABC is not designed to handle a network with multi-sink nodes. At the same time, ABC needs too much time to stabilize.

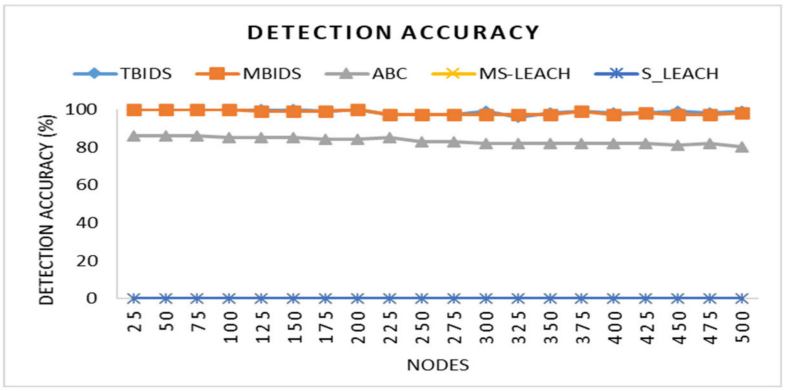


Figure 8. Two sink nodes and two sinkholes.

For large-scale networks, a different set of experiments with a range of nodes from 500 to 15,000 nodes is designed. In addition, MBIDS, TBIDS, and ABC algorithms with a single sinkhole are examined. As shown in Figure 9, the detection performance of ABC and MBIDS decreases with the increasing number of nodes. However, MBIDS still overperforms the ABC algorithm with, on average, 12%. At the same time, TBIDS performs almost the same with the increasing number of nodes with a small decreasing performance from the previous experiments. However, its accuracy, on average, is still 99%.

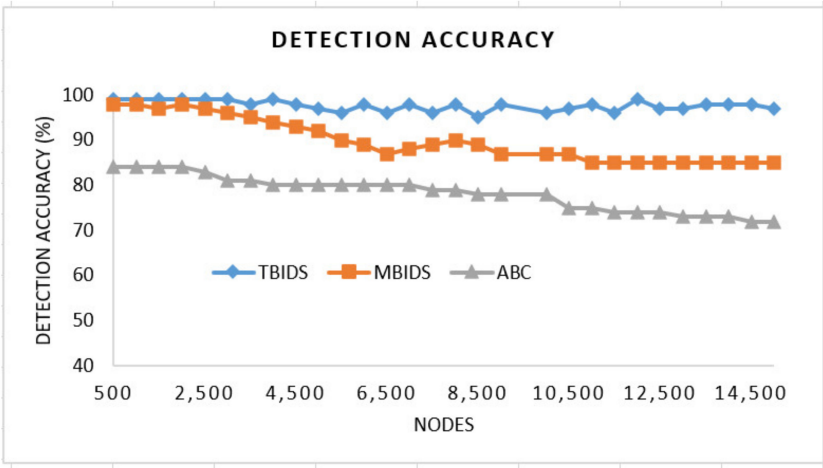


Figure 9. Detection accuracy for the large-scale network.

Figure 10 shows the detection performance with different sink nodes. In these experiments, the number of sink nodes changes as one, two, three, and five, while the number of nodes, 15,000 nodes, stays fixed, and the number of sinkholes is set to five. As can be seen, the detection accuracy of TBIDS is almost 100% in the three cases, where MBIDS is decreased to almost 84%. On the other hand, ABC is slightly reduced to 74% when three sink nodes are used.

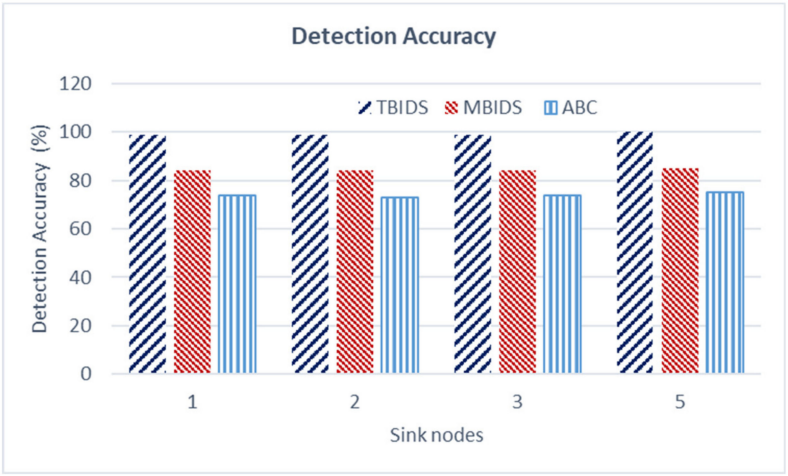


Figure 10. Detection accuracy for a large-scale network with 2, 3, and 5 sink nodes.

