OPTIMIZING SOURCE ANONYMITY OF WIRELESS SENSOR NETWORKS AGAINST GLOBAL ADVERSARY USING FAKE PACKET INJECTIONS

Anas Bushnag

Under the Supervision of

Dr. Ausif Mahmood (Advisor)
Dr. Abdelshakour Abuzneid (Co-Advisor)

DISSERTATION
SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE
AND ENGINEERING
THE SCHOOL OF ENGINEERING
UNIVERSITY OF BRIDGEPORT
CONNECTICUT

December, 2017

OPTIMIZING SOURCE ANONYMITY OF WIRELESS SENSOR NETWORKS AGAINST GLOBAL ADVERSARY USING FAKE PACKET INJECTIONS

Anas Bushnag

Under the Supervision of Dr. Ausif Mahmood

Approvals

Committee Members

Name	Signature	Date
Dr. Ausif Mahmood	[illegible]	12-7-2017
Dr. Abdelshakour Abuzneid	[illegible]	12/7/2017
Dr. Miad Faezipour	[illegible]	12/7/2017
Dr. Xingguo Xiong	[illegible]	12/07/2017.
Dr. Amir Esmailpour	[illegible]	12/7/2017

Ph.D. Program Coordinator

Dr. Khaled M. Elleithy	[illegible]	12/7/2017

Chairman, Computer Science and Engineering Department

Dr. Ausif Mahmood	[illegible]	12-7-2017

Dean, School of Engineering

Dr. Tarek M. Sobh	[illegible]	1/2/2018

OPTIMIZING SOURCE ANONYMITY OF WIRELESS SENSOR NETWORKS AGAINST GLOBAL ADVERSARY USING FAKE PACKET INJECTIONS

ABSTRACT

Wireless Sensor Networks (*WSNs*) have been utilized for many applications such as tracking and monitoring of endangered species in a national park, soldiers in a battlefield, and many others, which require anonymity of the origin, known as the Source Location Privacy (*SLP*). The aim of *SLP* is to prevent unauthorized observers from tracing the source of a real event (an asset) by analyzing the traffic of the network. We develop the following six techniques to provide anonymity: Dummy Uniform Distribution (*DUD*), Dummy Adaptive Distribution (*DAD*), Controlled Dummy Adaptive Distribution (*CAD*), Exponential Dummy Adaptive Distribution (*EDAD*), Exponential Dummy Adaptive Distribution Plus One (*EDADP1*), and Exponential Dummy Adaptive Distribution Plus Two (*EDADP2*). Moreover, an enhanced version of the well-known *FitProbRate* technique is also developed. The purpose of these techniques is to overcome the anonymity problem against a global adversary model that has the capability of analyzing and monitoring the entire network.

We perform an extensive verification of the proposed techniques via simulation, statistical, and visualization approaches. Three analytical models are developed to verify the performance of our techniques: A *Visualization* model is performed on the simulation data to confirm anonymity. A *Neural Network* model is developed to ensure that the introduced techniques preserve *SLP*. In addition, a *Steganography* model based on statistical empirical data is implemented to validate the anonymity of the proposed techniques. The Simulation demonstrates that the proposed techniques provide a reasonable delay, delivery ratio, and overhead of the real event's packets while keeping a high level of anonymity.

Results show that the improved version of *FitProbRate* massively reduces the number of operations needed to detect the distribution type of a data sequence despite the number of intervals when compared to the original. A comprehensive comparison between *EDADP1*, *EDADP2*, and *FitProbRate* in terms of the average delay, anonymity level, average processing time, Anderson-Darling test, and polluted scenarios is conducted. Results show that all three techniques have a similar performance regarding the average delay and Anderson-Darling test. However, the proposed techniques outperform *FitProbRate* in terms of anonymity level, average processing time, and polluted scenarios. *WSN* applications that need privacy can select the suitable proposed technique based on the required level of anonymity with respect to delay, delivery ratio, and overhead.

ACKNOWLEDGMENTS

My thanks are wholly devoted to God who has helped me along the way to complete this work successfully. I owe a debt of gratitude to my mother, grandmother, wife, and son for understanding and encouragement. You are always there for me. I will be ever grateful for my father, and I am sorry that he has not lived to see me graduate. His memory will be with me always.

I am honored that my work has been supervised by Dr. Ausif Mahmood and Dr. Abdelshakour Abuzneid. You supported me greatly and were always willing to help me. I want to thank you for your excellent cooperation and for all of the opportunities I was given to conduct my research and further my dissertation at the University of Bridgeport.

I would like to thank the committee members, Dr. Miad Faezipour, Dr. Xingguo Xiong, Dr. Amir Esmailpour, and Dr. Khaled M. Elleithy for their valuable guidance and support.

In addition, I would like to thank Dr. Charles Campbell, Camy Deck, and Rebecca Bruckenstein from the Tutoring and Learning Center at the University of Bridgeport for the English language support and their assistance in helping me to prepare my dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

ABBREVIATIONS

A-D	Anderson-Darling
ADC	Analog-to-Digital-Converter
APR	Anonymous Path Routing
ASLP	Aggregation-based Source Location Protection Scheme
CAD	Controlled Dummy Adaptive Distribution
CDF	Cumulative Distribution Function
ConstRate	Constant Rate
CSPSLP	Cloud-Based Scheme for Protecting Source-Location Privacy
DAD	Dummy Adaptive Distribution
DDS	Dummy Data Sources
DoS	Denial-of-Service
DRAA	Distribution Resource Allocation Algorithm
DUD	Dummy Uniform Distribution
DWUS	Dummy Wake-up Scheme
EDAD	Exponential Dummy Adaptive Distribution
EDADP1	Exponential Dummy Adaptive Distribution Plus One
EDADP2	Exponential Dummy Adaptive Distribution Plus Two
FIFO	First-In-First-Out

FitProbRate Fitted Probabilistic Rate

GAFG Group Algorithm for Fake-traffic Generation

GFS General Fake Source

GOA Globally Optimal Algorithm

GPS Global Positioning System

GPSR Greedy Perimeter Stateless Routing

HGA Heuristic Greedy Algorithm

ID Identifier

IDS Intrusion Detection Systems

MTM Mobile Trusted Module

NAA Naive Algorithm

NS-2 Network Simulator 2

OFS Optimal Filtering Scheme

OMNeT++ Objective Modular Network Testbed in C++

OPNET Optimized Network Engineering Tools

OSAP Optimal-cluster-based Source Anonymity Protocol in Delay-sensitive Wireless Sensor Networks

OTcl Object Tcl

PA-SLP Pollution Avoiding Source Location Privacy

PBA Probabilistic Algorithm

PeCo	Periodic Collection
PFS	Proxy-based Filtering
ProbRate	Probabilistic Rate
RCM	Recurrent Clustering Mechanism
RFID	Radio Frequency Identification
SECLOUD	Source and Destination Seclusion using Clouds
SLP	Source Location Privacy
SoSi	Source Simulation
SPR	Separate Path Routing
SVM	Support Vector Machine
TCH-WSN	Trusted Computing Enabled Heterogeneous
TCP	Transmission Control Protocol
TESP	Efficient Privacy Preservation
TFS	Tree-based Filtering
TMP	Trusted Platform Module
TTHS	Triple-type Homomorphic Signature
TTL	Time-To-Live
UDP	User Datagram Protocol
UHT	Unobservable Handoff Trajectory
WSNs	Wireless Sensor Networks

CHAPTER 1: INTRODUCTION

1.1 Research Problem and Scope

WSNs consist of homogeneous, small, and low-cost sensor nodes [1,2] that have limitations in resources such as processing power, memory [3–7] and battery life [8,9]. Since *WSNs* have limited resources [10,11], a sensor node should only compute basic operations [3]. Usually, sensors are used to sense information such as temperature, humidity, and light [12–15]. Sensor nodes notify the sink through intermediate sensor nodes that act as forwarders to pass the data to its final destination [16]. *WSNs* can also be used for monitoring and tracking applications such as monitoring and tracking endangered species in a national park, patients in a hospital, or soldiers in a battlefield [17–23]. In *WSNs*, communication between nodes consumes more power than the processing and computation inside the sensor node itself [3,24]. *WSNs* can be deployed in remote locations [25] that are unreachable by wired networks such as a hostile environment [26,27] or a vast forest [28]. Thus, security of sensor networks is critical, and it should be addressed very carefully [29–31].

Security in *WSNs* are classified into content threats [32–34] and context threats [35–38]. Content security focuses on protecting the content of packets by providing confidentiality, authentication, integrity [39], and many other encryption techniques [40–42], whereas context security such as *SLP* focuses on concealing the location of the source node [30,43–45]. The anonymity of a node means that this node should be untraceable under any statistical analyses applied by an adversary [29,46]. The main objective of *SLP* is to keep the originator node untraceable and unlinkable. Untraceability means that the adversary is unable to trace back the source node [29,47], whereas unlinkability means that the adversary cannot gain the identity of the origin [29]. In *SLP*, the real event (an asset) has three parameters: event type, event time, and event location [30]. These parameters are continually targeted by the adversary to gain information about them (Figure 1.1). Even after employing the most sophisticated encryption techniques, context security requires more convoluted techniques to secure the location of the source as the adversary attempts to locate the source by eavesdropping on the network traffic. Sink Location Privacy is another context security concern [48]. It prevents adversaries from gaining information about the sink location. The anonymity of the sink node is an entirely different problem. The focus of this work is only on *SLP*.

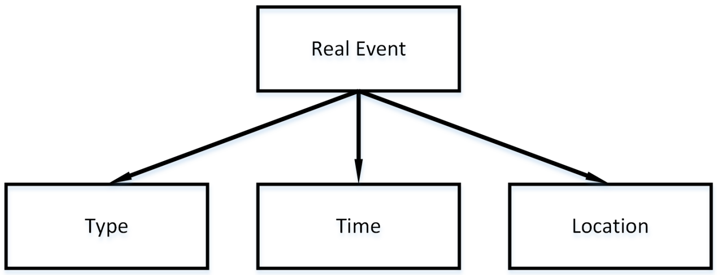


Figure 1.1. Real event parameters.

Adversaries can employ two types of attacks: active and passive [49,50]. An active attack occurs when the adversary attempts to alter the network traffic by modifying the packets' header, the packets' content, or even by injecting new packets into the network to apply some attacks such as Denial-of-Service (*DoS*). In contrast, a passive attack occurs when an adversary analyzes the network traffic without alteration by observing which sensor nodes are transmitting and which sensor nodes are not. A passive attack is more difficult to detect than an active attack because no modifications are noticed by the system. This work attempts to defend against passive attacks.

Two types of adversaries exist: local and global. A local adversary has a partial view of the network, limited resources, and is only able to analyze local traffic. The local adversary can be countered by modifying the existing routing protocols [51]. In contrast, a global adversary has a full view of the network, unlimited power, sufficient resources, and can analyze the entire traffic of the network [52]. A global adversary can apply sophisticated analyses such as *Rate Monitoring* and *Time Correlation* attacks [52–54]. Global adversaries require more sophisticated methods to be countered. This work only considers the global adversary as the attacking model.

In *WSNs*, tracking devices are often attached to assets, e.g., a Radio Frequency Identification (*RFID*) tag. A *RFID* can either employ active or passive tags. The active tag has a battery and is able to send signals to the sensor nodes, which can be utilized to simulate the movement of an asset at different locations within the network, whereas the passive tag does not have a battery and cannot send signals [55]. This work only considers passive tags because they are cheaper and more applicable when the *WSNs* is out of reach.

A variety of schemes exist to overcome context threats against a global adversary such as Separate Path Routing (*SPR*), Network Location Anonymization, Network Coding, and Dummy Data Sources (*DDS*) [4]. *SPR* creates multiple paths from a source to the sink, which means each packet of an event uses a different route to the destination [56]. Network Location Anonymization hides the identity of the source by using pseudonyms [57, 58]. Network Coding divides a packet into smaller pieces. These pieces will follow different routes to the sink [59, 60]. *DDS* [61] creates fake sources, which generate dummy traffic to hide and obfuscate the real traffic inside. *DDS* is by far the most effective method against global adversaries because the use of fake sources and dummy traffic confuses the adversary about the identity and location of the source [52,62]. *DDS* approach will be used in the proposed techniques because it provides higher anonymity than other methods.

There are three types of packets: real, fake, and corrupted. Real packets carry information related to the real event such as its location. Fake packets do not carry any information about the real event. They are often utilized to mislead and confuse the adversary about the actual location of the real event [4]. Corrupted packets are further classified into two types: injected and modified. The injected packets mean that new packets are inserted into the network by an adversary to apply, e.g., *DoS*. The modified packets mean that the existing packets are altered by an adversary [63]. Since the utilized global adversary model is passive, this work only considers real and fake packets. Figure 1.2 shows the different types of packets.

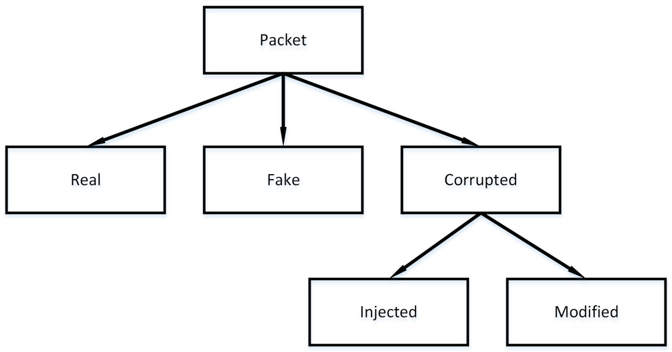


Figure 1.2. Types of packets.

The fundamental idea behind the proposed techniques is as follows: When a real event (an asset) is detected, it should be reported to the sink. Therefore, hiding the real event is mandatory, the network should be injected with dummy traffic using the lowest transmission rate as possible to confuse the adversary. This low transmission rate is needed to minimize the communication overhead. The dummy traffic is injected into the network in a probabilistic manner that leads to time and location privacy of the source node. In this work, the anonymity of an asset means hiding the existence of the asset at a certain time located nearby a sensor node, which subsequently leads to *SLP*.

This work is about optimizing source anonymity of *WSNs* with acceptable delay, delivery ratio, and overhead against a global adversary model that is capable of employing sophisticated traffic analyses. A *WSN* application that needs privacy should be able to select one of the proposed techniques that fits its required level of anonymity with respect to delay, delivery ratio, and overhead.

1.2 Motivation behind the Research

The detection of significant events such as the location of endangered species in a national park or soldiers in a battlefield needs to be securely communicated to the sink. Protecting assets from being discovered or even captured is essential. In order to protect these assets, the defensive team (the system) must make the offensive team (the adversary) confused about the existence of the real event. Therefore, the defensive team must inject the network with dummy traffic to prevent the adversary from tracing back the origin by analyzing the network traffic. Traffic analysis can be *Rate Monitoring*, *Time Correlation*, or *Size/Structure Correlation*.

1.3 Contributions of the Proposed Research

We developed several techniques that outperform existing approaches in providing source anonymity against a global adversary model. Most of the current techniques compare their work only to the previous ones. However, in this work, we provided a validation of the proposed techniques by developing analytical models to confirm the high anonymity of our techniques.

- Six different techniques: *DUD*, *DAD*, *CAD*, *EDAD*, *EDADP1*, and *EDADP2* are developed to provide source anonymity with acceptable delay, delivery ratio, and overhead.

- The proposed techniques can protect multiple real packets concurrently, unlike many of the previous techniques that can only protect one real packet at a time.

- Different analytical models are developed: *Visualization*, *Neural Network*, and *Steganography* to confirm the validation of the proposed techniques.

- An enhanced version of the *FitProbRate* technique is developed that reduces the number of operations by applying the Anderson-Darling test (A-D) on the last ten elements of the sequence instead of the entire sequence as implemented in the original *FitProbRate* technique.

- A comprehensive comparison between *EDADP1*, *EDADP2*, and *FitProbRate* is conducted to confirm the high performance of the proposed techniques. The following metrics are used in the comparison: average delay, anonymity level, average processing time, A-D test, and polluted scenarios.

- A novel software architecture for a specialized network simulator is developed, and targeted towards analysis and verification of anonymity algorithms.

1.4 Research Hypothesis

The global adversary is confused about the existence of the real event (an asset) when the *WSN* is injected with dummy traffic due to the large amount of noise introduced by the proposed techniques. In this work, Null (H_0) means that the adversary is confused about the existence of the real event. Alternative (H_1) means that the adversary is not confused about the existence of the real event.

The rest of the work is organized as follows: Chapter 2 discusses the background. Chapter 3 describes the literature. Chapter 4 discusses system models and proposed techniques. Chapter 5 shows the implementation and test plan. Chapter 6 presents the results.

CHAPTER 2: BACKGROUND

A sensor node consists of four components: sensing, processing, communication, and power subsystems [64]. The sensing subsystem senses the desired phenomena such as temperature and feeds the detected analog signal to an Analog-to-Digital-Converter (ADC) to prepare it for further processing. The processing subsystem is the core component of the sensor node architecture, and consists of a processor and memory. This subsystem is responsible for computation and making the data ready for transmission. The communication subsystem has a radio, and in some cases, its own processor to transmit, forward, or receive the data from/to neighboring nodes. Finally, the power subsystem generates DC power to provide the electric current to the other subsystems. All of these subsystems work together creating a functional sensor node that is able to interact with the environment and other nodes within its sensing range to achieve a particular task [3].

Security in *WSNs* has been a challenge because of the unique aspects they have. It is unsatisfactory to use the ordinary security mechanisms in *WSNs* due to the limitation of resources such as processing power and battery life. Sensors are independent and do not normally follow a central control entity because of their large scale and frequent topology changes. Therefore, traditional security solutions are inapplicable since they require significant overhead and sufficient memory. One of the basic defenses against security attacks is the ability to access network nodes physically. This is impractical in *WSNs* as many applications require sensor nodes to be deployed in remote and open locations that are difficult to reach, control, manage and protect from unauthorized physical accesses. Packets in *WSNs* are vulnerable to be lost or corrupted for several reasons such as routing failures or collisions. These challenges must be taken into consideration when developing a security technique for WSNs [3].

There are several kinds of attacks that can be a threat to the *WSNs* security such as *DoS*, routing attacks, transport layer attacks, data aggregation attacks, and privacy attacks, which are all shown in Figure 2.1. *DoS* occurs when an adversary prevents the network from functioning or providing the expected services to applications. *DoS* has two types of layer attacks: physical layer and link layer. One of the attacks on the physical layer is the *Jamming* attack; it interferes with radio frequencies of the sensor nodes preventing them from transmitting and receiving meaningful data. *Tampering* attack is also a physical layer attack that attempts to destroy or modify a sensor node physically. This attack might lead the adversary to obtain sensitive information that could lead to compromising the entire network. A *Collision* attack is a link layer attack that interferes with packet transmissions to make the network re-transmit the same packets repeatedly, which increases the overhead and power consumption [3].

A variety of routing attacks can be performed by an adversary. For example, a malicious node in the *Blackhole* attack and *Sinkhole* attack tries to convince the network that it is the data forwarder of many routes in the network. Once the malicious node receives the packets, this node drops them right away. Another attack is the *Selective Forwarding* attack, which is very similar to the *Blackhole* attack and *Sinkhole* attack. The only difference is that *Selective Forwarding* attack discards a certain number of packets based on specific criteria rather than dropping all incoming packets. Some other attacks target the on-demand routing protocols such as the *Rushing* attack. In this attack, the malicious node forwards all incoming route requests to nearby nodes without using the actual routing protocol policies, which involves more nodes in the route. Location-based routing protocols are weak against the *Sybil* attack that provides the adversary with multiple identities at different locations in the network. Legitimate nodes will think that the malicious node is one of its trusted neighboring nodes and start forwarding packets to the malicious node. *Wormhole* is another attack on routing. The *Wormhole* attack occurs when the malicious node has more capabilities such as bandwidth than other nodes in the network. This increase in capabilities might attract authorized nodes to forward their data to the unauthorized node since it has a high-speed connection. The *Wormhole* attack could help other attacks such as *Blackhole* to take place [3].