Due to a large number of nodes, experimenting with a large number of sink nodes is not possible due to the limitations of the running environment. Therefore, different sets of experiments are conducted with 8000 sensor nodes, 5 sinkholes, and variable sink nodes (10, 20, and 30). Figure 11 shows the detection accuracy for the three algorithms. It has been noticed that the performance of the three algorithms increases with the increase in the number of sink nodes. Besides, it is worth mentioning that the detection performance of MBIDS and ABC algorithms has been increased to reach, on average, 91% and 72%, respectively. So, it confirms that handling a fewer number of nodes by the sink node increases the detection accuracy. However, setting the required thresholds becomes harder, and the detection performance becomes more sensitive to small changes in those threshold values.

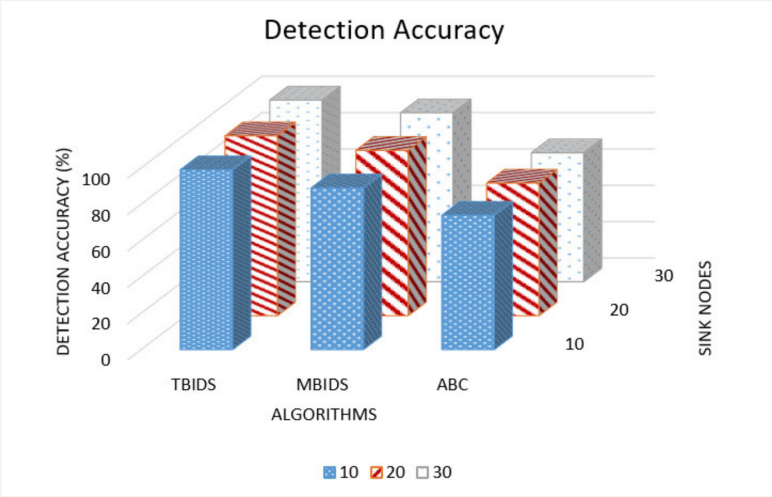


Figure 11. Detection accuracy for the large-scale network (8000 nodes) with 10, 20, and 30 sink nodes.

5.2.2. Average Power Consumption

Another set of experiments is conducted to examine the average power consumption for all algorithms. Again, the same configuration used in the previous experiments is followed. Small-scale and large-scale networks are examined.

For a small-scale set of experiments, it is considered more realistic to experiment with different sinkholes. This shows how the network is affected by the increasing number of sinkholes. The experiments are conducted for one, two, three, and five sinkholes and five sink nodes. Figure 12 shows the energy consumption results in all cases with a fixed number of packets sent through the network, 80,000 messages. Five hundred nodes randomly generate messages. The consumed energy is computed as a percentage out of the energy sum of the sensor nodes' initial energy. As can be seen in the figure, TBIDS gives the best energy consumption by almost 18% in all cases, while ABC and MBIDS consume, on average, almost 80%. However, MBIDS still overperforms ABC by 6% of the energy consumption; this percentage is increased to 9% when five sinkholes are considered. Regarding S-LEACH and MS-LEACH, as shown, they consume too much energy due to the cryptography algorithm that is used to authenticate the cluster heads.

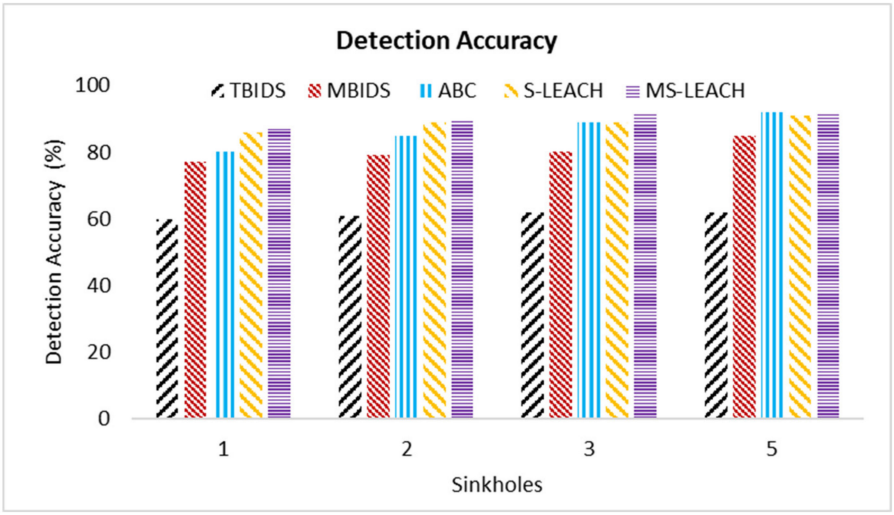


Figure 12. Energy consumption for a small-scale network with 1, 2, 3, and 5 sinkholes.

For a large-scale set of experiments, again, this section shows experiments with 15,000 nodes and five sink nodes. The number of messages to be sent by each sensor is assumed to be 100 messages. The results in Figure 13 show, on average, that TBIDS is the best among the five algorithms, while the worst is MS-LEACH. Besides, it turns out that ABC is much better than S-LEACH and MS-LEACH. However, MBIDS is still ahead of the ABC in terms of energy consumption by a slight difference. It is worth mentioning that after a deep investigation into the MBIDS, the overhead messages consume, on average, 5% of the overall network energy.

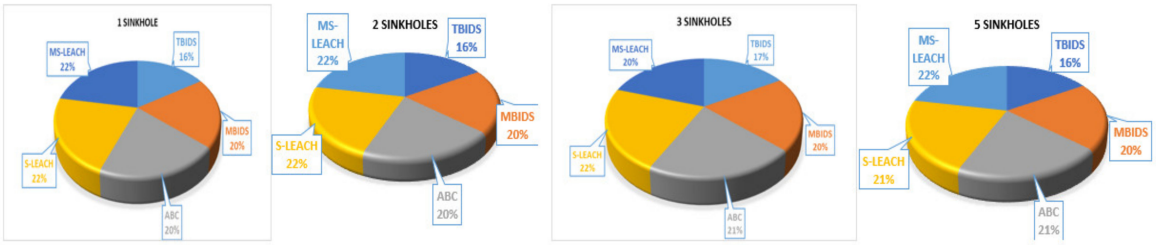


Figure 13. Energy consumption for a large-scale network with 1, 2, 3, and 5 sinkholes.

To examine the energy consumption of the overall network with a large number of sink nodes, the same set of experiments introduced in the previous section with 8000 sensor nodes, five sinkholes, and variable sink nodes (10, 20, and 30) is conducted. The results of this set of experiments are shown in Figure 14. In this figure, the energy consumption percentage is displayed to clarify the difference between the algorithms. Again, energy consumption has been significantly reduced with the increasing number of sink nodes. One clear note is that TBIDS energy consumption is reduced from 65% to 58%, and MBIDS is reduced from 72% to 63%. At the same time, S-LEACH is also showing similar energy consumption degradation percentage with almost 5%, on average. Moreover, although MS-LEACH is considered the worst to consume energy, it has also been enhanced, on average, 4%, when the number of sink nodes increases to 30 nodes.

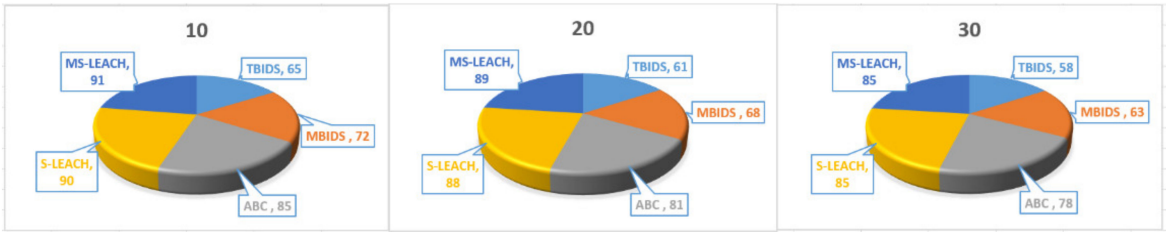


Figure 14. Energy consumption for a large-scale network, 8000 nodes, with 10, 20, 30 sink nodes, and 5 sinkholes.

5.2.3. Network Lifetime

In this section, another set of experiments conducted to measure the lifetime of the network under the operation of the five algorithms is presented. Once more, two sets of experiments are conducted, one for small-scale networks, 500 nodes, and another for large-scale networks, 15,000 nodes. Here, the experiment is conducted only with different sinkholes, 1, 2, 3, and 5. It is believed that experimenting with different sink node will not add that much information to the lifetime analysis.

Figure 15 shows the lifetime of small-scale networks. The lifetime is computed in terms of simulation running time and measured in milliseconds (ms). It is clear from the figure that TBDS has the largest lifetime with different sinkholes, between 15,000 ms and 14,700 ms, for 1–5 sinkholes, followed by MBIDS. ABC still gives a reasonable lifetime with different sinkholes, and its performance is much better than S-LEACH and MS-LEACH. For instance, the lifetime of MS-LEACH and S-LEACH

is the same when three and five sink nodes are used, 6000 ms. However, they reach almost 7000 ms with one and two sink nodes.