Transport layer protocols such as Transmission Control Protocol (*TCP*) and User Datagram Protocol (*UDP*) are vulnerable to the *Flooding* attack. A protocol such as *TCP* keeps state information allowing the adversary to send multiple connection requests that waste memory and reject any future connection requests even from authorized nodes. The *Desynchronization* attack attempts to block the transmission between two nodes by sending fake messages using a modified sequence number to both parties making each node believe that its packet has not been delivered. Therefore, a re-transmission is needed causing unnecessary overhead [3].

Data aggregation works by combining duplicated data from multiple sensor nodes to reduce the overhead and redundant information. There are many aggregation functions such as *Sum*, *Average*, *Count*, *Max*, and *Min* that can be easily modified by the adversary to make the network act differently [3].

Privacy attacks focus on analyzing the traffic of the network [35–38]. An adversary can obtain critical information by snooping on the network. The nature of *WSNs* facilitates attackers to monitor and capture the traffic between sensor nodes. Traffic analysis allows the adversary to identify the most important nodes in the network such as source and sink nodes or gain information about the hot-spot and high traffic regions in the network, known as the *Rate Monitoring* attack. *Time Correlation* is another privacy attack that monitors the difference between transmitting times of packets and if the network packets follow a specific distribution type in trying to find the relationship, e.g., transmitting times between real and fake packets. This could lead to exposing the location of the source node. Another attack is the *Size/Structure Correlation* [65], which focuses on the size and payload structure of packets to observe any differences. This work focuses on providing techniques that prevent privacy attacks against source nodes.

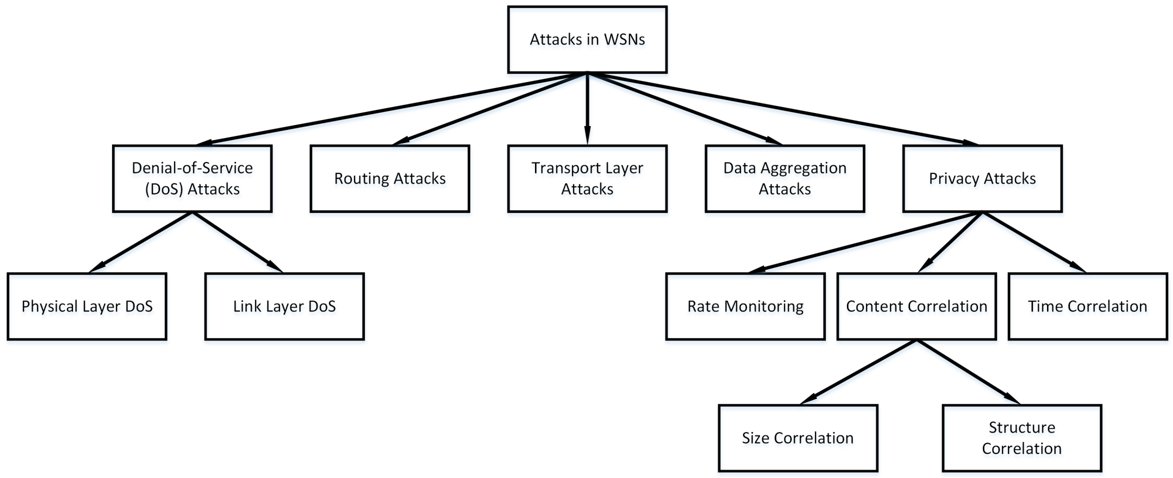


Figure 2.1. Types of attacks in *WSNs*.

CHAPTER 3: LITERATURE SURVEY

In this chapter, the existing techniques in the literature that rely on fake packet injections against a global adversary are presented. Additionally, two classifications of these techniques are conducted: The first classification categorizes the techniques based on the utilized characteristics. The second classification categorizes the techniques based on their assumptions for the global adversary.

3.1 Fake Packet Techniques

3.1.1 Periodic Collection (PeCo)

PeCo [66] is one of the first techniques that introduced the concept of fake packets against global adversaries. This technique works as follows: Each node in the network must obtain a shared individual key between itself and its neighboring nodes for encryption purposes. When the sensor node receives a packet, it decrypts and adds the packet to its buffer using the First-In-First-Out (*FIFO*) queuing mechanism. Every node has a timer that counts down; once the timer reaches zero, the first real packet in the buffer is encrypted and sent to the destination. However, if there is no real packet, a fake packet is generated and sent instead. On the receiving side, if a node received a fake packet, it discards the packet instantly. The main issue in *PeCo* is the buffer size. The network nodes should have a sufficient buffer size to manage all incoming packets. Overhead, power consumption and latency are also considered as serious issues in *PeCo* because the nodes generate a fake packet whenever there is no real packet to transmit.

3.1.2 Constant Rate (ConstRate)

The fundamental concept of *ConstRate* [54] is to divide the lifetime of the network into intervals. Packets are only sent or forwarded at these intervals whether they are real or fake to make them indistinguishable. If a node does not have a real packet to transmit during the interval, a fake packet is transmitted instead. However, the use of intervals concept increases the latency. In addition, this technique has a high power consumption due to the number of fake packets created to cover the real traffic.

3.1.3 Probabilistic Rate (ProbRate)

The difference between *ProbRate* [54] and *ConstRate* is that *ProbRate* selects the next interval to send or forward packets based on the exponential distribution to reduce the delay and number of fake packets. However, if the adversary knows μ, which is one over the transmission rate, and it is the only parameter in the exponential distribution, the network might be compromised. Therefore, the random number that is used to generate μ needs to be protected and unknown to adversaries.

3.1.4 Fitted Probabilistic Rate (FitProbRate)

FitProbRate [54] uses the same exponential distribution as *ProbRate* to generate dummy traffic. If a node detected a real event, it transmits real packets following the exact exponential distribution of the fake packets. Therefore, the adversary would be unable to distinguish the difference between real and fake packets. In order to reduce the traffic overhead, the transmission rate should be as low as possible; in return, this small transmission rate increases the delay relatively.

The network nodes generate a random number utilizing a unique seed to predict the following sending time interval. The seed can be known to the adversary, whereas the random number must be hidden. When a real event occurs, *FitProbRate* must use the same μ of the fake packets exponential distribution to avoid any *Time Correlation* attacks by the adversary. Concurrently, packets of the real event should be transmitted as soon as possible. Therefore, *FitProbRate* employs the *A-D* test to determine whether a series of intervals follow the exponential distribution or not. This is achieved by searching the first appropriate time interval that satisfies the *A-D* test, which at the same time does not break the exponential distribution sequence for the real packets. In case there is a scheduled fake interval, it is replaced by the real one. The fake interval will be rescheduled for later, as shown in Figure 3.1. Interval D is the fake one and it will be replaced by interval C, which has the real packet. Then, the fake packet in interval D will be rescheduled for transmission in interval E.

The disadvantages of this technique are as follows: First, it has significant traffic overhead that reduces the lifetime of the network. Second, using the *A-D* test every time a real event is detected increases power consumption and processing time. Lastly, transmission rate and delay cannot be controlled since *FitProbRate* does not provide a mechanism to ensure the maximum required delay by intolerant applications.

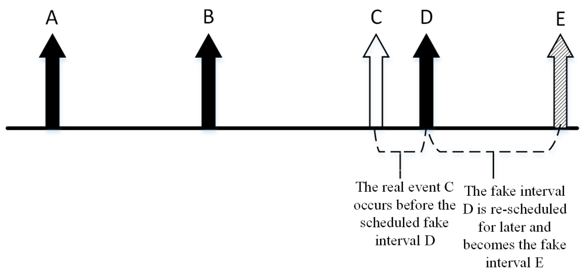

A
B
C
D
E
The real event C occurs before the scheduled fake interval D
The fake interval D is re-scheduled for later and becomes the fake interval E

Figure 3.1. An example of the *FitProbRate* technique.

3.1.5 Baseline

In the *Baseline* [67] technique, each node in the sensor network transmits real or fake messages pursuing a constant or exponential distribution. When a node detects a real event, it does not transmit the message immediately. Instead, the node waits for a while to ensure that the real packet follows the same distribution as the fake packets. As a result, the adversary cannot recognize the difference between real and fake events. However, this technique is very expensive for the network because it adds a massive amount of traffic overhead and decreases the delivery ratio of real packets since *Baseline* uses intervals to deliver the real event.

3.1.6 Proxy-based Filtering (PFS)

To overcome the issues of the *Baseline* technique such as high overhead and poor delivery ratio, *PFS* [67] was introduced. The primary concept of *PFS* is to hire some of the sensor nodes to act as designated proxies. These proxies filter the fake packets towards the sink, which reduces the overhead traffic while keeping the source anonymity.

In *PFS*, some of the sensor nodes are selected to filter the fake packets from neighboring nodes, as shown in Figure 3.2, which reduces the overhead as many of the fake packets are dropped before reaching the sink. A proxy node filters the packets to decide which packets will be forwarded and which packets will be dropped. *PFS*

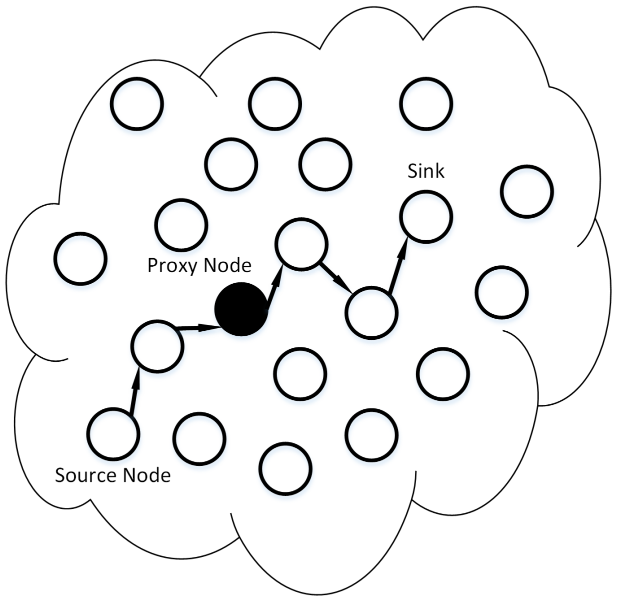


Figure 3.2. The filtration mechanism in *PFS*.

relies on the location of the proxy nodes; therefore, a proxy placement algorithm is performed to minimize the overhead of the network. Further, *PFS* should select the values of the proxies' buffer's parameters correctly such as size. This selection is necessary to handle the transmission delay of the real event at the source node. Authors of [67] claim that the *PFS* technique provides a nearly optimal proxy placement, high delivery ratio, and low bandwidth overhead.

The *PFS* technique divides the network into cells that allow every two nodes in neighboring cells to communicate directly with each other. When an event is detected, it belongs to the cell, not to the node. Each cell has a coordinator node that is responsible for all actions within the cell. A unique *ID* is assigned to each cell in the network. A node recognizes its cell by using a *GPS* or an attack-resilient localization scheme. The sink is assumed to be in the center of the network, and each event has a cell *ID*, event type, and event time.

After proxies have been selected, they broadcast a "hello message" that includes Time-To-Live (*TTL*) that has the ability to reach all cells in the network. Next, each cell records the nearest proxy based on the received "hello message" and assigns the selected proxy as the default one for future communications. Then, every cell responds back to the selected proxy to inform the proxy that it is the one selected by the cell. Each cell creates a pairwise key with its proxy using one of the keying schemes. In addition, each proxy has a shared key with the sink. When a cell has a message to transmit, this message is encrypted using the pairwise key and sent after encryption to the proxy using a multi-hop routing protocol such as Greedy Perimeter Stateless Routing (*GPSR*). However, these messages follow the exponential distribution whether they are real or fake to avoid any *Time Correlation* attacks by an adversary. Therefore, if a cell observed a real event, the real event is delayed until the cell finds the appropriate time interval that does not violate the exponential distribution. When a proxy receives a fake packet, it discards the packet immediately. However, if the received packet is a real packet, the proxy re-encrypts the packet using the key shared between itself and the sink. Then, the proxy forwards the packet after delaying it in its buffer for an appropriate time. In case the proxy node did not receive a real packet for some time, it transmits a fake packet to the sink instead.

Note that a proxy node is able to recognize the difference between real and fake packets. Moreover, if a proxy receives a packet from another proxy in the network whether the packet is real or fake, the proxy forwards the packet to the next hop without filtration. A message routes through multiple proxies on its way to the sink; however, it is only being filtered in the original proxy. Eventually, optimizing the proxies location is essential to avoid undesirable traffic overhead. A disadvantage of this technique is that the sink must be in the center of the network. Another disadvantage is the filtration delay as the packet has to be filtered by a proxy before arriving at the sink.

3.1.7 Tree-based Filtering (TFS)

TFS [67] is an improved version of *PFS*. The difference between them is that *TFS* has several layers of filtration, the proxies nearest the sink filter fake packets coming from the proxies that are far from the sink, which leads to less overhead. In *TFS* each proxy has a parent proxy, and can have some child proxies using a tree concept to decrease the number of fake messages towards the sink. In contrast, more delay is required because the message is delayed in each and every proxy towards the sink. Therefore, the relationship can be described as a trade-off between overhead and delay while keeping high anonymity. Nevertheless, assuming the sink node is in the center of the network will limit the applications that can implement this technique. Using multiple proxies for filtration might also impact the performance of *TFS*.

3.1.8 Optimal-cluster-based Source Anonymity Protocol (OSAP)

Techniques like *ConstRate* and *FitProbRate* provide source anonymity but they are costly since they inject the network with a large number of fake messages. Further, all nodes in the network transmit traffic towards the sink that causes the network to be imbalanced as nodes nearest the sink consume more power than nodes far from the sink, which leads to a shorter lifetime of the network.

To overcome the high overhead in *ConstRate* and *FitProbRate*, and latency in *PFS* and *TFS*, *OSAP* [68] was developed. *OSAP* is based on *FitProbRate* and the authors of [68] argue that the use of unequally clustering mechanism will reduce the traffic overhead, improve the network balance, and decrease the latency of the real event. The unequally clustering mechanism is achieved by adjusting the transmission rate and the radius of unequal clusters. As a result, this mechanism fixes the overhead issue by transforming the issue into a mathematical programming problem that is solved by mathematical methods.

If a node has a real event to report, it becomes a source that generates some real packets, unlike *FitProbRate*, which allows all nodes to transmit packets whether they are real or fake. In addition, authors of *OSAP* assume that the sink is placed in the center of the network and works as the data collector for all events. Each packet has a source *ID*, event description, event time, and packet type (real or fake). Each node determines the number of hops to the sink by using the following formula: $(2n-1)\pi r^2\theta$ where n is the number of hops, r is the transmission range and θ is the network density of nodes distribution. After a node knows its hops count to the sink, the sensor network is divided into uneven clusters. Clusters, which are close to the center of the network are larger than the ones that have a further distance from the center, as shown in Figure 3.3. The sink node has the largest cluster and nodes at edges of the network have the smallest clusters. Nodes are categorized into cluster heads and cluster members. The purpose of the cluster head is to filter the incoming fake packets, and forward the real packets of the cluster members to the sink. The cluster head of the cluster with radius R_1 is the sink and all nodes within this cluster transmit their packets directly to the sink. All nodes with distance R_2, R_3, ...,R_n are cluster head candidates that have cluster radius of $R_2 - R_1$, $R3 - 2R_2 + R_1$, ..., $R_n - R_{n-1} - randi_{n-1}$, respectively. After cluster head nodes are selected, they broadcast BEACON packets with TTL, which includes the radius of the clusters they belong to. Member nodes select their cluster head based on the least communication cost, which is decided by the received BEACON packets. Then, each member node notifies its selected cluster head by a BEACON response. Therefore, the network will consist of rings and each ring consists of clusters. The distance between cluster head nodes and the sink is R_2, R_3, ...,R_n.

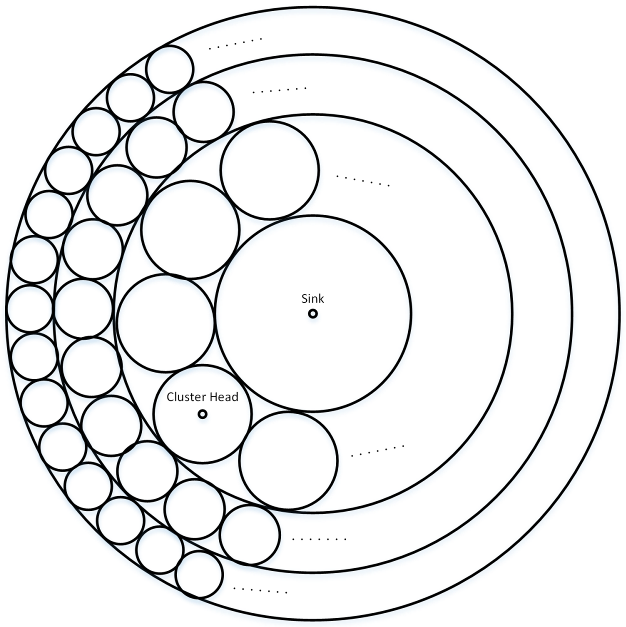


Figure 3.3. Overview of the unequally clustering mechanism in *OSAP*.

This technique assumes that each member node shares a pairwise key with its cluster head using a keying scheme. Each cluster head shares a key with its neighboring cluster head nodes. Once the member node detects an event, it sends the event to its cluster head using a multi-hop routing protocol. The member node delays the real packet to the next interval. Therefore, the adversary cannot distinguish the real packets from the fake ones using time analysis. Then, the cluster head decrypts the message and forwards it to the next cluster head encrypted by the shared key between them. In order to satisfy source anonymity, the transmitting time intervals follow the exponential distribution as exhibited by the *FitProbRate* technique. When a fake packet is received by the head cluster, it is discarded. In contrast, if the cluster head received a real packet, it re-encrypts and forwards the packet towards the sink after an appropriate time that follows the exponential distribution. However, if there is no real packet, the cluster head sends an encrypted fake packet instead. In case a cluster head received a packet from another cluster head, it forwards the packet after an appropriate time whether it is real or fake without filtration. The total delay of the real event must be less than the maximum required delay by the application. In addition, it is obvious that member nodes at the edge of the network have more cluster heads on the path to the sink, which increases the delay. Therefore, selecting the appropriate μ for each cluster is mandatory to balance the latency between clusters and to avoid any time analysis attacks from the adversary. Finally, balancing the power consumption between clusters is made by adjusting the radius of the unequal clusters and transmission rate.

OSAP still has some limitations; it assumes that the sink location is in the center of the network. This assumption is impractical for many applications that require the sink to be at different locations in the network. Another limitation is that the communications within the network are based on the same cluster heads that do not change during the lifetime of the network. Using the same cluster heads every time decreases the lifetime of the network because the cluster heads will consume more power than cluster members causing the network to be imbalanced regarding power consumption.

3.1.9 Recurrent Clustering Mechanism (RCM)

The reason current techniques have failed is that the nodes nearest the sink consume more power than the nodes far from the sink, *RCM* [69] authors argued. In order to balance the network, a clustering technique was introduced. All nodes use the *FIFO* queuing mechanism. Each cluster has a cluster head that coordinates the activities within the cluster. Each node has a timer, once the timer is equal to zero, the sensor node checks if it has a real packet in its buffer to transmit. In case there is no real packet, a fake packet is sent instead to the sink. Remaining energy of cluster heads is computed every time there is a packet to transmit. The higher cluster head in terms of the remaining energy is selected to forward the packet. In this technique, the sink location is known to the adversary. Since this technique uses clustering and selects the highest cluster node with remaining energy, *RCM* improves the power consumption by half compared to other techniques, the authors of [69] argued. Moreover, *RCM* reduces the overhead because of the clustering mechanism. However, the authors of [69] did not mention how they control the delay and delivery ratio. Overhead is still a concern because every time the node does not have a real packet, it generates a fake packet instead, which might lead to traffic overhead.