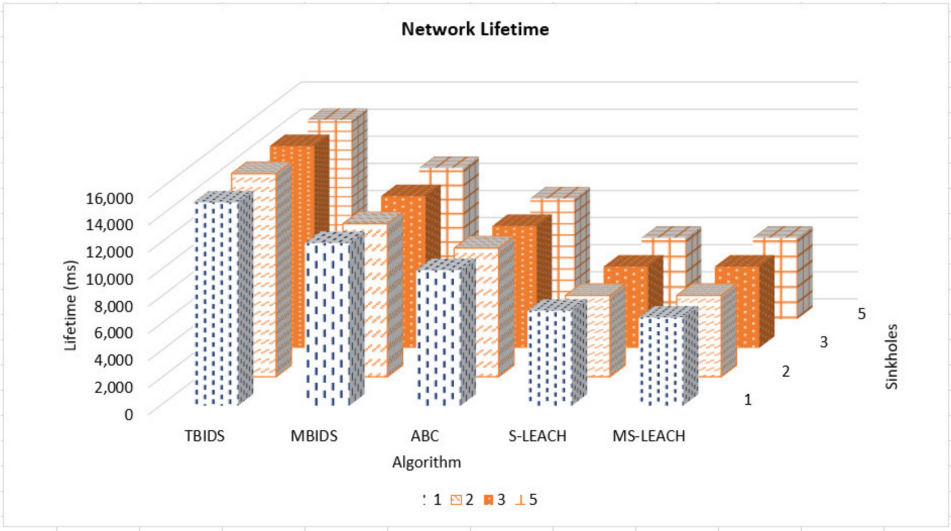


Figure 15. Lifetime for small-scale networks.

Figure 16 shows the network lifetime for large-scale networks, 15,000 nodes. Again, TBIDS gives the best lifetime, followed by MBIDS and ABC, with different numbers of sinkholes. The lifetime of S-LEACH and MS-LEACH is the worst with all of the sinkholes. However, it is worth noting that the lifetime difference between MBIDS and ABC is not that large; it is, on average, 2000 ms. Besides, the lifetime of the MBIDS is affected by the overhead control messages by a small value, on average, 3.5%, where the control messages drop messages with almost 0.01%.

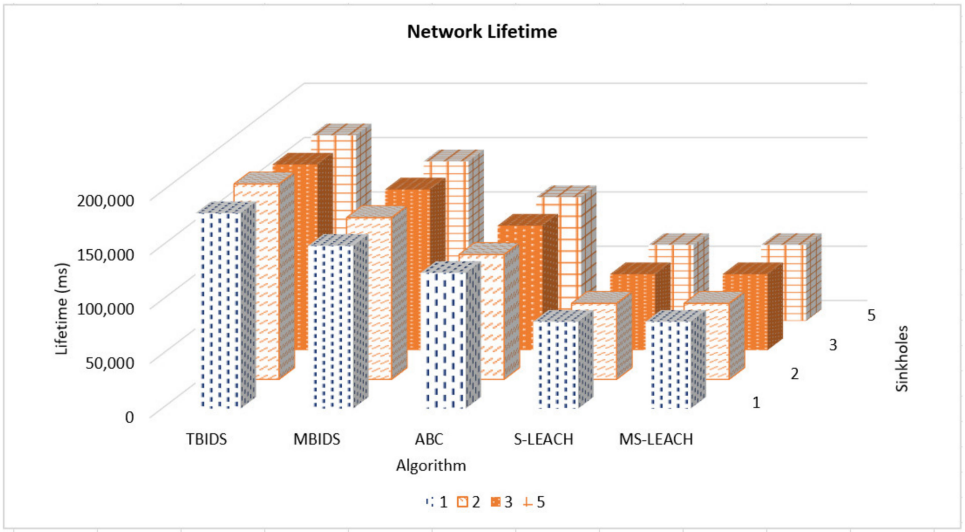


Figure 16. Lifetime for large-scale networks.

Figure 17 examines the network lifetime with 8000 sensor nodes and 10, 20, and 30 sink nodes. For these experiments to be conducted, huge resources are required to get the average results. However, it is important to look at the network from a different point of view. As can be seen in the figure, the results confirm the enhancement of the network operations mentioned in the previous sections where the lifetime of TBIDS is still much better than others. Besides, the algorithms could be ranked as TBIDS, MBIDS, ABC, S-LEACH, and MS-LEACH in terms of lifetime. At the same time, the average

lifetime of the algorithms seems to be much better when the number of sink nodes increases. It is also noticeable that ABC, S-LEACH, and MS-LEACH are enhanced with a better rate than TBIDS and MBIDS. Nonetheless, it has been noticed that the best lifetime is accomplished at three special cases where sink nodes are distributed on the borders of the monitored field, around the center of the monitored area, and when even clusters are generated.