3.1.10 General Fake Source (GFS)

The *GFS* [55] technique attempts to simulate the movement of a real asset at different locations in the network to mislead the adversary about the actual location of the source node, as shown in Figure 3.4. This mechanism can be implemented easily if the *RFID* type is active. However, the goal of *GFS* is to simulate the movement of the asset using a passive *RFID*. *GFS* generates dummy traffic of a fake source. A shared token is used to determine which node should act as a fake source. Then, the fake source generates a fake event just after detecting the real asset. Next, the token is passed between nodes to simulate the movement of the real asset. In order to simulate a real asset, the number of intervals, which is represented by *simulateRound* that a fake source will transmit should vary from node to node, to mislead the adversary. Another variable is the *realCount*; it is increased by one whenever a node sends a real message. Once a node stops sending real messages, the *realCount* is reset to one, and *simulateRound* is updated.

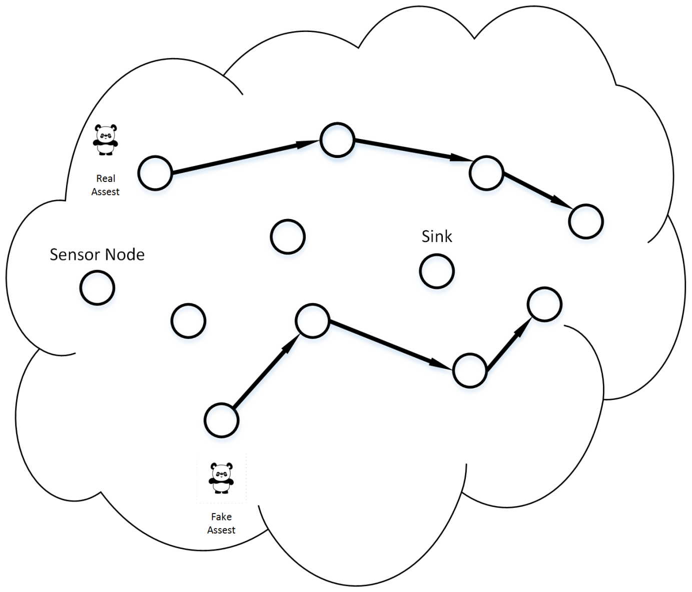


Figure 3.4. Overview of the real asset simulation in *GFS*.

The fake source is selected randomly and generates fake messages until *simulateRound* becomes zero; then the token is passed. However, to avoid passing the token between only two nodes, the last fake source is recorded in the *preNode* variable. When the fake source has a real message, the sink should be informed and the token is passed to the next node. Further, if there are real or fake messages without a token, the message can be passed normally to the sink. However, if the fake message has a token, the receiving node will get the *preNode* and *tokenID*. Then, the receiving node becomes the new fake source.

Since the token creates extra traffic that might be noticed by the adversary, each fake source needs to send a fake report to make the token message look like an ordinary real report. Eventually, real and fake messages have to be transmitted together in every interval to distract the adversary. However, *GFS* has some drawbacks: First, it assumes that the sink is placed in the center of the network. Second, if the token is passed among three nodes or more rather than between two nodes, there is no mechanism to handle this kind of situations. Third, *GFS* could fail to provide anonymity because only fake messages are created when a real event takes place. Therefore, the adversary might detect that a real event is being sent resulting in exposing the location and time of the event in case the adversary has enough capabilities and resources to examine all nodes at once.

3.1.11 Naive Algorithm (NAA)

Each node in *NAA* [70] broadcasts a fake message periodically. The time duration of these periods should be long enough to avoid draining the nodes' battery very quickly. If a node has a real message to transmit, it waits until the upcoming fake message is ready. However, instead of transmitting the fake packet, it is replaced by the real message. Once the real message is received by intermediate nodes, the process is repeated until the real message reaches its destination. The adversary cannot distinguish the difference between real and fake messages since they are sent using the same transmission rate. Nevertheless, the delay in this technique is high because it uses fixed long periods to transmit real messages.

3.1.12 Globally Optimal Algorithm (GOA)

The *GOA* [70] technique is an upgraded version of *NAA*. It was developed to decrease the delay of the real event. *GOA* provides each node with a timer that is defined by a pseudo-random number generator. This timer is utilized to allow the node to transmit real packets. If the time count reaches zero, and there is no real packet to transmit, a fake packet is transmitted instead. All nodes must use the same pseudo-random number generator and obtain the seed used by other nodes. This concept allows *GOA* to decide the shortest path to the sink. The *GOA* technique improves the throughput and power consumption when compared to *ConstRate*. However, the primary issue in *GOA* is that it should have knowledge about the entire topology of the network, which is not required by *ConstRate*.

3.1.13 Heuristic Greedy Algorithm (HGA)

The *HGA* [70] technique is very similar to *GOA*, but *HGA* only needs to obtain the seed and location of neighboring nodes allowing each node to select the best neighboring node to send or forward the packet.

3.1.14 Probabilistic Algorithm (PBA)

The *PBA* [70] technique is developed to reduce the overhead of dummy traffic in *GOA* and *HGA*. It fo1lows the same process as *HGA*. However, nodes do not have to transmit a fake packet every time the time count reaches zero in case there is no real packet to transmit. *PBA* uses a probability p to decide whether the node should send a fake packet or not; p is considered as a threshold, and used to trade-off between anonymity and overhead.

3.1.15 Distribution Resource Allocation Algorithm (DRAA)

The *DRAA* [71] technique uses the same concept as *ConstRate*. Each node in *DRAA* measures the best transmission rate for dummy traffic to reduce the overhead of the network. The main purpose of *DRAA* is to hide the real traffic within minimal dummy traffic. This mechanism can be achieved without applying the entire process of the original *ConstRate* technique. The *DRAA* technique provides *SLP* with reduced power consumption when compared to *ConstRate* according to *DRAA* developers.

3.1.16 Optimal Filtering Scheme (OFS)

The *OFS* [72] technique is based on *TFS* and *PFS* techniques. Each node in *OFS* has the possibility to become a proxy. This method can provide an optimal routing and filtering that eventually leads to optimal lifetime for the network. Once a sensor node is selected to be a proxy, it can use its filtration rules for further optimization to maximize the network lifetime. Moreover, a proxy has the ability to filter packets coming from other proxies, unlike *TFS*, where the proxy transmits all incoming packets from other proxies without any filtrations. In this protocol, proxies have two options: First, proxies can work individually, which increases the delay and provides a high level of anonymity. Second, they work together to decrease the delay with a low level of anonymity. This technique is considered as a trade-off between delay and anonymity. The challenge in *OFS* is to select the best locations for the proxies.

3.1.17 Aggregation-based Source Location Protection Scheme (ASLP)

The *ASLP* [73] technique uses similar filtration techniques as *OFS*. *ASLP* has three phases: fake packets, packet encryption, and data aggregation. In *ASLP*, the *WSN* is divided into clusters, and each cluster has a cluster head. Cluster heads follow a tree structure scheme to reach the sink. Every cluster member shares a key with its cluster head to encrypt the traffic between them. In addition, nodes communicate with their cluster head using the exponential distribution to reduce the delay and overhead. The exponential distribution is only controlled by one parameter λ, which is the transmission rate. Cluster nodes report the event to their cluster head, which also can be the sink node periodically; this period is decided by the value of τ. In the second phase, values of λ and τ are distributed throughout the network by the sink. In the third phase, the actual data aggregation and reporting take place. A node transmits a real packet whenever it detects a real event to the sink using the exponential distribution. Otherwise, the node transmits a fake packet instead. The cluster heads receive the packet from cluster members and forward it to the sink using an encrypted channel. Values of λ and τ are utilized in this technique to trade-off between latency and power consumption. This trade-off can be adjusted according to the requirements of the applications.

3.1.18 Trusted Computing Enabled Heterogeneous (TCH-WSN)

The network in *TCH-WSN* [74] is also divided into clusters. Each cluster contains a high-performance node with two modules: Trusted Platform Module (*TMP*) and Mobile Trusted Module (*MTM*). These two modules are used to provide data security and integrity. The sink and cluster heads are assumed to have continuous power supply. The sink communicates with all nodes directly. However, a sensor node communicates with the sink only through its cluster head. The cluster head assigns one of the nodes in its cluster to act as a fake source for a specific duration of time. Then, the fake source transmits dummy traffic to the sink.

3.1.19 Efficient Privacy Preservation (TESP)

The *TESP* [75] technique uses cluster heads to filter fake packets that are being transmitted to the sink. *TESP* consists of three phases: The first phase, the sink provides public and private keys to each node in the network. The keying mechanism contributed by the sink is based on elliptic curve cryptography, which is preferred over the traditional asymmetric key algorithms. In the second phase, nodes are deployed, organized, and assigned to clusters. All clusters have a cluster head that connects to all other cluster heads as well as to the sink in a tree structure manner. The sink is placed at the root of the tree. Neighboring nodes have a symmetric key between them to communicate securely. In the final phase, each sensor node checks its buffer for real packets. If there is no real packet, the node generates a fake packet encrypted by the public key of the cluster head. However, if the node has a real packet in its buffer, this packet is encrypted by the sink's public key. When a packet reaches the cluster head, if the cluster head is able to decrypt the packet successfully, that means it is a fake packet. Therefore, the packet is dropped immediately. Each cluster head collects all packets from its cluster members. Then, the cluster head waits for a higher cluster head or the sink to ignite the data collection signal. Once the cluster head receives the signal, it sends the collected data after re-encryption to the higher cluster head or the sink. In case the higher cluster head received the data from a lower cluster head, it adds its collected data to the incoming data. This process is repeated until all collected data reaches the sink.

3.1.20 Cloud-Based Scheme for Protecting Source-Location Privacy (CSPSLP)

In *CSPSLP* [43], the authors assume that multiple sensor nodes detect the asset, and these nodes attempt to inform the sink about the location of the asset simultaneously, which creates a traffic hot-spot. Normally, the adversary seeks hot-spot areas to discover the source node. The main aim of *CSPSLP* is to hide the actual source inside a cloud that consists of multiple nodes. Therefore, the real hot-spot is hidden within a larger cloud. *CSPSLP* consists of three stages: In the first stage, each node is assigned a unique *ID*, a secret pairwise key with its neighboring nodes, and a shared key with the sink. These *ID* and keys are used by nodes to build pseudonyms that are very similar to the hidden identities utilized in Anonymous Path Routing *APR* [76, 77].

The second stage is executed at the deployment of the nodes, and it is called the bootstrapping stage. In this stage, each node sends its location to the sink to obtain the shortest path between itself and the sink. The following step is to select some of the nodes randomly that are at most *h* hops away from the sensor node itself. These selected nodes become fake sources, and they exchange data with each other to create pseudonyms among them. Each node separates its neighboring nodes into different groups. All created groups are placed in opposite directions from each other. Further, every node shares a secret key with each group.

The third stage is the event transmission. When a source detects a real event, it decides which node will act as a fake source. Furthermore, the source node broadcasts the packet to one of its groups that was already created. However, the chosen group must have one member on the path towards the selected fake source. The transmitted message contains the actual event encrypted by the pairwise shared key between the source and the sink. Moreover, the pseudonym is shared among the real and fake sources as well as among intermediate nodes, fake sources, real sources, and fake sinks. An intermediate node adds the packet to its buffer only when its pseudonym is in the packet. Otherwise, the intermediate node generates a fake packet with a *TLL* counter. This packet is forwarded until the *TLL* counter reaches zero. When the pseudonym of a node is in the packet, the node moves the pseudonym to the next intermediate node and forwards the packet to the group that the node belongs to. All members of the group receive the packet; if the member's pseudonym is not in the packet, the member generates a fake packet. Nevertheless, if the packet has the pseudonym of the member, the member node adds the packet to its buffer and repeats the process until the packet reaches the fake source. Once the fake source receives the packet, it forwards the packet to the sink using the same process as *APR*. However, *CSPSLP* has only one difference from the original *APR*; *CSPSLP* re-encrypts the packet that has the event information between all intermediate nodes using the shared key between itself and the sink. When the sink receives the packet, it searches for the pseudonym of the source and the fake source to select the appropriate key to decrypt the packet. Finally, dummy traffic that belongs to the same cloud is filtered by the intermediate nodes. For example, if one of the intermediate nodes had many fake packets from the same cloud, the intermediate node has the right to use a filtration mechanism. This filtration is utilized to only forward one of the fake packets to reduce overhead.

3.1.21 Dummy Wake-up Scheme (DWUS)

The key concept of *DWUS* [78] is to create multiple dummy traffic streams to direct the adversary away from the actual location of the source node. These streams have to be toward the sink. *DWUS* coordinates the dummy traffic streams to act similar to real traffic streams. This technique consists of three phases: First, the *WSN* is divided into different groups of dummy populations. Each group has a group leader that is changed periodically. This group leader is responsible for selecting the fake sources. The second phase is called the Wake-up, each group leader hires a nearby fake source and sends a wake-up message to the selected fake source. In the third phase, once the fake source receives the wake-up message, it sends a fake message towards the sink using one of the selected intermediate nodes. Phase two and three are repeated using a fixed transmission rate to simulate the existence of the real asset at different locations, which subsequently confuses the adversary about the original location of the real asset.

3.1.22 Group Algorithm for Fake-traffic Generation (GAFG)

The fundamental notion of *GAFG* [79] is very similar to *DWUS*. Every node in the network transmits its packets to the sink following a predefined path. Forwarding nodes have a higher transmission rate than source nodes. Once a source detects a real event, *GAFG* transmits the event to the sink using the exponential distribution. Then, *GAFG* attempts to create fake data reports that have a very close μ to the real data report. Moreover, this technique ensures that many nodes in the *WSN* will transmit fake data at specific times according to the real event exponential distribution. Nevertheless, *GAFG* is vulnerable to *Rate Monitoring* attacks since forwarding nodes have higher transmission rates than source nodes.

3.1.23 Source Simulation (SoSi)

This technique [80] assumes that the global adversary can only trace the presence of a moving asset. The adversary tracks every trace in the network whether it is real or fake. Moreover, each trace is assumed to be a candidate of a real trace. The main aim is to create several traces to mislead the adversary about the presence of the real event. Additionally, *SoSi* has its own definition for privacy. Privacy occurs when the adversary observes a set of transmissions, which indicates that the asset is nearby one of the nodes. Virtual assets are created to copy the behavior of the real asset to confuse the adversary. Another purpose of these virtual assets is to create dummy traces. In order to implement the virtual assets, some sensor nodes are selected randomly. Each selected node obtains a token in the deployment stage. These nodes are called the token nodes. *SoSi* forces the *WSN* to function in rounds; each round has a fixed time. In each round, a fake event is transmitted to the sink by the token node through its neighboring nodes, which create a stream of traffic. By the end of each round, the token node selects the next node that will act as a token node. The *SoSi* technique does not increase the delay since the dummy streams do not affect the real event. However, the use of dummy streams increases overhead.

3.1.24 Source and Destination Seclusion using Clouds (SECLOUD)

The *SECLOUD* [81] technique has three steps: First step, each node transmits a “hello message” to all nodes in the network using flooding routing protocol. The “hello message” includes a *TTL* counter in trying to find the nearby nodes within a specific range h. These nodes create a cloud; if a node of a cloud desires to transmit, it selects some of the cloud members to act as pseudo-sources. The basic idea of pseudo-sources is to generate fake packets using the same transmission rate as the real source. All fake packets will be dropped by the receivers. The sink has its cloud and uses the same procedure of ordinary nodes. In the second step, *SECLOUD* provides the source node with multiple options. The first option is to communicate with the sink using single path routing algorithm. The other option is to use delegate sources and delegate sinks. In the second option, the source node assigns some of its cloud members to be delegate sources, and the sink assigns some of its cloud members to be delegate sinks. Therefore, packets can travel between the source and sink through delegate nodes. In the last step, the fake sources and fake sinks that are created in the first step can be utilized to transmit dummy traffic between them. This step is essential to hide the real traffic, which confuses adversaries about the location of the real event. However, the utilized mechanism to create fake clouds is not explained in details. Moreover, the use of the cloud concept might increase the power consumption and overhead of the system.

3.1.25 Unobservable Handoff Trajectory (UHT)

The *UHT* [82] technique has a unique assumption about the adversary. This assumption is that the asset can enter and move to a random point within the network. Nodes have information about the poisson distribution of assets entering the *WSN*. The traveled distance by assets is based on the uniform distribution. In addition, all nodes use the same shared key to encrypt and decrypt messages among them. In this solution, the edge nodes are called the perimeter nodes because they are the first nodes to detect the asset in the network. The perimeter nodes transmit fake messages in case there are no real events detected by them. A fake message should have a *length* variable that behaves similarly to *TTL*. The next step is to *XOR* the fake message with the *ID* of the next node that will send the fake message. The fake message is routed to the sink through intermediate nodes. All nodes use a broadcast technique to transmit their messages. Once a node receives a message, it attempts to *XOR* the message with its *ID* to determine whether the node should generate a fake message or not. If a readable message is the output of the *XOR* operation, the *length* variable is decreased. In case the *length* variable is not zero, the node sends a fake message after *XORing* it with the next node *ID*. In contrast, if the *length* variable is zero, the message is *XOR*-ed with the current node *ID*. Then, the node transmits the fake message to the sink through intermediate nodes. By sending messages from fake events, the adversary becomes completely confused about the candidate event and whether it is real or fake.

3.1.26 Pollution Avoiding Source Location Privacy (PA-SLP)

In this technique [63], messages are transmitted in specific time slots to avoid *Time Correlation* attacks. In addition, *PA-SLP* implements random network coding on the transmitted packets to prevent *Size/Structure Correlation* attacks. In *PA-SLP*, there are three different types of packets: real, fake, and polluted (packets created or modified by an adversary). Therefore, a mechanism called Triple-Type Homomorphic Signature (*TTHS*) was developed to filter the unwanted packets such as the fake and polluted ones. Moreover, a signature equation with a secret key is used to determine the type of the forwarded packets without exposing their contents. If the packet type is real, it will be forwarded to the next node towards its destination. However, if the packet is fake, it will be discarded to reduce the overhead of the network. Lastly, in case the packet is polluted, it will be utilized to improve the Intrusion Detection Systems (*IDS*) by locating the captured nodes. The entire process is employed by intermediate nodes to enhance the overall performance of the network.

3.2 Classification of Fake Packet Techniques

This section provides a classification of fake packet techniques regarding the use of: *Intervals*, *Timers*, *Random Numbers*, *Probability*, *Clusters*, *Proxies*, *Constant Distribution*, *Exponential Distribution*, and *Poisson Distribution.* The classification is presented in Table 3.1. *Intervals* indicate that the lifetime of the network is divided into time durations. These durations can be equal or unequal based on the implemented distribution. *Timers* are utilized as a trigger for a node to transmit either real or fake packets. *Random Numbers* decide the probability of a node to transmit a packet. *Probability* decides whether a node will transmit a packet or not. *Clusters* indicate that the network is divided into groups of nodes (Figure 3.5). Each group has multiple member nodes and one head node. *Proxies* are commonly used to filter fake packets towards the sink. *Constant Distribution* breaks down the time into equal intervals, whereas *Exponential Distribution* decides the next interval according to this equation: $X = \log \frac{(1-u)}{-\lambda}$, where X is the next interval, u is the uniform random number, and λ is the transmission rate. *Poisson Distribution* calculates its intervals

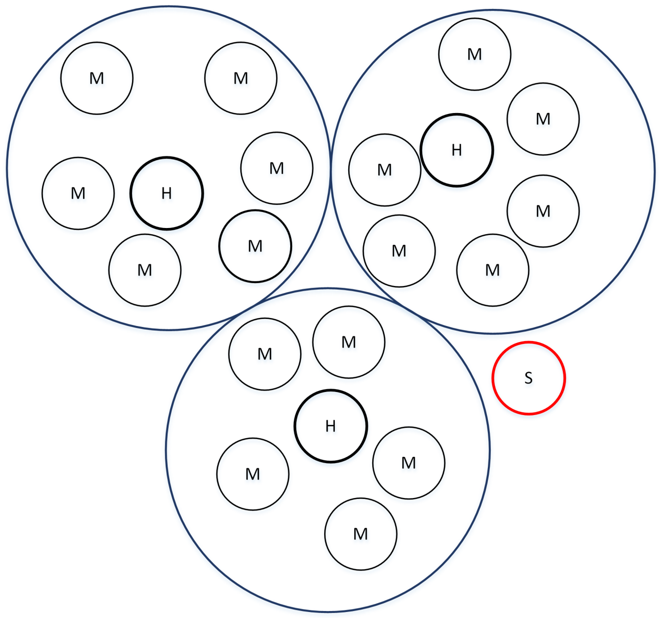


Figure 3.5. The clustering mechanism. H is a cluster head. M is a cluster member. S is the sink.

distribution based on the following equation: $P(x, \lambda) = \frac{\lambda^x e^{-\lambda}}{x!}$, where $P(x, \lambda)$ is the probability, x takes a whole number, and λ is the average number of events per interval. All techniques are assumed to implement similar structure, format and size for both real and fake packets. Using different size or format for fake packets makes real packets easily detectable if an adversary applies a *Size/Structure Correlation* analysis.

Table 3.1. Classification of fake packet techniques.

Technique	**Intervals**	**Timers**	**Random Numbers**	**Probability**	**Clusters**	**Proxies**	**Distribution Type**
PeCo [66]	•	•	-	-	-	-	-
ConstRate [54]	•	-	-	-	-	-	Uniform
ProbRate [54]	•	-	-	-	-	-	Exponential
FitProbRate [54]	•	-	•	•	-	-	Exponential
Baseline [67]	•	-	-	•	-	-	Uniform Exponential
PFS [67]	•	-	-	-	•	•	Exponential
TFS [67]	•	-	-	-	•	•	Exponential
OSAP [68]	-	-	-	-	•	-	Exponential
RCM [69]	-	•	-	-	•	-	-
GFS [55]	•	•	-	-	•	-	-
NAA [70]	•	-	-	-	-	-	-
GOA [70]	•	•	•	-	-	-	-
HGA [70]	•	•	•	-	-	-	-
PBA [70]	•	•	•	•	-	-	-
DRAA [71]	•	-	-	•	•	-	Uniform
OFS [72]	•	-	-	-	-	•	-
ASLP [73]	•	-	•	-	•	•	Exponential
TCH-WSN [74]	-	-	-	-	•	-	-
TESP [75]	-	-	-	-	•	•	-
CSPSLP [43]	-	-	-	-	•	-	-
DWUS [78]	•	-	-	-	•	-	Uniform
GAFG [79]	•	-	-	•	•	-	Uniform Exponential Poisson
SoSi [80]	•	-	-	-	-	-	-
SECLOUD [81]	-	-	-	-	•	-	-
UHT [82]	•	-	-	-	-	-	Poisson
PA-SLP [63]	•	-	-	•	-	•	Uniform

• Used.
- Not used.

3.3 Global Adversary Models

This section provides a detailed description of the used global adversary models in fake packet techniques, and how assumptions are made in each one of them.

The *PeCo* technique assumes that the adversary is able to deploy its own sensor network to monitor and analyze a *WSN*. Furthermore, the adversary network consists of a smaller number of nodes than the targeted network as the adversary only eavesdrops on the radio signals of legitimate nodes. Additionally, the adversary does not sense the environment like the authorized nodes. The adversary is also equipped with *GPS* to detect the location of communications precisely [66].

The adversary is considered as *external*, *global* and *passive* in *ConstRat*, *ProbRate* and *FitProbRate* techniques. External means that the adversary cannot control or compromise a sensor node physically. Global means that the adversary has a full view of network communications as well as sufficient resources and unlimited power. Passive has three different aspects: First, an attacker cannot expose the content of a real event message that could lead to the source *ID*. Second, in the situation where messages are encrypted the same way during the forwarding process, the attacker has the capability to trace back the origin. Third, the adversary can apply complicated traffic analyses such as *Rate Monitoring* and *Time Correlation*. In *Rate Monitoring* attacks, the adversary focuses on the difference of the transmission rates between nodes, especially those nodes with higher rates. Nevertheless, the *Time Correlation* attack works on the diversity of transmission times between transmitting packets. It is also assumed that the adversary has enough resources to apply all of these advanced attacks [54].

Baseline, *PFS*, *TFS* and *OSAP* techniques have the same assumptions for the adversary, which is *external*, *global*, and *passive* [67]. However, *OSAP* provides more assumptions: First, the adversary knows the location of all nodes in the network. Second, it knows the distribution type of the *WSN*. Third, the adversary has the capability to compare its time interval observations with the known distribution type. Additionally, the adversary is assumed to be unable to disclose the content of packets or identify whether the packet type is real or fake [68].

In *RCM*, the adversary cannot decrypt communications of the network. Therefore, packets appear completely random from the adversary perspective. Moreover, the adversary is aware of the sink location [69].

The *GFS* technique builds its own technique based on the following adversary model assumptions. The adversary deploys its network to overhear the radio transmissions among legitimate nodes. Adversary nodes have unlimited processing power and battery life. The adversary can only eavesdrop on the traffic, but it is unable to alter the traffic or compromise the sensor nodes. Data is encrypted and the adversary cannot gain any meaningful information about the packets' content. Further, real and fake packets are identical in size and structure making the adversary unable to distinguish the difference between them. Lastly, the adversary has knowledge about the sink location, the network topology, and the implemented routing protocol [55].

NAA, *GOA*, *HGA* and *PBA* techniques utilize a similar adversary model. This model assumes that the attacker knows the location of all sensor nodes in the targeted network. Moreover, the attacker has the ability to snoop on the traffic of the entire network. The adversary has enough resources to keep all collected data for further offline analyses. However, the attacker is unable to break the encrypted packets or compromise sensor nodes physically [70].

DRAA, *OFS*, *ASLP* and *TESP* techniques assume the popular *external*, *global* and *passive* adversary model [71–73, 75]. However, the adversary in *TCH-WSN* can monitor the entire network traffic [74].

The *CSPSLP* adversary model deploys its monitoring devices in the targeted areas within the network. The adversary collects information from multiple areas, but not from the entire network. These areas are called the observation points, and their location is adjustable to be as close as possible to the real asset. This model has three different characteristics: passive, well-equipped, and informed. Passive means that the adversary does not modify the network transmissions; it only observes transmissions among nodes. Well-equipped means that each attacking device can measure the angle of arrival and strength of the signal to determine the source location. However, the adversary is unable to spot the receiving nodes because all nodes within the range of the transmitting node will receive the signal. Lastly, the adversary is informed; it knows the sink location and is able to monitor the sink node traffic as well [43].

DWUS assumes that the adversary is passive and has unlimited resources. Further, the adversary can distribute attackers throughout the network to sense all transmitted packets [78].

In the *GAFG* technique, the adversary is aware of the sink location and network topology. The adversary is passive and can detect the time and location of all transmissions. Moreover, it has the ability to perform sophisticated statistical methods for detection. However, the adversary cannot break the encryption of packets [79].

The adversary model in *SoSi* [80] is considered to be fast and effective. It has two possibilities: First, the adversary uses a large number of devices to monitor the entire network. Second, the adversary can deploy a smaller number of devices that have more capabilities and resources. However, authors of [80] argue that the second option is impractical because of the high cost. They also assume that the adversary can sense the actual asset instead of overhearing the traffic among authorized nodes. In this technique, the adversary uses *GPS* or one of the localization techniques to determine roughly the occurrence area of the real event.

In *SECLOUD*, the adversary capabilities are a combination of different adversary models presented in [66, 83, 84]. In this model, the attacker knows the network topology and retains the measurements of the entire network traffic including *Rate Monitoring* and *Time Correlation*. In addition, the attacker has the ability to visualize the network traffic and regulate the network links density. However, the adversary is passive and cannot compromise network nodes. The attacker has its own network that consists of several malicious nodes. These nodes collaborate together using a different frequency band to transmit the collected data to a centralized malicious entity [78].

The *UHT* technique follows the external, global, and passive adversary model. However, *UHT* has further assumptions. In cases where the adversary knows the location of the sink, it will snoop on all communications within the network. The adversary can attack communications among intermediate nodes in a parallel manner. After collecting data, the adversary checks the content of packets to gain information about the source *ID*. However, in a situation where packets are well-encrypted, there are two possibilities: First, if packets remain the same without re-encryption when they are forwarded, the adversary can trace them back to the origin. Second, if packets are decrypted and encrypted every time before forwarding, the adversary will apply complex analysis methods such as *Rate Monitoring* and *Time Correlation* [82].

The *PA-SLP* adversary model is assumed to be both external and internal. Moreover, it can perform active and passive attacks. The adversary has the ability to eavesdrop on the entire network. In addition, it can apply *Time Correlation* and

Size/Structure Correlation attacks. The adversary might capture intermediate nodes and disclose the secret keys. Exposed keys are utilized to pollute the traffic by inserting new packets into the network or altering the existing packets [63].

3.4 Classification of Global Adversary Models

This section provides a classification of the used global adversary models (Tables 3.2 and 3.3) in the discussed fake packet techniques regarding the usage and knowledge of the following assumptions: *Topology Information*, *Sink Location*, *Localization*, *Interval Distribution*, *Rate Monitoring*, *Time Correlation*, *Visualization*, *Machine Learning*, and *Statistical Methods*. *Topology Information* means that the adversary knows the location of all nodes in the network and how they are connected to each other. *Sink Location* means that the adversary is able to determine the location of the sink node. *Localization* means that the adversary can use *GPS* or one of the localization techniques. *Interval Distribution* means that the adversary knows the distribution type of packets. *Rate Monitoring* means that the adversary can differentiate between transmission rates of network nodes. If a node has a higher transmission rate than other nodes, this node is easily detectable, e.g., the 0.4 transmission rate some nodes use in Figure 3.6.

Time Correlation means that the adversary can distinguish the difference of transmission times between packets. For instance, in Figure 3.7, interval G does not seem that it follows the same distribution of the other intervals. Therefore, if network nodes transmit packets using uniform distribution, and one of these nodes decided to transmit packets without using the uniform distribution, the adversary can easily differentiate between these packets. *Visualization* means that the adversary has the ability to convert the sending/not-sending behavior of the network nodes into a binary image. This image can be analyzed to gain meaningful information about the real event. *Machine Learning* means that the adversary can employ a classifier such as the neural network to analyze the traffic of the *WSN*. Finally, *Statistical Methods* mean that the adversary is able to perform a sophisticated statistical analysis.

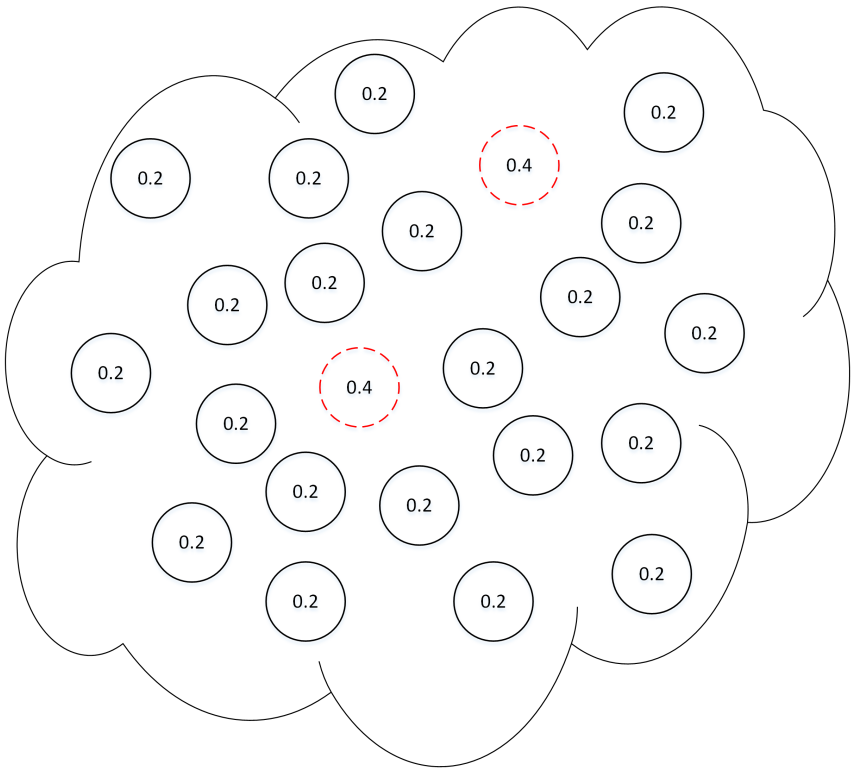


Figure 3.6. Rate monitoring. Solid circles are nodes with transmission rate of 0.2 per second. Dashed circles are nodes with transmission rate of 0.4 per second.

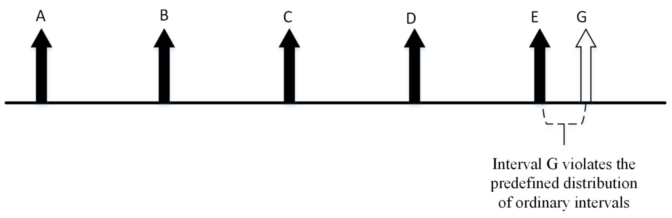

A
B
C
D
E
G
Interval G violates the predefined distribution of ordinary intervals

Figure 3.7. An example of time correlation.

Table 3.2. Classification of global adversary models (1).

Technique	**Topology Information**	**Sink Location**	**Localization**	**Interval Distribution**	**Rate Monitoring**
PeCo [66]	not known	not known	used	not known	not used
ConstRate [54]	not known	not known	not used	known	used
ProbRate [54]	not known	not known	not used	known	used
FitProbRate [54]	not known	not known	not used	known	used
Baseline [67]	not known	not known	not used	not known	used
PFS [67]	not known	not known	not used	not known	used
TFS [67]	not known	not known	not used	not known	used
OSAP [68]	known	not known	not used	known	used
RCM [69]	not known	known	not used	not known	not used
GFS [55]	known (including the used routing algorithm)	known	not used	known	used
NAA [70]	known	known	not used	not known	used
GOA [70]	known	known	not used	not known	used
HGA [70]	known	known	not used	not known	used
PBA [70]	known	known	not used	not known	used
DRAA [71]	not known	not known	not used	known	used
OFS [72]	not known	not known	not used	not known	used
ASLP [73]	not known	not known	not used	known	used
TCH-WSN [74]	known	not known	not used	not known	not used
TESP [75]	not known	not known	not used	not known	used
CSPSLP [43]	known (location of sending nodes only)	known	used	not known	used
DWUS [78]	not known	known	not used	known	not used
GAFG [79]	known	known	used	known	not used
SoSi [80]	not known (routing algorithm is known)	not known	used	known	used
SECLOUD [81]	known	not known	not used	not known	used
UHT [82]	not known	known	not used	known	used
PA-SLP [63]	not known	not known	not used	not known	used

Table 3.3. Classification of global adversary models (2).

Technique	**Time Correlation**	**Visualization**	**Machine Learning**	**Statistical Methods**
PeCo [66]	not used	not used	not used	not used
ConstRate [54]	used	not used	not used	used
ProbRate [54]	used	not used	not used	used
FitProbRate [54]	used	not used	not used	used
Baseline [67]	used	not used	not used	not used
PFS [67]	used	not used	not used	not used
TFS [67]	used	not used	not used	not used
OSAP [68]	used	not used	not used	not used
RCM [69]	not used	not used	not used	not used
GFS [55]	used	not used	not used	not used
NAA [70]	used	not used	not used	not used
GOA [70]	used	not used	not used	not used
HGA [70]	used	not used	not used	not used
PBA [70]	used	not used	not used	not used
DRAA [71]	used	not used	not used	not used
OFS [72]	used	not used	not used	not used
ASLP [73]	used	not used	not used	not used
TCH-WSN [74]	not used	not used	not used	not used
TESP [39]	used	not used	not used	not used
CSPSLP [43]	not used	not used	not used	not used
DWUS [78]	not used	not used	not used	not used
GAFG [79]	used	not used	not used	used
SoSi [80]	not used	not used	not used	not used
SECLOUD [81]	used	used	not used	not used
UHT [82]	used	not used	not used	not used
PA-SLP [63]	used	not used	not used	used

CHAPTER 4: SYSTEM MODELS AND PROPOSED TECHNIQUES

4.1 System Models of the Proposed Techniques

In this section, we present the network model, routing model, adversary model, anonymity model, and Anderson-Darling model of the proposed techniques.

4.1.1 The Network Model

A number of sensor nodes are distributed in the area of interest. Distribution of nodes can be random or in fixed locations. All sensors have the same resources such as memory, processing power, and battery life. A sensor node collects information about the asset within its sensing area, and transmits this information to the sink that has more resources than ordinary sensor nodes. The sink node can be placed at any location in the network such as at one side of the network or in the center of the network. Sensing area for each node can be calculated as follows:

$$R_{area} = \pi r^2 \tag{4.1}$$

Where R_{area} is the transmission area and r is the transmission range. In the case of forwarding packets from one node to another, all nodes satisfy Equation (4.2) are considered as neighboring nodes of the transmitting node.

$$(x_n - x_{source})^2 + (y_n - y_{source})^2 < r^2 \tag{4.2}$$

Where x_{source} and y_{source} are coordinates of the source node. x_n and y_n are coordinates of the receiving node. Sensor nodes can locate their position by using one of the localization techniques in the deployment stage. There are many localization techniques in the literature [3, 85–87] used by *WSNs* to provide each node in the network with its current location as well as neighboring nodes' locations. Many of the localization techniques provided in the literature can work just fine as an additive module to our framework. Each application has its specific requirements such as the lifetime of the network and the maximum delay. The lifetime of the *WSN* is divided into time intervals. Each interval can further include many sub-intervals. This mechanism can be adjusted to meet the application requirements. All real and fake packets are assumed to be encrypted using a shared key between senders and receivers. Therefore, the message payload is secure. Real and fake packets are identical in terms of size and structure to avoid *Size/Structure Correlation* attacks by an adversary. When a receiving node receives a packet, it is able to differentiate between real and fake packets by decrypting them using the shared key.

4.1.2 The Routing Model

The routing protocol, in our proposed model, is based on the location of sensor nodes. It is also called geographical routing [88]. Therefore, each node should know its coordinates and its neighboring nodes' coordinates as well. The routing protocol selects the next node on the path towards the sink based on the following equation:

$$d = \sqrt{(x_c - x_{sink})^2 + (y_c - y_{sink})^2} \tag{4.3}$$

Where x_{sink} and y_{sink} are the coordinates of the sink, x_c and y_c are the coordinates of the candidate node, and d is the destination between the sink and candidate node. The node with the smallest d value will be the next hop on the path. The developed location-based routing protocol is flexible. This flexibility means that if the selected candidate node is out of battery or has physical damage, the routing protocol will choose the second best candidate node that satisfies the minimum d. The entire process will be repeated until the packet is received by the sink. This routing protocol does not cause high power consumption. It is employed to reduce the total number of packets throughout the network, which leads to less overhead. Flooding-based routing is not utilized in this work because it surges the number of overhead packets and causes high power consumption.

4.1.3 The Adversary Model

In this work, the employed global adversary is passive, external, and global that is similar to the model proposed in [54,68]. Passive adversary means that the observer can analyze and collect packets. An external adversary cannot compromise a sensor node physically, whereas a global adversary has a full view of the network as well as sufficient resources and unlimited power. The adversary has malicious nodes that are deployed to monitor the entire network and generate a high level of statistical analysis. The capabilities of the adversary are further extended such that the adversary can create a dataset of many observed intervals for each sensor node during the lifetime of the network. The adversary uses this dataset to analyze the *SLP*. For example, a neural network can be trained on this dataset in trying to expose the existence of the real event. The adversary also has the ability to visualize the dataset by converting it into a binary image, and extracting any suspicious patterns that could point to the existence of the real event. Lastly, the adversary can employ a steganography method to measure the binary relative entropy and uncertainty of the system.

4.1.4 The Anonymity Model

The anonymity of the proposed system can be broken into three main parts: (1) existence of the real event, (2) location of the real event, and (3) time of the real event. In order to satisfy all three parts, the existence of the real event must be unknown to the adversaries. Then, location and time can be subsequently achieved.