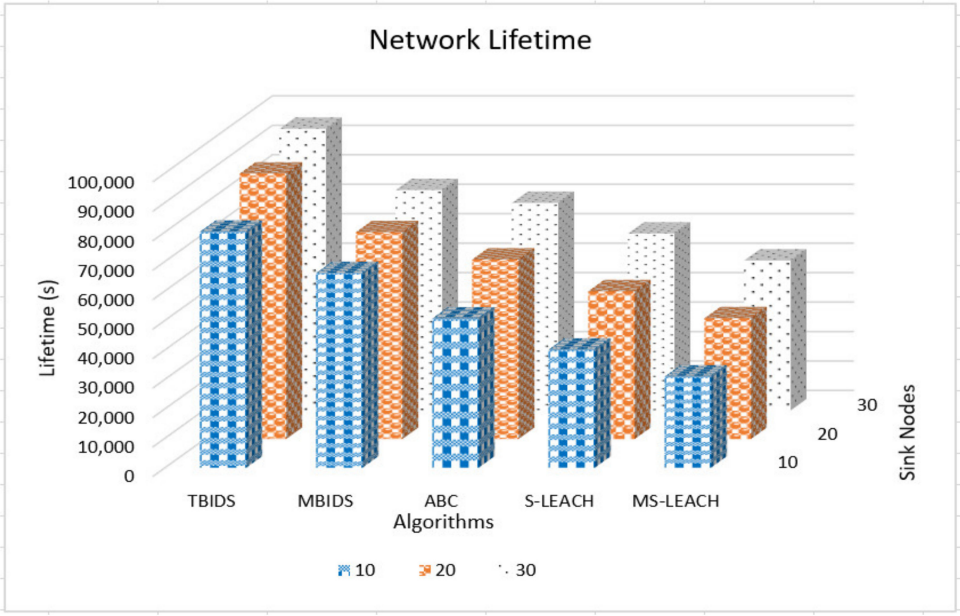
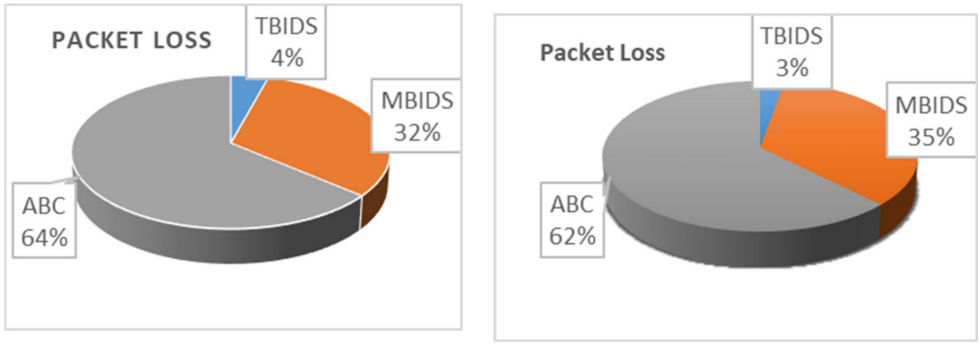


Figure 17. Lifetime for large-scale networks, 8000 sensor nodes, with 10, 20, and 30 sink nodes.

5.2.4. Packet Loss

In this set of experiments, the packet loss due to the sinkholes is measured. This is an indicator of how fast the detection process has occurred. However, for a fair comparison, TBIDS depends mainly on the detection period set at the sink node during its configuration time. In this set of experiments, the detection is assumed to be done with each message received. Besides, adding more than one sink node enhances the detection accuracy, as seen before. Therefore, it has been found that it is beneficial to examine the proposed algorithms with the worst sinkhole scenario and the best sink node scenario. It is assumed that there are five sinkholes and five sink nodes with a fixed number of sensors, 500 and 15,000 nodes. However, after investigating S-LEACH and MS-LEACH, it has been found that once CHs are compromised, no message will be sent to the sink node or at least modified by the adversary. So, both algorithms are excluded from those sets of experiments.

Figure 18a shows the packet loss for 500 nodes network with five sink nodes and five sinkholes. The packet loss for TBIDS, MBIDS, and ABC is 2%, 15%, and 30%, respectively. This is also an indicator of the speed of sinkhole detection. The same results are confirmed in Figure 18b, where a large-scale sensor network is with 15,000 nodes, five sink nodes, and five sinkholes. However, it has been noticed that TBIDS shows a fixed packet loss percentage, and the ABC packet loss is 45%, and MBIDS increases to 25%.



(**a**) (**b**)

Figure 18. Packet loss (**a**) for small-scale networks, (**b**) for large-scale networks.

Figure 19 shows the results of another set of experiments in which 8000 sensor nodes are deployed in the monitored field with a different number of sink nodes (10, 20, and 30). The figure shows significant improvement in the packet loss percentage, where TBIDS reaches almost 1% when 30 sink nodes are used. Besides, ABC algorithm performance enhances from 55% to 37%, which is an immense achievement.

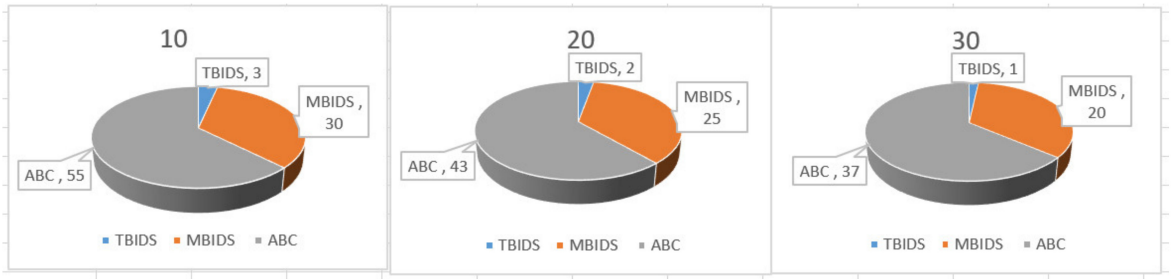


Figure 19. Packet loss for large-scale networks, 8000 sensor nodes, with 10, 20, and 30 sink nodes.

6. Discussion

Based on the previous extensive set of experiments, it has been shown that TBIDS and MBIDS are overperforming the ABC, S-LEACH, and MS-LEACH in almost every measured criterion. This is due to the design nature of TBIDS and MBIDS. In TBIDS, the detection is fully occurred at the sink node as well as in MBIDS. Besides, TBIDS is a threshold-based algorithm where the required computation is very light, and it could be done in a small number of processing cycles. It also checks only the CHs on the received message route; so, it does not have to go through all of the routing table records. On the other hand, MBIDS has to receive the Ctrl message from two different paths; therefore, there is a delay in receiving them. At the same time, the Ctrl message might be lost, and the sink node has to wait for the timeout period. This explains the high percentage of packet loss in MBIDS. ABS, unfortunately, depends on artificial Bees, where it takes some time to stabilize. This also interprets the lower performance of the ABC algorithm. Other algorithms, such as S-LEACH and MS-LEACH, work perfectly against sinkhole attack as long as the used secure keys are not compromised. Once a node is compromised, the sinkhole attack could go without detection. In addition, they require extra overhead in terms of messages and computation due to the pair secure keys and their authentication process. It has been noticed that the number of sink nodes has a great effect on network performance. The results of 8000 sensor nods and 30 sink nodes show that it is recommended to reduce the number of nodes to be served by the sink node. However, that puts a burden on the accurate threshold values required by the algorithms.

The weakness of the TBIDS algorithm is that it bases its decision on certain thresholds. The thresholds could differ from one network to another according to its size and network configurations. This threshold has to be learned at the setup phase of the network. In this paper, the trial and error approach is followed to find the best threshold values, which is a very exhaustive and time-consuming process. It could be more efficient to have an artificial intelligence training method to identify the best threshold values for different types of networks. In fact, this is scheduled as part of my future work.

MBIDS also suffers from two overheads—a control message that is sent along with the data message and the same control message that is sent in a separate path. This causes a slight problem in the detection time as well as in packet loss performance. In addition, the sink node has to investigate the whole route to check on the sinkhole attack.

7. Conclusions

This paper has discussed the importance of smart cities applications and the WSN as a core network in smart cities. Due to the limitations of the sensing devices, it is exposed to different security attacks. The paper has discussed one of the critical attacks on WSNs—sinkhole attack. Therefore, the paper has proposed two intrusion detection and prevention algorithms for the sinkhole attack. The first algorithm is based on multipath routing, MBIDS, where a control message has to be sent with the data message as well as through a separate route. The second algorithm is based on a threshold that is checked for each cluster head. Those two proposed intrusion detection algorithms are compared to three recent algorithms, which are ABC, S-LEACH, and MS-LEACH. Through an extensive set of experiments, the results show that TBIDS and MBIDS overperform all of the three algorithms using different criteria. It is worth mentioning that TBIDS results are very light and promising, and TBIDS could be implemented on a small-scale and large-scale network. The possible extension of this paper could be in three directions: (1) Investigating different methodologies to identify the best threshold value for TBIDS algorithm; (2) distributed intrusion detection systems where cluster heads can cooperate to detect the sinkhole node and isolate it; (3) reducing the overhead in MBIDS and lookup operation in the routing table at the sink node.

In addition, the future direction of smart cities is to have sensors that are smart enough to perform data analysis to its neighbors' behavior and discover any intrusion process gaining on. It is believed that smart cities will be designed based on security as the first approach, and its planning will be focused on its potential risk. For instance, 5G technology is extending the connectivity in smart cities, and at the same time, it opens an opportunity for attackers to step in. Therefore, a holistic security approach has to come to the play during the design of the smart city. It is also expected that artificial intelligence (AI) and big data (BD) analytics will be the "city brain" of smart cities where every "thing" could be represented as a neuron in the huge smart city brain. Therefore, the potential research directions in smart cities could be in sensing technologies, security, AI, and BD. Certainly, communication will be a major factor in smart cities.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Naphade, M.; Banavar, G.; Harrison, C.; Paraszczak, J.; Morris, R. Smarter cities and their innovation challenges. *Computer* **2011**, *44*, 32–39. [CrossRef]
2. Ijaz, S.; Ali, M.; Khan, A.; Ahmed, M. Smart cities: A survey on security concerns. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*. [CrossRef]
3. Tian, M.; Jiao, W.; Liu, J.; Ma, S. A charging algorithm for the wireless rechargeable sensor network with imperfect charging channel and finite energy storage. *Sensors* **2019**, *19*, 3887. [CrossRef] [PubMed]