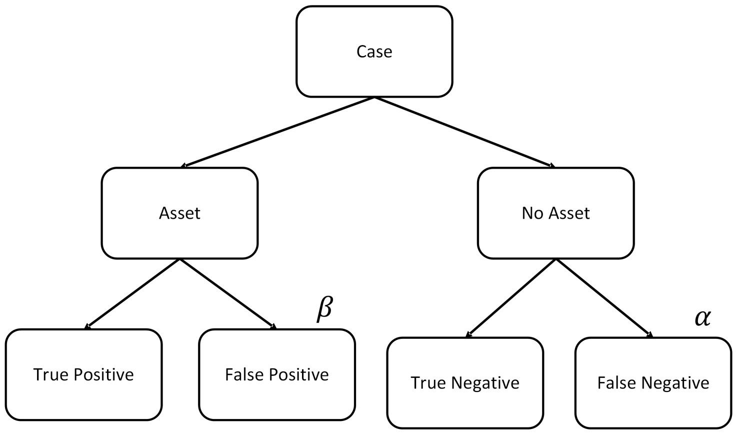


Figure 4.1. α is the false negative and β is the false positive.

The existence of the real event can be represented as follows:

$$f(event) = \begin{cases} 1, & \text{a real event exists.} \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

The anonymity of the system can be calculated by the following steganography equation [89, 90]:

$$d(\alpha, \beta) = \alpha \log_2 \frac{\alpha}{1-\beta} + (1-\alpha) \log_2 \frac{1-\alpha}{\beta} \tag{4.5}$$

Where α is the probability of an adversary to falsely detect the real event. α is the false negative of the system, which means the possibility of the adversary to say there is an asset, and actually, there is no asset (Figure 4.1). In contrast, β is the probability of the adversary not to detect the presence of the real event. β is the false positive of the system, which means the possibility of the adversary to say there is no asset, and actually, there is an asset (Figure 4.1). Therefore, in order to achieve anonymity, the system should satisfy:

$$d(\alpha, \beta) \leq \varepsilon \tag{4.6}$$

Where $d(\alpha, \beta)$ is the binary relative entropy, and considered as the anonymity provided by a specific technique, whereas ε is the anonymity required by the application. The smaller ε is, the higher probability an adversary will fail to detect the real event correctly.

4.1.5 The Anderson-Darling Test Model

The A-D test [91–96] is a statistical analysis algorithm. The A-D test is a goodness of fit test. It is used to determine if a series of data follows a specific distribution. For example, if there is a sequence of samples that follows an exponential distribution, the output of the A-D test should be either null or alternative hypotheses. Null (H_0) means the data follows an exponential distribution and alternative (H_1) means the data does not follow an exponential distribution.

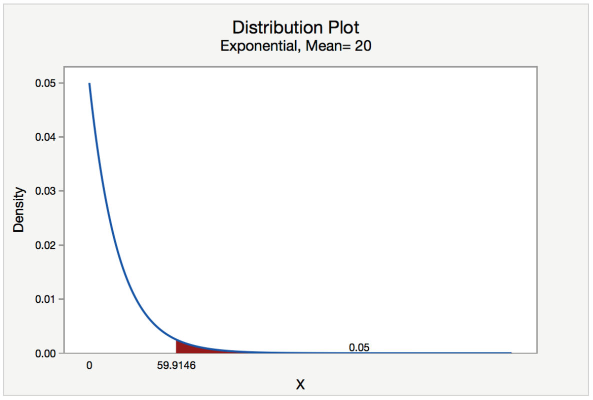


Figure 4.2. The CDF of the exponential distribution.

In Figure 4.2, if the p-value that is represented as x falls in the red area under the curve, this indicates H_0 can be rejected, which means the data is not exponentially distributed. On the other hand, if the p-value falls in the white area under the curve, that indicates H_0 cannot be rejected, which means the data is exponentially distributed.

In Figure 4.3, random exponential data is examined by the A-D test for normality and exponentiality to illustrate the difference. The blue dots are the examined

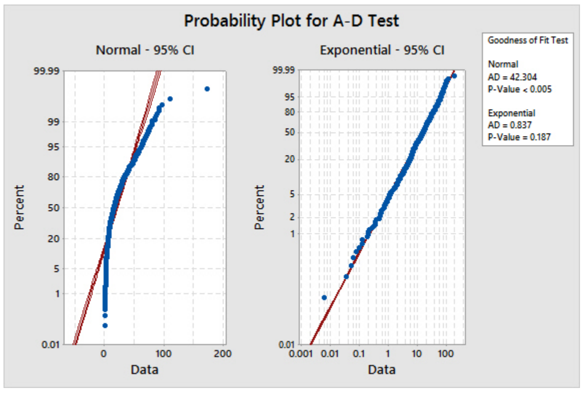


Figure 4.3. Normality and exponentiality test for samples that follow the exponential distribution.

samples and the red lines are the expected values. On the right side of Figure 4.3, distributed samples follow the expected values indicating the samples are distributed exponentially. On the left side of Figure 4.3, it shows that the distributed samples do not follow the expected values indicating the samples are not normally distributed.

The A-D test works as follows: First, A^2 is the mathematical notation of the A-D test, and it can be calculated by the following equations:

$$A^2 = -N - S \tag{4.7}$$

$$S = \sum_{i=1}^{N} \frac{2i-1}{N}[\ln F(Y_i) + \ln(1 - F(Y_{N+1-i})] \tag{4.8}$$

Where N is the total number of samples, i is the sample ID, Y_i is the samples ordered in an ascending way, Y_{N+1-i} is the samples ordered in a descending way, F is the Cumulative Distribution Function (CDF) of the data. A^2 is compared to the corresponding critical value of the significance level α. The critical value of $\alpha = 0.05$ is 1.321 in the exponentiality test, which means that the output of the A-D test should be less than this number (threshold) for the data to pass the test. In case the number of samples is small, the following equation is utilized to increase the accuracy of the test.

$$A^* = A^2(1.0 + \frac{0.6}{\sqrt{n}}) \tag{4.9}$$

The complexity of *A-D* test is $O(n \log n)$ since it uses Quicksort to order its elements. The *A-D* test is employed by several algorithms to analyze the data in order to determine the characteristics of the samples' distribution.

4.2 Proposed Techniques for Source Anonymity against Global Adversary

In this work, six techniques are developed to provide source anonymity and overcome the statistical analysis of a global adversary. These techniques are Dummy Uniform Distribution (*DUD*), Dummy Adaptive Distribution (*DAD*), Controlled Dummy Adaptive Distribution (*CAD*), Exponential Dummy Adaptive Distribution (*EDAD*), Exponential Dummy Adaptive Distribution Plus One (*EDADP1*), and Exponential Dummy Adaptive Distribution Plus Two (*EDADP2*). All proposed techniques are based on injecting the network with dummy traffic to confuse the adversary about the existence of the real event. Therefore, reducing the number of fake packets while keeping a high level of source node anonymity is essential.

The notion of all techniques is to divide the lifetime of the network into intervals. Hence, if an asset is detected by a node and a series of real packets needs to be transmitted, instead of transmitting the real packet immediately after the occurrence of an event, the packet will be transmitted at the end of the interval. This mechanism is necessary to avoid the *Time Correlation* attack. All other nodes in the network send fake packets at the end of the interval based on probability if they do not have any real packets. When a sensor node receives a fake packet, the node will discard it right away. However, if the packet is real, the node will add the packet to its buffer and try to forward it in the upcoming interval based on probability. In case the real packet is not transmitted in the upcoming interval, the real packet waits for one more interval, and the node will attempt to forward it once again based on probability.

This process means that each node in the network has its private pattern for sending real packets, which look entirely random to the observer. Since real and fake packets are sent only at the end of the interval, they should be both indistinguishable from the adversary perspective.

4.2.1 Dummy Uniform Distribution (DUD)

Injecting a fake packet in every interval for each node during the lifetime of the network is going to consume a significant amount of power and resources. The motivation behind DUD is to have the same transmission rate for both real and dummy traffic. DUD works as follows: Each node will throw a random number num_{random} between 0 and 1. If the random number is smaller than the predefined/constant transmission rate, e.g., $num_{random} < rate_{constant}$, send the real packet, but if there is no real packet in the node's buffer, it will send a fake packet instead. By applying this mechanism, the adversary cannot recognize if the transmitted packet is real or fake. When a node detects a real event, it uses the selected transmission rate, e.g., 0.1, in trying to transmit the first real packet in the upcoming interval. For instance, if a real event was detected between interval 5 and 6, the node will throw a random number between 0 and 1. If this number is less than the selected threshold (transmission rate), which is 0.1 in this case, the real packet will be transmitted in interval 6. However, if the random number is larger than the threshold, the real packet will not be transmitted, and the node will try to send it in the following interval (interval 7). This process is repeated until all real packets are sent.

Fake packets are only generated if a node does not have a real event. In addition, fake packets are also sent based on probability. If the random number is less than the threshold and there are no real packets, a fake packet is generated and transmitted instead. In contrast, if the random number was larger than the threshold, the node will not transmit any packets. However, this scheme does not always guarantee the arrival of the real event's packets because real packets can take a long time to be delivered especially if there is a required maximum delay by the application. It is obvious that increasing the transmission rate reduces the delay and increases the overhead, or vice versa.

4.2.2 Dummy Adaptive Distribution (DAD)

To reduce the delay and increase the delivery ratio of received real packets to the total number of real packets, the *DAD* technique is introduced and works as follows: All nodes in the network are categorized into fake nodes and real nodes. At the beginning, all nodes will be considered as fake nodes using a predefined/constant transmission rate as presented in *DUD*. Fake nodes do not generate fake packets in every interval. They only generate and transmit fake packets if the thrown random number between 0 and 1 is less than the selected threshold. However, if a node detects a real event or forwards packets of a real event, then it becomes a real node. A real node will increase the transmission rate of its real traffic by a specific value (Equation (4.10)). Then, the real node decreases the transmission rate of its dummy traffic by the same specific value (Equation (4.11)) after all real packets are transmitted. This reduction in transmission rate occurs after the interval at which the last real packet is transmitted. The increasing and decreasing percentages of the real and fake transmission rates are based on the original transmission rate. Real and fake transmission rates of the real nodes are given by the following equations:

$$R_{real} = R_{const} + \frac{N_{real}}{I_{total}} \tag{4.10}$$

$$R_{fake} = R_{const} - \frac{N_{real}}{I_{total}} \tag{4.11}$$

$$R_{const} = \frac{R_{real} + R_{fake}}{2} \tag{4.12}$$

Where R_{real} and R_{fake} are the real and fake transmission rates. R_{const} is the predefined transmission rate of the network, N_{real} is the number of real packets, and I_{total} is the number of total intervals.

Since the average of real traffic and dummy traffic is equal to the original constant transmission rate (Equation (4.12)), the adversary will not notice any change in the transmission rate of the network, which is essential to avoid *Rate Monitoring* attacks. Additionally, in order to perform well, *DAD* needs to satisfy the following equation:

$$N_{real} \leq R_{fake} * I_{total} \tag{4.13}$$

Otherwise, the number of fake packets will be smaller than the number of real packets throughout the network. This situation allows the adversary to easily detect the real event. *DAD* keeps the same level of anonymity and overhead as *DUD*, but it

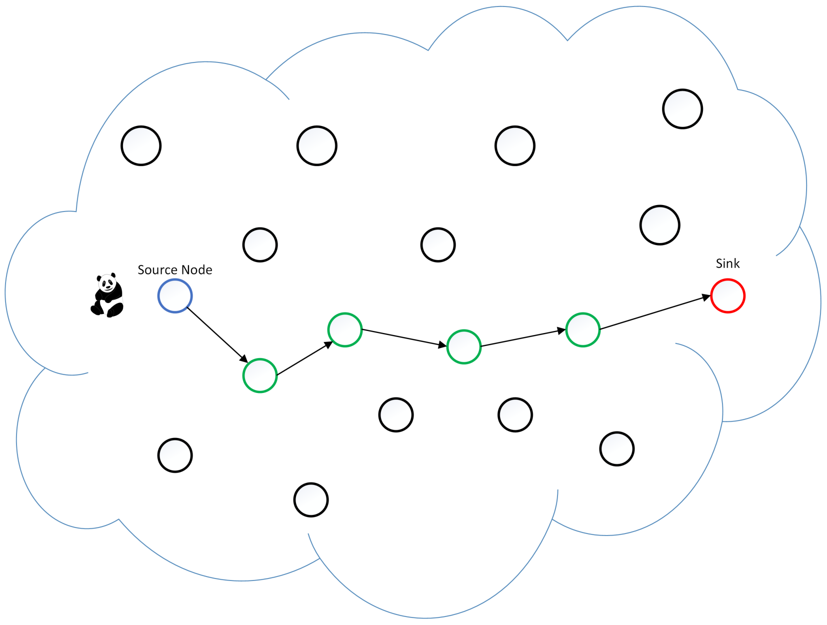


Figure 4.4. An example of the *DAD* technique.

increases the delivery ratio and reduces the average delay of the real event. However, *DAD* is still unable to guarantee the maximum delay of real packets. An example of *DAD* is shown in Figure 4.4. The transmission rate of green nodes is only modified using Equations (4.10) and (4.11) when they have a real event to report. The black nodes will keep their original transmission rate without any change because they only generate fake packets.

4.2.3 Controlled Dummy Adaptive Distribution (CAD)

Since *DUD* and *DAD* schemes could fail in delivering packets of a real event within a specific delay, *CAD* is introduced to maximize the delivery ratio and minimize the delay to guarantee the arrival of all packets in the real event to the sink within the required constraints. Based on *DAD*, *CAD* increases the real traffic transmission rate and decreases the dummy traffic transmission rate using the same Equations (4.10), (4.11) and (4.12). However, if a real node fails to transmit a real packet using the real traffic transmission rate for n-intervals, this node will send the first real packet in its buffer without using any probability (transmission rate is equal to one). Then, the node reuses the original real traffic transmission rate for the following real packet. By repeating this process, all real packets can be delivered within guaranteed n-intervals as presented in Algorithm 4.1. This technique is a trade-off between delay and anonymity. If n-intervals number is large, it means that there will be more delay and a higher level of anonymity. If n-intervals number is small, that means less delay and a lower level of anonymity. Therefore, adjusting the number of n-intervals is based on the application requirements and the level of tolerance.

Algorithm 4.1 *CAD*

```
1: $constantRate$ ← select the desired transmission rate
2: $realRate$ ← apply Equation (4.10)
3: $fakeRate$ ← apply Equation (4.11)
4: $realIntervalCount$ ← the current number of intervals a real packet waited
5: $maxIntervalCount$ ← the maximum number of intervals desired by the application
6: $randomSending$ ← a random number between 0 and 1
7: for $slot_i$ < number of lifetime slots do
8:    for each node in the network do
9:       if node has a real packet in its buffer then
10:         if $slot_i$ == sending interval then
11:            if maxIntervalCount == realIntervalCount then
12:               make realRate equal to 1
13:            else
14:               make the transmission rate equal to realRate
15:            end if
16:            if randomSending $\leq$ realRate then
17:               remove the real packet from node's buffer
18:               adjuest the sending time for other packtes in buffer
19:               add partition to time slot
20:               run the routing protocol desired
21:               send the real packet
22:            else
23:               delay all packets in buffer and try next interval
24:            end if
25:         else
26:            if randomSending $\leq$ fakeRate then
27:               send a fake packet
28:            end if
29:         end if
30:      end if
31:   end for
32: end for
```

4.2.4 Exponential Dummy Adaptive Distribution (EDAD)

To reduce the overhead of a *WSN* without sacrificing the anonymity of the source node, *EDAD* is introduced. The exponential distribution has only one parameter λ, which represents the transmission rate. Having one parameter helps to fix the transmission rates flow of all nodes in the network. Therefore, the adversary is unable to distinguish the difference between real and fake events. However, each node still has its own sending interval pattern that is based on probability. The next sending interval for real and fake packets can be obtained from the following equation:

$$p = e^{-\lambda t} \tag{4.14}$$

Where p is the probability, λ is the transmission rate, and t is the next sending interval. In order to make the exponential distribution equation predict the next sending interval, the equation needs to be rearranged. By taking the ln of both sides, the equation will be as follows:

$$\ln p = \ln e^{-\lambda t}$$

$$\ln p = -\lambda t$$

$$t = \frac{\ln p}{-\lambda} \tag{4.15}$$

Each node in the network implements Equation (4.15) to transmit both real and fake packets. If there is no real event, nodes will keep sending fake packets. Once a node detects a real event, it starts to transmit the real packets. The next scheduled fake packet is replaced by the real one. This mechanism is necessary not to violate the exponential distribution sequence. Otherwise, the real event can be easily detected if an adversary applies *Time Correlation* attacks. Since all nodes use the same value of λ, and each node has a different sending interval pattern, the adversary will be confused about the existence of the real event.

4.2.5 Exponential Dummy Adaptive Distribution Plus One (EDADP1)

EDADP1 is based on the original *EDAD*. However, *EDADP1* gives the priority to real packets over the fake ones by increasing the transmission rate of real packets within a certain percentage. This percentage cannot be extremely high; otherwise, the real event will be easily detected by the adversary. Since the exponential distribution is a statistical solution, the result should have multiple outcomes that satisfy the analytical models. Therefore, the adversary should not be able to distinguish the change of the transmission rate if the increase did not exceed a specific range, e.g., 30%. For fake packets, *EDADP1* uses the original transmission rate to reduce the overhead. If a real packet is detected, *EDADP1* increases the transmission rate by a certain percentage to reduce the delay of the real event without sacrificing the anonymity of the system. The decision of the appropriate threshold (maximum transmission rate increase) relies on the original transmission rate and the number of real packets. Algorithm 4.2 shows how *EDADP1* works.

Algorithm 4.2 *EDADP1*

```
1: $numberOfScenarios \leftarrow$ number of the scenarios to generate
2: $numberOfObservations \leftarrow$ number of the samples in the sequance
3: $numberOfRealPackets \leftarrow$ number of real packets
4: $rate \leftarrow$ the original transmission rate
5: $increasedRatePercentage \leftarrow$ the increased transmission rate in percentage
6: select the desired $numberOfObservations$
7: for $scenario_i < numberOfScenarios$ do
8:     if there are real packets then
9:         for $realPacket_i < numberOfRealPackets$ do
10:            increase the transmission rate by $increasedRatePercentage$
11:        end for
12:    else
13:        use the original transmission rate
14:    end if
15: end for
```

4.2.6 Exponential Dummy Adaptive Distribution Plus Two (EDADP2)

EDADP2 is an alternative of *EDADP1*. *EDADP2* increases the transmission rate of only the odd real packets, and for the even real packets, *EDADP2* uses the same transmission rate as the fake packets. This variety provides better anonymity for the system but in return, the delay will be higher for the real event. *EDADP2* needs slightly more processing time since it filters which packets are odd and which packets are even. This increase in processing time can be considered as a drawback when compared to *EDADP1*. Algorithm 4.3 shows how *EDADP2* works.

Algorithm 4.3 *EDADP2*

```
1: $numberOfScenarios \leftarrow$ number of the scenarios to generate
2: $numberOfObservations \leftarrow$ number of the samples in the sequance
3: $numberOfRealPackets \leftarrow$ number of real packets
4: $rate \leftarrow$ the original transmission rate
5: $increasedRatePercentage \leftarrow$ the increased transmission rate in percentage
6: select the desired $numberOfObservations$
7: for $scenario_i < numberOfScenarios$ do
8:    if there are real packets then
9:       for $realPacket_i < numberOfRealPackets$ do
10:         if $realPacket_i$ is odd then
11:            increase the transmission rate by $increasedRatePercentage$
12:         else
13:            use the original transmission rate
14:         end if
15:      end for
16:   else
17:      use the original transmission rate
18:   end if
19: end for
```

4.3 Metrics of the Proposed Techniques

The following four metrics affect the performance of the proposed techniques: transmission rate, number of real packets, number of total intervals, and maximum delay a real packet can wait before transmission, which is only applicable to the *CAD* technique. Using a higher transmission rate will increase the probability of delivering all real packets within the application required delay. In addition, higher transmission rates enhance the anonymity of the entire network and decrease the probability of an adversary to detect the real event. However, higher transmission rates will cause more dummy traffic leading to unnecessary overhead. The real event consists of several real packets; the larger number of real packets, the more difficult it is to hide inside the dummy traffic. Subsequently, more dummy traffic is required leading to a network overhead.