4. Zou, T.; Xu, W.; Liang, W.; Peng, J.; Cai, Y.; Wang, T. Improving charging capacity for wireless sensor networks by deploying one mobile vehicle with multiple removable chargers. *Ad. Hoc. Netw.* **2017**, *63*, 79–90. [CrossRef]
5. Merenda, M.; Praticò, F.G.; Fedele, R.; Carotenuto, R.; Corte, F.G.D. A real-time decision platform for the management of structures and infrastructures. *Electronics* **2019**, *8*, 1180. [CrossRef]
6. Sohrabi, K.; Gao, J.; Ailawadhi, V.; Pottie, G.J. Protocols for self-organization of a wireless sensor network. *IEEE Pers. Commun.* **2000**, *7*, 16–27. [CrossRef]
7. Saglam, O.; Dalkili, M.E. A self organizing multihop clustering protocol for wireless sensor networks. In Proceedings of the 2009 Fifth International Conference on Mobile Ad-hoc and Sensor Networks, Fujian, China, 14–16 December 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 33–40.
8. Kacimi, R.; Dhaou, R.; Beylot, A.-L. Energy-aware self-organization algorithms for wireless sensor networks. In Proceedings of the IEEE GLOBECOM 2008–2008 IEEE Global Telecommunications Conference, New Orleans, LO, USA, 30 November–4 December 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 1–5. [CrossRef]
9. Loganathan, S.; Arumugam, J. Clustering algorithms for wireless sensor networks survey. *Sens. Lett.* **2020**, *18*, 143–149. [CrossRef]
10. Aghbari, A.Z.; Khedr, A.M.; Osamy, W.; Arif, I.; Agrawal, D.P. Routing in wireless sensor networks using optimization techniques: A survey. *Wirel. Pers. Commun.* **2020**, *111*, 2407–2434. [CrossRef]
11. Benhaddou, D.; Al-Fuqaha, A. (Eds.) *Wireless Sensor and Mobile Ad-Hoc Networks: Vehicular and Space Applications*; Springer: New York, NY, USA, 2015. [CrossRef]
12. Yang, Q.; Zhu, X.; Fu, H.; Che, X. Survey of security technologies on wireless sensor networks. *J. Sens.* **2015**, *2015*, 1–9. [CrossRef]
13. Aboshosha, B.W.; Dessouky, M.M.; Ramadan, R.; EI-Sayed, A. Evaluation of lightweight block ciphers based on general feistel structure (GFS). *WAS Sci. Nat.* **2020**, *2*, 1–8.
14. Mingyan, L.; Koutsopoulos, I.; Poovendran, R. Optimal jamming attack strategies and network defense policies in wireless sensor networks. *IEEE Trans. Mob. Comput.* **2010**, *9*, 1119–1133. [CrossRef]
15. Osanaiye, O.; Choo, K.-K.R.; Dlodlo, M. Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework. *J. Netw. Comput. Appl.* **2016**, *67*, 147–165. [CrossRef]
16. Ngai, E.C.H.; Liu, J.; Lyu, M.R. An efficient intruder detection algorithm against sinkhole attacks in wireless sensor networks. *Comput. Commun.* **2007**, *30*, 2353–2364. [CrossRef]
17. Saranya, T.; Sridevi, S.; Deisy, C.; Chung, T.D.; Khan, M.K.A.A. Performance analysis of machine learning algorithms in intrusion detection system: A review. *Procedia Comput. Sci.* **2020**, *171*, 1251–1260. [CrossRef]
18. Krontiris, I.; Giannetsos, T.; Dimitriou, T. Launching a sinkhole attack in wireless sensor networks; the intruder side. In Proceedings of the 2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, Avignon, France, 12–14 October 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 526–531. [CrossRef]
19. Zhang, R.; Xiao, X. Intrusion detection in wireless sensor networks with an improved NSA based on space division. *J. Sens.* **2019**, *2019*, 1–20. [CrossRef]
20. Li, K. Scheduling parallel tasks with energy and time constraints on multiple manycore processors in a cloud computing environment. *Futur. Gener. Comput. Syst.* **2018**, *82*, 591–605. [CrossRef]
21. Sharmila, S.; Umamaheswari, G. Detection of Sinkhole attack in wireless sensor networks using message digest algorithms. In Proceedings of the 2011 International Conference on Process Automation, Control and Computing, Coimbatore, India, 20–22 July 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 1–6. [CrossRef]
22. El-Saadawy, M.; Shaaban, E. Enhancing S-LEACH security for wireless sensor networks. In Proceedings of the 2012 IEEE International Conference on Electro/Information Technology, Indianapolis, IN, USA, 6–8 May 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 1–6. [CrossRef]
23. Ferreira, A.C.; Vilaça, M.A.; Oliveira, L.B.; Habib, E.; Wong, H.C.; Loureiro, A.A. On the security of cluster-based communication protocols for wireless sensor networks. In Proceedings of the International Conference on Networking, Reunion Island, France, 17–21 April 2005; pp. 449–458. [CrossRef]
24. Papadimitriou, A.; Le Fessant, F.; Viana, A.C.; Sengul, C. Cryptographic protocols to fight sinkhole attacks on tree-based routing in wireless sensor networks. In Proceedings of the 2009 5th IEEE Workshop on Secure Network Protocols, Princeton, NJ, USA, 13 October 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 43–48. [CrossRef]