The total number of intervals and the maximum delay required by an application plays an imperative role in real packets delivery ratio. The total number of intervals is calculated using the following equation:

$$I_{total} = \frac{t_s}{I_r} \tag{4.16}$$

Where t_s is the simulation time, and I_r is the interval rate. More intervals increase the probability of real packets to arrive at the sink, which improves the overall performance of the network. However, more intervals require more fake packets that will consume resources and power. In *CAD* technique, the n-intervals number represents how many intervals a node will wait before transmitting the real packet without using probability. More n-intervals mean higher anonymity and delay, whereas smaller n-intervals mean less anonymity and delay.

Selecting the suitable values of these metrics will rely on the requirements of the application. The trade-off between delay and overhead versus privacy must be taken into consideration.

4.4 The Proposed Specialized Event-Driven Network Simulator for Security and Anonymity Applications of Wireless Sensor Networks

This work presents a novel software architecture for a specialized network simulator that is targeted towards analysis and verification of anonymity algorithms for *WSNs*. Even though many different network simulators exist and are popular such as *NS-2*, none of these can be adapted easily for testing of advanced *WSN* anonymity algorithms. For example, in trying to implement anonymity, fake messages need to be generated. A strong anonymity technique has to randomize the generation of fake messages and improve the ratio of real to fake messages so that excessive power is not being wasted by a sensor node. In order to analyze the effectiveness of different anonymity algorithms, an event-driven network simulator is developed that provides statistical and visualization features. This simulator can be easily configured to any *WSN* topology and routing protocol. The software architecture of the simulator allows for easy pluggability of different algorithms making it a valuable tool in *WSN* security/anonymity deployment and research.

Developing a comprehensive *WSN* simulator is vital for most of the applications so that engineers, developers, and researchers can examine their algorithms and network policies to verify the functionality and efficiency before implementing them in real applications. There are many WSN simulators available on the market such as NS-2. Each one of them has its advantages and disadvantages [97]. Nevertheless, one of the primary features lacking in prevailing *WSN* simulators is the support of direct high-level real/fake packet injections, which is very crucial for anonymity applications. Such applications require the capability of injecting dummy traffic into the network to mislead the adversary about the location and time of a real event.

Since this feature is unavailable in a straightforward manner in current simulators, a new simulator is introduced to overcome this problem by creating a suitable and comfortable environment for developers who work in the security and anonymity field.

4.4.1 Background of Network Simulators

There is a variety of *WSN* simulators for the developers to choose from such as *NS-2*, *OPNET*, and *OMNeT++*. In this section, a brief review regarding each one is presented.

NS-2 stands for network simulator version 2. It is an object-oriented discrete event-driven network simulator developed in *C++* and *OTcl* programming languages. *NS-2* provides many models such as sensor channel and power models that are easy to manipulate. *NS-2* is used to simulate both wired and wireless networks. It has some built-in protocols that are readily available such as LEACH and Directed Diffusion [98]. However, if a developer desires to design a customized protocol or routing technique, numerous modifications need to be made to *NS-2* system files to incorporate the created *C++* files with the *OTcl* setup. This process can be quite complicated and not easy to achieve. Further, *NS-2* lacks accommodating variations in specialized built-in protocols for *WSN*, which can be inconvenient for developers [99]. Another drawback is the complicated setup structure of *NS-2* that increases the difficulty level of debugging. Moreover, it contains bugs such as unreliability and simulation validation. *NS-2* consumes a significant memory, and the speed of the simulator is very slow, especially when simulating large networks [100].

OPNET is a *WSN* simulator that is based on an object-oriented design. *OPNET* is popular and considered as a commercial modeling and simulation tool. It uses a fast discrete event simulation engine that interacts with a parallel simulation kernel. *OPNET* does not provide many built-in protocols, and even the built-in protocols are hard to adjust. Therefore, protocols and modules need to be developed from scratch [101].

OMNet++ is an object-oriented paradigm network simulator, and it is also a discrete simulation framework. *OMNet++* can run on Windows, Linux, and Mac OS. This simulation supports various structures for *WSN* such as the *mobility framework*, *MiXiM*, *Castalia*, *INET*, and *NesCT* [102, 103]. *OMNet++* is not a *WSN* simulator itself. It needs to be bundled with other models to simulate *WSNs*, which can be complicated [104].

4.4.2 The Proposed Network Simulator Architecture

When designing a reliable simulator, timing is crucial since it is the principal character of any network simulator. A timing wheel concept is introduced, and it works as follows: The timing wheel is employed to simulate the time, and it is broken into many timing slots that are considered as the lifetime of the network. Each time slot (interval) can denote a time duration such as 1 ns or 1 ms. Moreover, each time slot consists of several partitions that represent concurrent activities, and

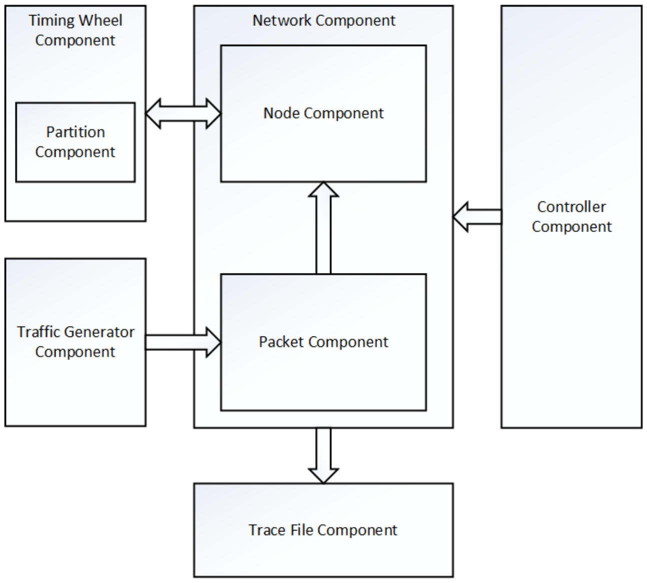


Figure 4.5. The proposed network simulator architecture.

thus allow the simulator to manage several packets simultaneously. The architecture of the network simulator contains many components, which collaborate with each other providing an accurate and trusted outcome. The components are *Packet*, *Node*, *Network*, *Partition*, *Timing Wheel*, *Trace File*, *Traffic Generator*, and *Controller*, as shown in Figure 4.5. All of the components are developed using C#.

4.4.2.1 The Packet Component

A packet is a component that needs to be passed between nodes until it arrives its destination. Each packet has a unique *ID* to distinguish itself from other packets in the network during the simulation time. In addition, a packet must keep the source node *ID* and the destination node *ID* to find its way throughout the network. Another value that a packet has to retain is the starting transmitting interval, which allows the simulator to calculate the packet delay. In some of the applications, a packet type is necessary. In this kind of applications, some packets can be real, and other can be fake to confuse the adversary about, e.g., the actual location of the event. Therefore, adding the capability of having different types of packets is essential to help the developer accommodate customized implementations. Lastly, the packet also contains the actual data, known as the payload. An overview of the packet component is shown in Figure 4.6.

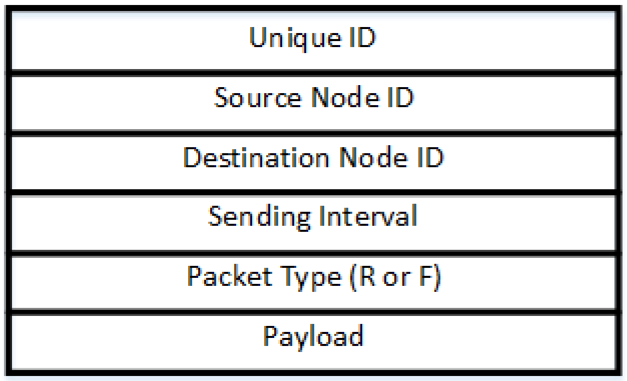


Figure 4.6. The packet component.

4.4.2.2 The Node Component

A node component is utilized to transmit, forward, or receive packets. For instance, a source node transmits a sequence of packets to the sink node to notify it about the current temperature. Each node should obtain a unique *ID* to have clear communications between nodes. Some of the existing applications use location-based routing, which requires the coordinates of nodes in order to select the shortest path towards the destination. Therefore, a node should be capable of obtaining and maintaining its current position, and this is accomplished in the deployment stage. Node transmission range is another specialty included in the architecture, which can be customized by the developers to fit their assumptions and requirements. Furthermore, nodes have the capability to calculate the hop count between themselves and the sink node as well as keeping a record of all neighboring nodes in the case of using a different kind of routing protocol such as flooding. Additionally, a node can keep track of all previous packets that were transmitted or forwarded during the lifetime of the network. This feature is useful in the case where the developers want to use a smart flooding (each node sends or forwards a packet only once even if the packet is received multiple times by the same node) to avoid undesirable overhead. All of these features in the node component provide the proposed architecture with more flexibility and practicality. This allows developing new anonymity algorithms

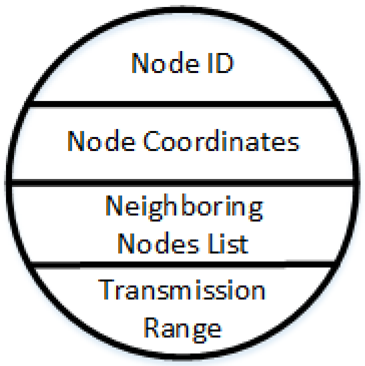


Figure 4.7. The node component.

in an easier way and straightforward manner. An overview of the node component is shown in Figure 4.7.

4.4.2.3 The Network Component

The network component consists of nodes that are connecting to each other creating a functional *WSN*. This component allows the developer to select source and sink nodes of the event as well as the transmitting time. Commonly, the network covers an area of interest such as a vast forest or a battlefield. Hence, the network has two parameters to decide: dimensions of x-axis and y-axis. Each network has its own lifetime based on the application requirements and topology failures. A scenario might be deploying the nodes randomly in a particular area of interest or having specific locations for each node in the network. All of these alternatives can be configured by the developer based on the requirements of the application. The network component requires each node in the network to provide a list of its neighboring nodes during the deployment stage.

4.4.2.4 The Timing Wheel Component

The timing wheel is the primary component of the system since it manages the simulation time and determines which nodes are transmitting and which nodes are receiving. The timing wheel is divided into time slots based on the lifetime of the simulation. Furthermore, each slot is broken into partitions (Figure 4.8). A slot represents the sending, receiving, or forwarding time, and the partition represents the sending, receiving, or forwarding node. Once the simulation commences, a method is invoked that has a nested loop. The first loop iterates through the time slots, and the second loop iterates through the partitions inside each slot. Moreover, the routing protocol is a part of the timing wheel component. The routing protocol decides which time slot will have which partition and action type (sending, receiving, or forwarding). Algorithm 4.4 shows how the timing wheel component works. Some activities such as sending or forwarding will cause new activities like receiving. The new activity needs to be added to the timing wheel, as shown in Figure 4.8.

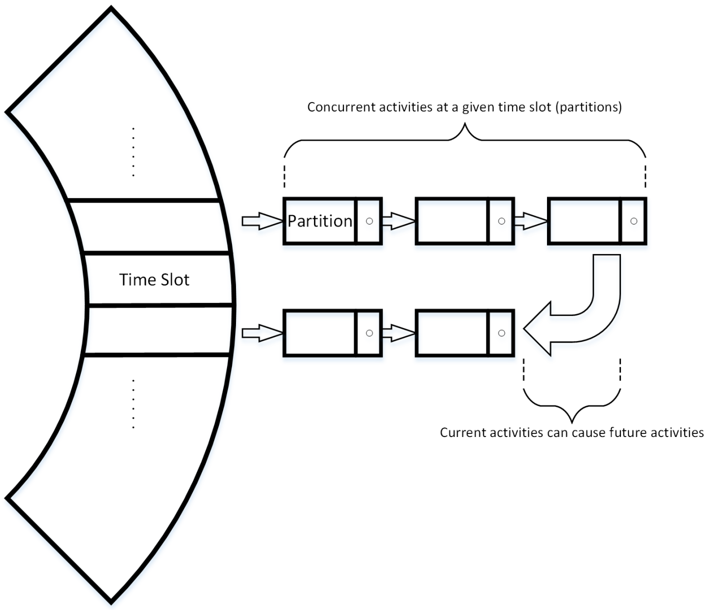

Concurrent activities at a given time slot (partitions)
Partition
Time Slot
Current activities can cause future activities

Figure 4.8. The timing wheel component.

Algorithm 4.4 Timing Wheel

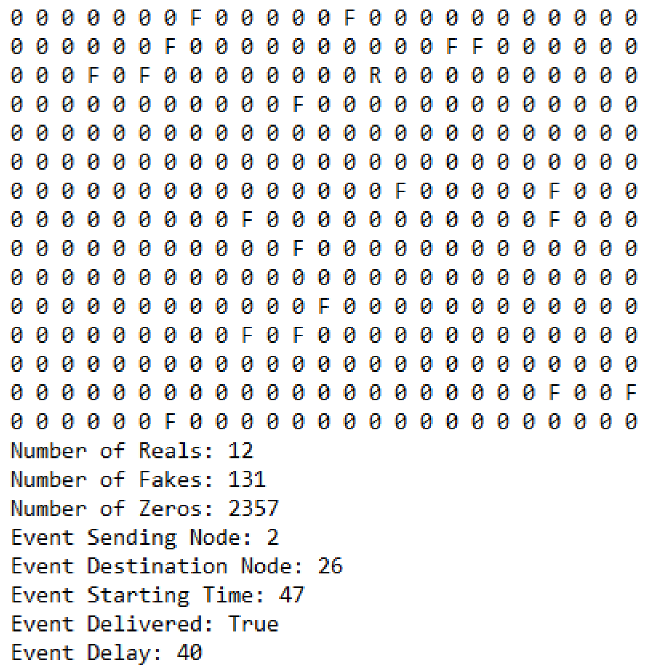
```
1:  for $slot_i$ < number of lifetime slots do
2:    for each node in the network do
3:      if node has a packet in its buffer then
4:        if $slot_i$ == sending interval then
5:          - remove packet from node's buffer
6:          - adjuest the sending time for other packtes
7:           in buffer
8:          - add partition to time slot
9:          - run the desired routing protocol
10:       end if
11:     end if
12:   end for
13: end for
```

4.4.2.5 The Partition Component

Each time slot in the timing wheel may contain several partition components. A partition points to a node object, which represents the sending or receiving node. Additionally, a partition contains the type of the packet such as real or fake. It might further have an action such as sending, receiving or forwarding. Conclusively, the most significant aspect of the partition component is its ability to possess a pointer to the following partition that eventually generates a linked list of different events inside every time slot in the timing wheel. This mechanism allows the network to deal with multiple packets concurrently.

4.4.2.6 The Trace File Component

The trace file component is essential since it contains the results and statistics of the simulation. Three files are generated: The first one has the number of packets sent and dropped, packets type, which nodes are transmitting/forwarding and which are not, the sender node, the destination node, event starting time, and event delay. All information is combined into one file to make it easier for the developer to trace the implementation (Figure 4.9). The second trace file has the statistics of all nodes such as node *ID*, node coordinates, hop count to the sink, and the list of all neighboring nodes (Figure 4.10). The last trace file includes the number of the cases in which



```
0 0 0 0 0 0 0 F 0 0 0 0 0 F 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 F 0 0 0 0 0 0 0 0 0 0 F F 0 0 0 0 0 0
0 0 0 F 0 F 0 0 0 0 0 0 0 0 R 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 F 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 F 0 0 0 0 0 F 0 0 0
0 0 0 0 0 0 0 0 0 F 0 0 0 0 0 0 0 0 0 0 0 F 0 0 0
0 0 0 0 0 0 0 0 0 0 0 F 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 F 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 F 0 F 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 F 0 0 F
0 0 0 0 0 0 F 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Number of Reals: 12
Number of Fakes: 131
Number of Zeros: 2357
Event Sending Node: 2
Event Destination Node: 26
Event Starting Time: 47
Event Delivered: True
Event Delay: 40
```

Figure 4.9. The network trace file (1).

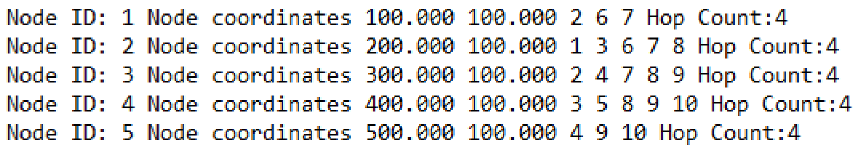

```
Node ID: 1 Node coordinates 100.000 100.000 2 6 7 Hop Count:4
Node ID: 2 Node coordinates 200.000 100.000 1 3 6 7 8 Hop Count:4
Node ID: 3 Node coordinates 300.000 100.000 2 4 7 8 9 Hop Count:4
Node ID: 4 Node coordinates 400.000 100.000 3 5 8 9 10 Hop Count:4
Node ID: 5 Node coordinates 500.000 100.000 4 9 10 Hop Count:4
```

Figure 4.10. The network trace file (2).

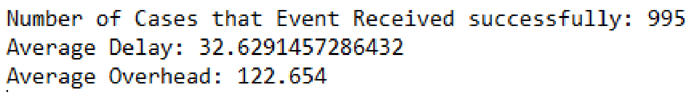

```
Number of Cases that Event Received successfully: 995
Average Delay: 32.6291457286432
Average Overhead: 122.654
```

Figure 4.11. The network trace file (3).

the event was successfully received, average delay, and average overhead for all cases (Figure 4.11). Each case represents a complete scenario that has a different location and starting time for the event.

4.4.2.7 The Traffic Generator Component

When simulating or testing a particular technique, a traffic generator is required. This component is responsible for generating network traffic. The proposed architecture can create random events including random events' locations and times. The generated traffic can also be controlled and specified for a particular location and time. The adaptability of the architecture enables developers to have a variety of selections to choose from. Algorithm 4.5 shows how the traffic generator component works.

Algorithm 4.5 Traffic Generator

```
1: $sendingNode \leftarrow$ source node with event to send
2: $receivingNode \leftarrow$ sink node
3: $startingInterval \leftarrow$ event starting time
4: for each node in network do
5:     if node id $==$ $sendingNode$ then
6:         for $i <$ number of packets required do
7:             - create a packet
8:             - add the packet to source node's buffer
9:         end for
10:    end if
11: end for
```

4.4.2.8 The Controller Component

This portion of the simulator architecture can be viewed as the brain of the design since it brings all components together. This component decides which one of the other architecture components should be created and which one should wait. The controller focuses on producing different objects of the components, and how they interact with each other to construct a scenario that satisfies the developer's demands. Additionally, this component is responsible for providing the interface that will be controlled by the user to select the proper options for the desired simulation scenario.

CHAPTER 5: IMPLEMENTATION AND TEST PLAN

5.1 Simulation Test Plan for DUD, DAD, CAD, and EDAD

In the simulation, the *WSN* covers an area of interest: 600 m by 600 m. The network consists of 25 sensor nodes and the lifetime of the simulation is 100 intervals. These nodes monitor the movement of an asset such as a panda. The network only has one sink on the right side of the network, as shown in Figure 5.1. If a node detects a panda, it starts communicating with its neighbors to inform the sink about the current location of the panda. All sensor nodes have a transmission range of 200 m. The network is tested at different transmission rates: 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, and 0.4. Then, if the random number between 0 and 1 is less than the selected transmission rate, and a real event is received, a real packet will be sent; otherwise, a fake packet will be sent instead. The number of packets in the real event is selected

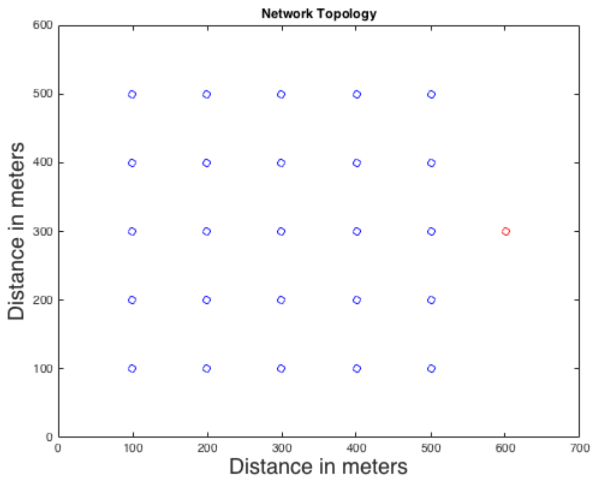


Figure 5.1. Network topology. Blue nodes are conventional sensors. The red node is the sink.

to be 3. If the random number exceeds the transmission rate, no packets will be sent. For example, if the transmission rate is 0.05, $rate_{real}$ and $rate_{fake}$ will be 0.08 and 0.02, respectively by using Equations (4.10) and (4.11) for DAD and CAD techniques. Finally, the maximum number of intervals to wait before mandatory transmission in CAD is selected to be 5 (the maximum wait before the node is forced to transmit the real packet).

The panda can only be detected by one node in the network. One thousand random cases are created for each transmission rate to evaluate the performance of the *WSN*. Each case has a random position for the panda and a random starting interval for the event between 0 and 49. A comparison between different transmission rates and how they affect the average delay, delivery ratio, and overhead of the real event is conducted. All of the proposed techniques were developed using C#.

5.2 Anonymity Test Plan for DUD, DAD, CAD, and EDAD

There are many ways to validate that the proposed techniques are increasing the delivery ratio and reducing the average delay as well as the overhead without sacrificing the anonymity of the source node. In this work, three different approaches are developed: visualize the output data of the simulation, feed the output data of the simulation to a trained neural network, and apply Equation (4.5) for anonymity testing as mentioned in the anonymity model described in Chapter 4. In the visualization and neural network models, output data is converted into a binary matrix. When a sensor node transmits a packet whether it is real or fake, the transmission is represented by a binary value of 1, whereas if the node does not transmit, it is represented by a binary value of 0. Since some of the intervals in the simulation will have values of 1 and others will have values of 0, the output of the simulation can be represented as a binary matrix. Analytical models were developed using MATLAB.

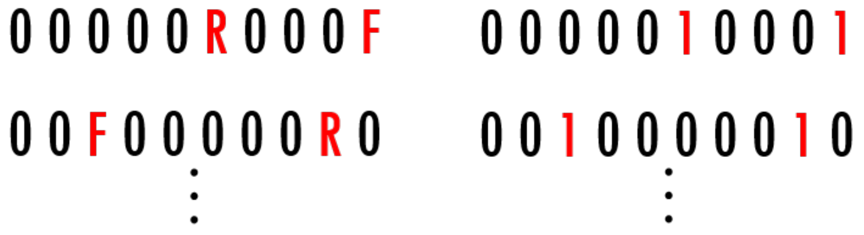
5.2.1 The Visualization Model

In this approach, the output data, which is represented as a binary matrix, is converted into a binary image. This conversion is made by representing ones as black pixels and zeroes as white pixels to make the image visualization better at low transmission rates. The main objective of this model is to check if there are any visible patterns inside the image, such as a set of black pixels that look like a row or a column, as shown in Figure 5.2b, which indicates the existence of a real event. A weak technique refers to any technique that provides a binary image that is not entirely random and can be very easily distinguishable from a completely random binary image such as the one in Figure 5.2a. However, if there are no visible patterns and the image looks completely random, the adversary will not be able to differentiate if the real event exists. In the experiment, one thousand stochastic simulation scenarios are created and converted into one image. Then, this image is compared to the binary images generated by proposed techniques to check if there are any patterns. Different transmission rates of 0.05, 0.1, and 0.15 are evaluated to examine the performance of the proposed techniques.

(a) (b)

Figure 5.2. (a) The completely random binary image. (b) A weak technique binary image.

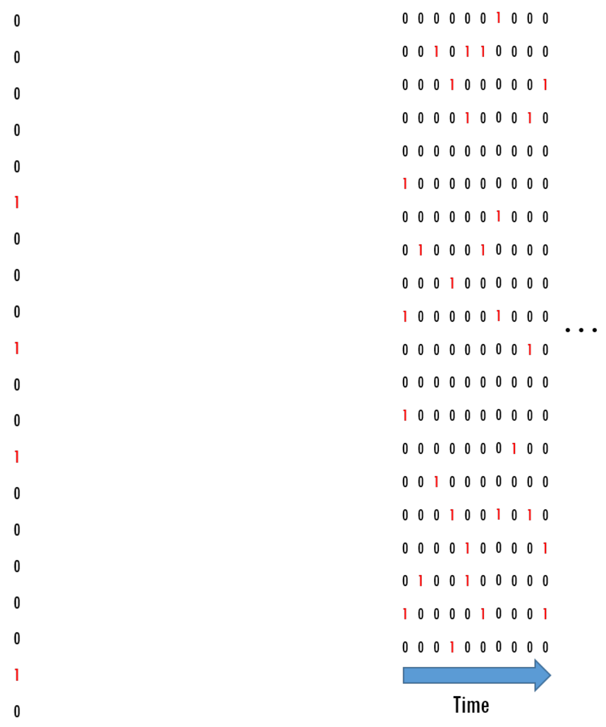
During the lifetime of a *WSN*, some nodes transmit packets, and some nodes do not. In the case of transmitting, there are two possibilities for the packet type: real or fake. However, if there are no transmissions, there is only one possibility: no packet. Therefore, real packets are represented as R, fake packets are represented as F, and no transmissions are represented as 0. Figure 5.3a illustrates a sample of the simulation output. For example, R in the first row of Figure 5.3a means that a real packet is transmitted by node 6 in interval one. F in the second row of Figure 5.3a means that a fake packet is transmitted by node 3 in interval two. This representation is from the system perspective since the system can distinguish between real and fake packets. However, the adversary is unable to differentiate between real and fake packets. Therefore, any packet transmissions whether they are real or fake will be represented as 1 and for no transmissions, they are represented as 0. Figure 5.3b illustrates a sample of the simulation output from the adversary perspective.



(a) (b)

Figure 5.3. (a) Simulation output from the system perspective. (b) Simulation output from the adversary perspective.

The adversary will combine the simulation output rows from Figure 5.3b into a massive column, as shown in Figure 5.4a. This column has one location for the asset. Since the adversary monitors the network over time, several columns are generated; and each column has a different location for the asset to simulate the movement of it. Figure 5.4b shows the output binary matrix when the adversary monitors the *WSN* over time. This binary matrix will be converted into a binary image to extract any suspicious patterns that might lead to the existence of the real event (Figure 5.2b). The target of the proposed techniques is to provide completely random binary images such as the one in Figure 5.2a. As a result, the adversary will be confused about the location of the asset as long as there are no visible traces in the produced images.



(a) (b)

Figure 5.4. (a) Simulation output rows combined into one column. (b) An adversary monitors the *WSN* over time.

5.2.2 The Neural Network Model

Another way to show that the proposed solutions provide a high level of anonymity is to create a neural network and train it on many different binary matrix patterns produced by the proposed techniques. Then, feed the testing data to the trained neural network to see if it can detect the occurrence of an event. Each approach matrix is compared with the random matrix to see if the neural network is able to recognize the difference. In cases where the neural network can distinguish the difference, this would mean that our solutions have a security flaw. Otherwise, the proposed techniques provide a high level of anonymity.

Two thousand cases are used as the training input for the neural network. One thousand cases are random scenarios and the other one thousand cases represent one of our solutions (DUD, DAD, CAD, and $EDAD$). The WSN has 25 sensor nodes and 100 intervals, by multiplying them, each case will have 2500 inputs. These cases are fed to the neural network. W is the weights that are generated by the neural network. At the beginning, these weights are selected randomly. Then, the weight values are changed using a training back-propagation algorithm called Gradient Descent to train the neural network. The aim of Gradient Descent is to find the best combination of weights that minimizes errors between the neural network actual output and expected output. b is a constant. A graphical representation of the neural network is shown in Figure 5.5.

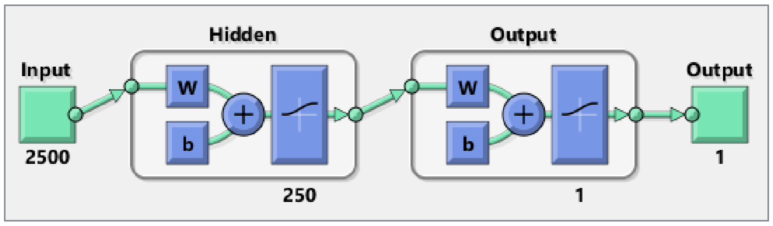


Figure 5.5. Neural network configuration.

The transmission rates of 0.05, 0.1, and 0.15 are used to test each technique. The hidden layer in the feed-forwarded neural network has 250 neurons. Sigmoid is utilized as the activation function for both hidden and output layers. The output layer has one neuron and produces 0, if there is no real event, or 1, otherwise. Two neural network models have been created: one without validation data and the other uses validation data to avoid overfitting the neural network.

5.2.3 The Steganography Model

In the steganography model, the adversary attempts to measure the uncertainty of the system. The ideal case occurs when $d(\alpha, \beta)$ is equal to 0, which indicates that the system is perfectly secure. Therefore, the proposed techniques should provide

$d(\alpha, \beta)$ that is very close to 0 in order to confirm the high anonymity of our techniques. Since the neural network in the previous subsection created some probabilities for false negatives and false positives of the system, these probabilities can be utilized to generate α and β, respectively.

5.3 Enhanced FitProbRate

The *FitProbRate* scheme [54] is developed to provide source anonymity. It uses exponential distribution to predict the next time interval for fake packets. If a real packet is detected, it uses the first interval that satisfies the *A-D* test. However, if more than one real packet is detected, *FitProbRate* recovers the current mean (μ) of the intervals to predict the next interval within:

$$(1-\epsilon)\mu \leq \mu \leq (1+\epsilon) \tag{5.1}$$

Where ϵ is the maximum increase or decrease percentage of the current μ. This mechanism is essential to avoid a mean reduction for the current intervals sample, which could lead the adversary to detect the presence of the real event. In addition, the predicted interval must also satisfy the *A-D* test. This technique will be tested under the proposed neural network model to show how it performs. Furthermore, an enhanced version of *FitProbRate* is developed and compared to the original one regarding the number of operations needed to decide whether a specific sequence follows the exponential distribution or not. In the original technique, each node performs the *A-D* test on the entire sequence of intervals, which increases the number of operations rapidly since the complexity of *A-D* test is $O(n \log n)$. The *A-D* test provides high accuracy when the number of intervals is 7 or more ($n \geq 7$) [91–96]. Therefore, instead of applying the *A-D* test on the entire sequence, it will only be applied on the last ten intervals of the sequence, which reduces the number of operations needed to test the exponentially of a selected sequence.

5.4 EDADP1 vs. EDADP2 vs. FitProbRate

A comprehensive comparison between two of the proposed techniques *EDADP1*, *EDADP2*, and the original *FitProbRate* is conducted. One thousand scenarios of each technique are generated using the same criteria such as the exponential distribution generator function to have a fair comparison. In addition, the average result of the simulation output scenarios is calculated to have a more stable and accurate comparison. The percentage increase in the transmission rate for real packets in the proposed techniques is selected to be 30%. This increase is reasonable because if the percentage increase was extremely high, the existence of the real event can be easily exposed. However, this value should be evaluated based on the application requirements and level of tolerance, which is considered as a trade-off between latency and anonymity.

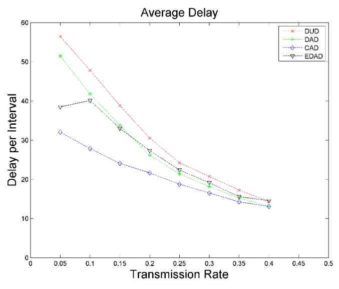
The x-axis of the comparison represents the different number of packets in the real event, whereas the y-axis represents the tested metric. Therefore, every metric in each technique is tested under a different number of real packets to evaluate the technique's performance. The comparison is divided into five different parts: average delay, anonymity level, average processing time, *A-D* test, and polluted scenarios. First, the proposed techniques are compared to *FitProbRate* regarding the average delay. In the anonymity level, *EDADP1*, *EDADP2*, and *FitProbRate* are examined by the proposed neural network model to show the difference between them in terms of anonymity. Moreover, all techniques are tested regarding the average processing time that can be further utilized to provide an indication of the overhead and power consumption. Another metric is the *A-D* test, which shows how strict the generated data sequence follows the exponential distribution. Lastly, polluted scenarios indicate how reliable and stable the proposed techniques are when compared to *FitProbRate*.

CHAPTER 6: RESULTS

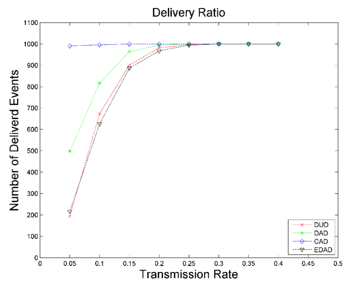
6.1 Performance Results of DUD, DAD, CAD, and EDAD

In Figure 6.1a, *CAD* performs much better than *DAD* and *EDAD* regarding the average delay especially at low transmission rates, which is desired by most of the applications. *DUD* has the worst performance since it does not have any mechanism to improve the delay of real packets. Regarding the delivery ratio, as shown in Figure 6.1b, *CAD* has the best performance. *CAD* has an advantage over the other three techniques because it forces packets of a real event to be sent after a specific waiting time, which is five-intervals in this experiment. *DAD* still performs better than *DUD* since the real packets have a higher probability to be sent than the fake ones. Also, *EDAD* has a higher performance than *DUD* because it uses exponential distribution. Meanwhile, the overhead traffic in Figure 6.1c, is almost the same for *DUD*, *DAD*, and *CAD* techniques because the total number of generated fake packets stays the same for all three techniques. In other words, *CAD* and *DAD* reduce the average delay and increase the delivery ratio without gaining additional overhead when compared to *DUD*.

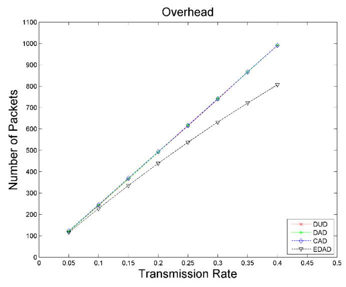
Figure 6.1c shows that *EDAD* reduces the overhead when compared to the other techniques. It decreases the number of overhead packets whenever the transmission rate increases. This reduction indicates that *EDAD* can be used at high transmission rates, which increases the delivery ratio and reduces the delay. However, the performance of *EDAD*, in terms of average delay, is very similar to *DAD* and is not as good as *CAD* because *EDAD* uses the exponential distribution without forcing the real packets to be delivered within maximum delay. Therefore, *EDAD* does not guarantee the maximum delay of the real event. The CAD technique provides a high delivery ratio at low and high transmission rates because it uses a specific mechanism that forces the node to transmit its real packet after a predefined number of intervals. However, this mechanism might lead to a privacy flaw because it violates the uniform distribution sequence. $EDAD$ has a similar performance to CAD regarding the delivery ratio at high transmission rates. However, at low transmission rates, the number of delivered packets is decreased in $EDAD$ because of the delay caused by the exponential distribution. In conclusion, $EDAD$ reduces the overhead, but it increases the delay while keeping a high level of anonymity. In contrast, CAD increases the overhead and decreases the delay with an acceptable level of anonymity.



(a)



(b)



(c)

Figure 6.1. (a) Average delay of real packets. (b) Delivery ratio of real packets. (c) Overhead of real packets.

6.2 Anonymity Analysis Results of DUD, DAD, CAD, and EDAD

6.2.1 The Visualization Model

In Figures 6.2, 6.3, 6.4 and 6.5, at all transmission rates, the proposed techniques binary images look entirely random and are very similar to the random cases binary

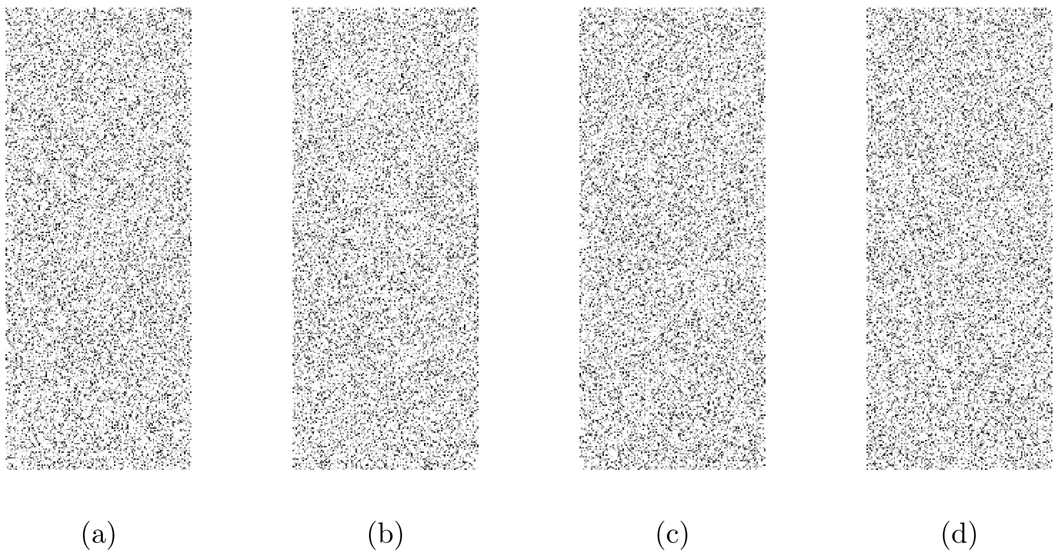


Figure 6.2. The produced binary images with transmission rate of 0.15. (a) The completely random image. (b) The *DUD* image. (c) The *DAD* image. (d) The *CAD* image.

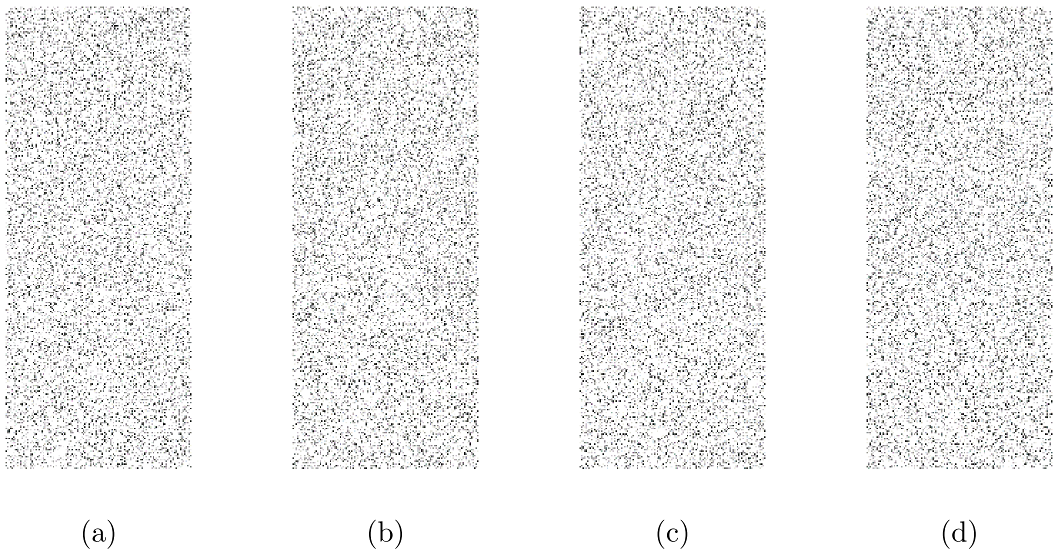


Figure 6.3. The produced binary images with transmission rate of 0.1. (a) The completely random image. (b) The *DUD* image. (c) The *DAD* image. (d) The *CAD* image.

image. The images depict transmission patterns for the *DUD*, *DAD*, *CAD* and *EDAD* techniques do not indicate any visual patterns in the output images, which confirms that the proposed techniques have a high level of anonymity.