25. Chen, C.; Song, M.; Hsieh, G. Intrusion detection of sinkhole attacks in large-scale wireless sensor networks. In Proceedings of the 2010 IEEE International Conference on Wireless Communications, Networking and Information Security, Beijing, China, 25–27 June 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 711–716. [CrossRef]
26. Coppolino, L.; D'Antonio, S.; Romano, L.; Spagnuolo, G. An intrusion detection system for critical information infrastructures using wireless sensor network technologies. In Proceedings of the 2010 5th International Conference on Critical Infrastructure (CRIS), Beijing, China, 20–22 September 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 1–8. [CrossRef]
27. Sheela, D.; Kumar, C.N.; Mahadevan, G. A non cryptographic method of sink hole attack detection in wireless sensor networks. In Proceedings of the 2011 International Conference on Recent Trends in Information Technology (ICRTIT), Chennai, India, 6 May 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 527–532.
28. Nithiyanandam, N.; Latha, P. Artificial bee colony based sinkhole detection in wireless sensor networks. *J. Ambient Intell. Humaniz. Comput.* **2019**, 1–14. [CrossRef]
29. Teng, L.; Zhang, Y. SeRA: A secure routing algorithm against sinkhole attacks for mobile wireless sensor networks. *Int. Conf. Comput. Model. Simul.* **2010**, 79–82. [CrossRef]
30. Liu, S.; Yang, Y.; Wang, W. Research of AODV routing protocol for Ad Hoc Networks1. *AASRI Procedia* **2013**, *5*, 21–31. [CrossRef]
31. Antil, P.; Malik, A. Hole Detection for quantifying connectivity in wireless sensor networks: A survey. *J. Comput. Netw. Commun.* **2014**, *2014*, 1–11. [CrossRef]
32. Zhu, S.; Setia, S.; Jajodia, S. LEAP+: Efficient Security mechanisms for large-scale distributed sensor networks. *ACM Trans. Sens. Netw.* **2006**, *2*, 500–528. [CrossRef]
33. Khaled, A.E.E.-D.; Ramadan, R.A.; Fayek, M.B. Vegk: Secure clustering for efficient operation in WSNs. *Commun. Comput. Secur.* **2013**, *3*. [CrossRef]
34. Perrig, A.; Szewczyk, R.; Tygar, J.D.; Wen, V.; Culler, D.E. SPINS: Security protocols for sensor networks. *Wirel. Netw.* **2002**, *8*, 521–534. [CrossRef]
35. Ngai, E.C.H.; Liu, J.; Lyu, M. On the intruder detection for sinkhole attack in wireless sensor networks. In Proceedings of the 2006 IEEE International Conference on Communications, Istanbul, Turkey, 11–15 June 2006; IEEE: Piscataway, NJ, USA, 2006; pp. 3383–3389. [CrossRef]
36. Salehi, S.; Razzaque, M.A.; Naraei, P.; Farrokhtala, A. Detection of sinkhole attack in wireless sensor networks. In Proceedings of the 2013 IEEE International Conference on Space Science and Communication (IconSpace), Coimbatore, India, 20–22 July 2011; IEEE: Piscataway, NJ, USA, 2013; pp. 361–365. [CrossRef]
37. Wang, S.-S.; Yan, K.-Q.; Wang, S.-C.; Liu, C.-W. An integrated intrusion detection system for cluster-based wireless sensor networks. *Expert Syst. Appl.* **2011**, *38*, 15234–15243. [CrossRef]
38. Hamedheidari, S.; Rafeh, R. A novel agent-based approach to detect sinkhole attacks in wireless sensor networks. *Comput. Secur.* **2013**, *37*, 1–14. [CrossRef]
39. Zhan, G.; Shi, W.; Deng, J. Design and implementation of TARF: A trust-aware routing framework for WSNs. *IEEE Trans. Dependable Secur. Comput.* **2012**, *9*, 184–197. [CrossRef]
40. Liu, Y.; Ma, M.; Liu, X.; Xiong, N.N.; Liu, A.; Zhu, Y. Design and analysis of probing route to defense sink-hole attacks for internet of things security. *IEEE Trans. Netw. Sci. Eng.* **2020**, *7*, 356–372. [CrossRef]
41. Cupcarbon. Available online: http://cupcarbon.com/ (accessed on 1 June 2020).
42. BenSaleh, M.S.; Saida, R.; Kacem, Y.H.; Abid, M. Wireless sensor network design methodologies: A survey. *J. Sens.* **2020**, *2020*, 1–13. [CrossRef]