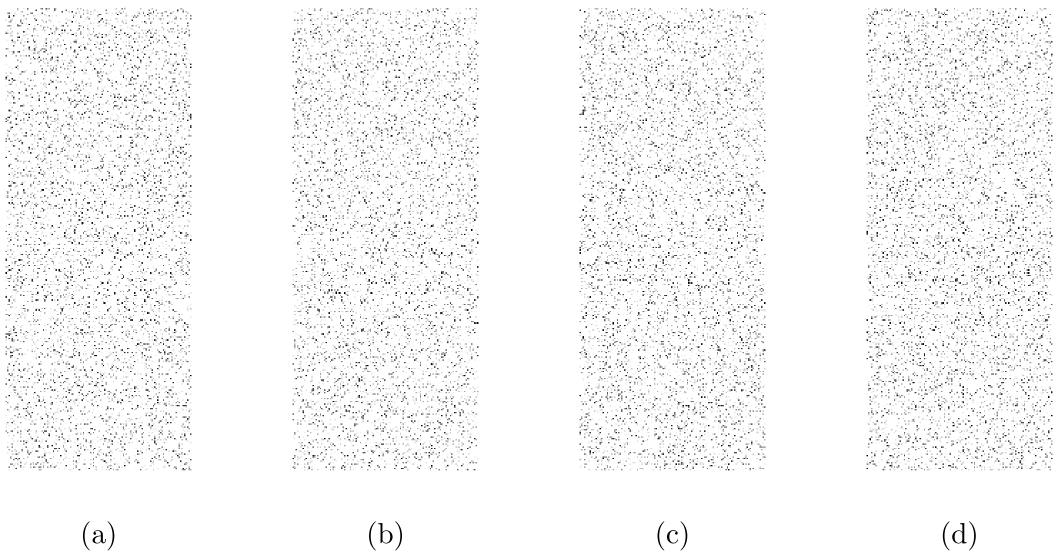


Figure 6.4. The produced binary images with transmission rate of 0.05. (a) The completely random image. (b) The *DUD* image. (c) The *DAD* image. (d) The *CAD* image.

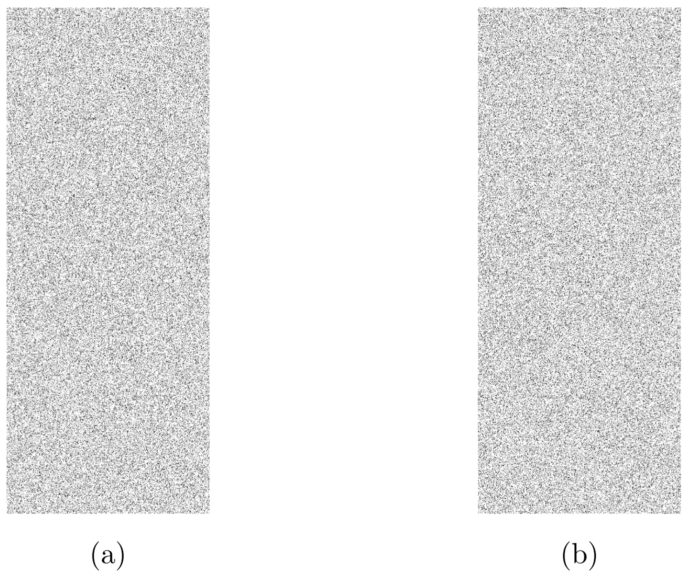
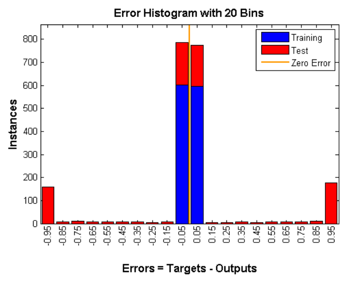


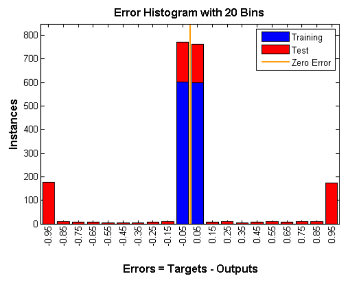
Figure 6.5. The produced binary images with transmission rate of 0.15. (a) The completely random image. (b) The *EDAD* image.

6.2.2 The Neural Network Model

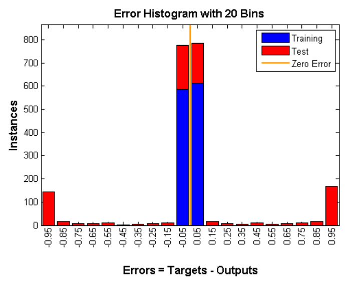
The first neural network model (without validation data) is shown in Figures 6.6, 6.7 and 6.8. The number of instances is the y-axis and errors are the x-axis. Errors are the difference between *targets* that represent the expected outputs and *outputs* that represent the actual outputs. The training data is represented by the blue columns and testing data is represented by the red columns. The error of the training data (blue columns) is almost 0 indicating that the neural network is trained properly, whereas the error of testing data (red columns) is high and almost divided evenly between 0 and 1 in the three techniques at all transmission rates. Moreover, Table 6.1 demonstrates that the uncertainty of the neural network for all techniques at different transmission rates is around 50%, which is the ideal case. That means the neural network is confused about the existence of the real event even after training the network successfully. This confusion is an indication that the proposed techniques are very reliable and guarantee a high level of source anonymity.



(a)

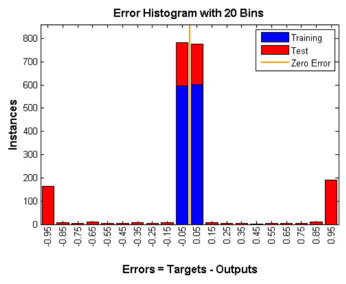


(b)

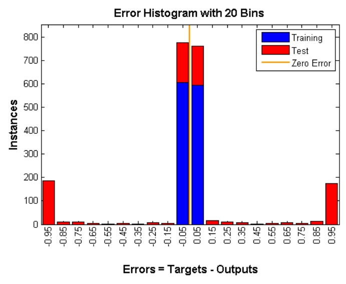


(c)

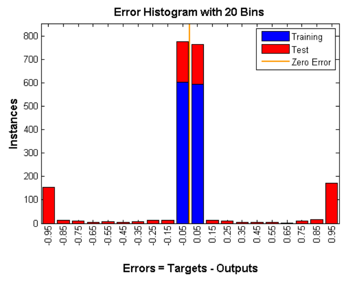
Figure 6.6. *DUD* with transmission rate of (a) 0.05, (b) 0.1, and (c) 0.15.



(a)

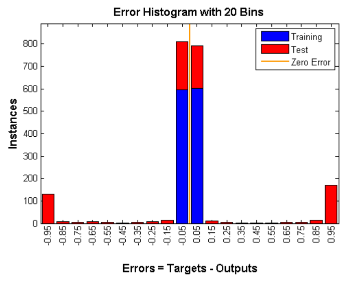


(b)

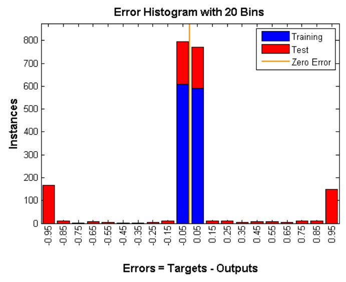


(c)

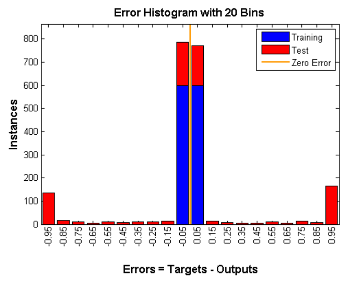
Figure 6.7. *DAD* with transmission rate of (a) 0.05, (b) 0.1, and (c) 0.15.



(a)



(b)



(c)

Figure 6.8. *CAD* with transmission rate of (a) 0.05, (b) 0.1, and (c) 0.15.

The second neural network model (with validation data) is shown in Figure 6.9. The first row of sub-figures in Figure 6.9 shows that the number of instances is the y-axis and errors are the x-axis. Errors are the difference between *targets* that represent

Table 6.1. Percentage error of the testing data.

Rate	**DUD**	**DAD**	**CAD**
0.05	50.4%	49.4%	56.5%
0.1	48.1%	48.6%	53.0%
0.15	52.3%	51.3%	53.4%

the expected outputs and *outputs* that represent the actual outputs. The training data is represented by the blue columns, testing data is represented by the red columns, and validation data is represented by the green columns. The neural network cannot achieve 100% accuracy on the training data (blue columns) since the accuracy does not improve on the validation data (green columns). The second row of sub-figures in Figure 6.9 shows that after 11 or 12 epochs, based on the tested technique, the neural network has an early stop to avoid overfitting, which reduces the generalization of the neural network. The early stop occurs because the large amount of noise that was introduced by the proposed techniques to disturb the neural network. The noise is generated by injecting the *WSN* with dummy traffic that confuses the neural network about the presence of the real event.

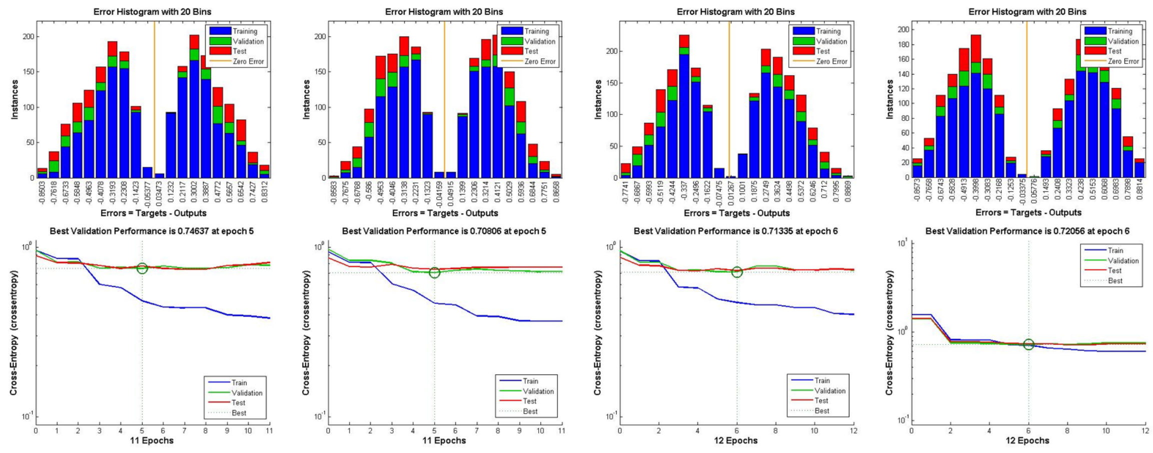


Figure 6.9. Error histogram and validation performance of *DUD*, *DAD*, *CAD*, and *EDAD* when using validation data.

Figure 6.10 shows the test confusion matrices of all techniques. The green square in the first row represents the true negative of the system, the green square in the second row represents the true positive of the system, the red square in the first row represents the false negative of the system, the red square in the second row represents the false positive of the system, and the blue square in the third row represents the accuracy of the neural network. The ideal anonymity from the system perspective is when the accuracy of the neural network is 50%, which indicates that the neural network is totally uncertain about the existence of the real event. DUD achieved 49.3% accuracy, DAD achieved 54.3% accuracy, CAD achieved 53.7% accuracy, and $EDAD$ achieved 56.3% accuracy. These percentages are very close to 50% even after avoiding the overfitting. Accordingly, the test confusion matrix is still unable to distinguish the difference between real and fake events for all techniques, which validate the high level of anonymity provided by the proposed techniques.

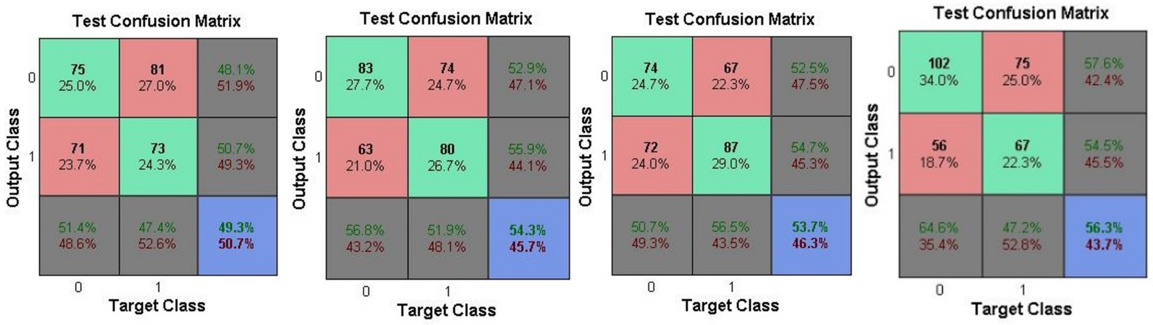


Figure 6.10. The test confusion matrix of DUD, DAD, CAD, and $EDAD$ when using validation data.

6.2.3 The Steganography Model

Table 6.2 shows that $d(\alpha, \beta)$ is very close to 0 in DUD, DAD, and CAD techniques at different transmission rates, which satisfy a very small ϵ that is required by most of the applications to have a high level of anonymity. Moreover, $EDAD$ is tested at the transmission rate of 0.2, which produced $d(\alpha, \beta)$ of 0.001 and a neural network accuracy of 49.4% (Figure 6.11). These numbers indicate the high level of anonymity $EDAD$ technique provides.

Table 6.2. The $d(\alpha, \beta)$ output when uses different probabilities for α and β.

Rate	**DUD**			**DAD**			**CAD**		
	β	α	$\boldsymbol{d(\alpha,\beta)}$	β	α	$\boldsymbol{d(\alpha,\beta)}$	β	α	$\boldsymbol{d(\alpha,\beta)}$
0.05	0.494	0.499	**0.0001**	0.504	0.508	**0.0004**	0.426	0.442	**0.0507**
0.1	0.516	0.521	**0.0004**	0.511	0.517	**0.0023**	0.468	0.472	**0.0104**
0.15	0.474	0.481	**0.0058**	0.486	0.491	**0.0015**	0.461	0.471	**0.0134**

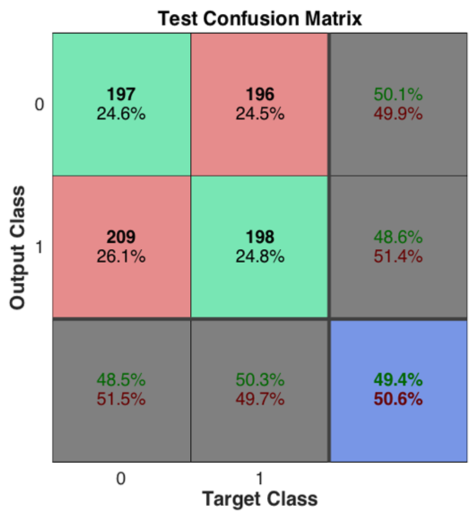


Figure 6.11. The test confusion matrix of *EDAD*.

All analytical models show that the proposed techniques provide a high level of anonymity and the global adversary cannot recognize the existence of the real event.

6.3 Enhanced FitProbRate

6.3.1 Neural Network Model vs. A-D Test Model

This section shows the reliability of the proposed neural network model when compared to the *A-D* test model. The proposed model takes into consideration the *Rate Monitoring*, *Time Correlation*, and the type of distribution when measuring the anonymity of a system. However, the *A-D* test in *FitProbRate* only considers the type of distribution as a measurement for anonymity. The original *FitProbRate* in Figure 6.12a is tested by the proposed neural network model. Three categories are created: The real event has one real packet in the first category, two real packets in the second category, and three real packets in the third category. Each category consists of one thousand cases, further, each case has a different location and time that are generated randomly for the real event. Results show that the proposed neural network is confused about the existence of the real event when it has only one real packet since one packet can be easily hidden within dummy traffic. However, when the number of real packets increases, the proposed neural network model started to detect more cases correctly. In the second category, when the number of real packets is three, the proposed model detected around 66% of the cases correctly, whereas in the third category when the number of real packets is five, the proposed model detected more than 83% of the cases correctly. Therefore, the proposed model performs much better when compared to the ordinary *A-D* test, which fails to identify any of the cases accurately. The reason is that the *A-D* test only relies on testing the sequence distribution to decide whether a technique provides high anonymity or not. In contrast, the proposed model evaluates the anonymity based on many factors that include *Rate Monitoring*, *Time Correlation*, and the type of distribution.

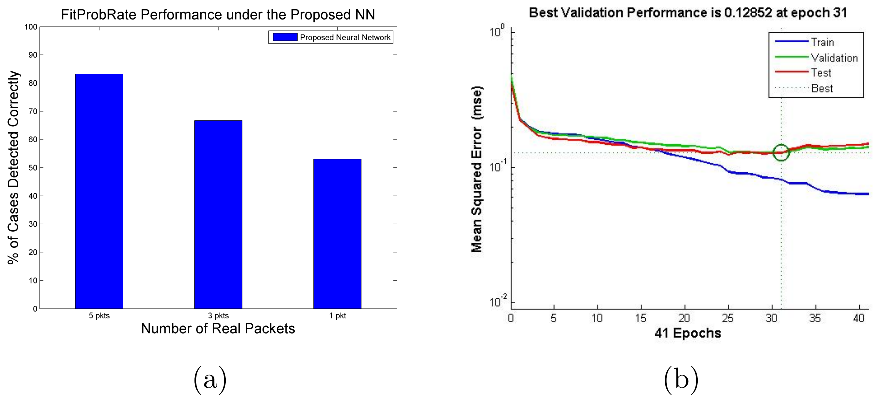
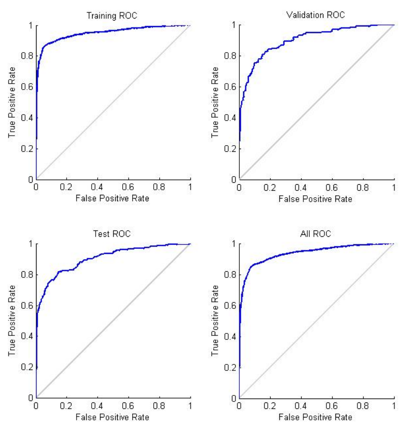
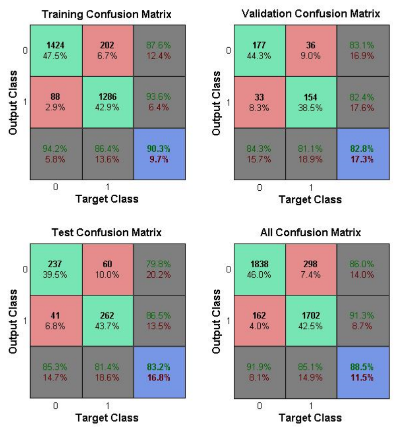


Figure 6.12. (a) The original *FitProbRate* analyzed by the proposed neural network model. (b) The validation performance of the proposed neural network model.

The third category that has five packets in the real event is explained in more details: Figure 6.12b shows that the neural network stopped training after 41 epochs to avoid overfitting. The neural network stopped training since there are no further improvements on the validation data. Figure 6.13a is the receiver operating characteristic, and provides the relationship between the true positive rate and false positive rate. The neural network has a higher performance when the blue line is stretched to the top left corner. If the blue line is stretched to the bottom right corner, low performance is achieved. In Figure 6.13a, training, validation, and testing data are very close to the top left corner indicating the high performance of the proposed model. Figure 6.13b demonstrates the confusion matrix of the training data, validation data, test data and all of the data combined. In the training confusion matrix, the neural network achieved 90.3% accuracy, which is very reasonable because the neural network cannot have 100% accuracy on the training data to avoid overfitting. Validation data achieved 82.8% accuracy and testing data achieved 83.2% accuracy for detection of the real event.





(a) (b)

Figure 6.13. (a) The receiver operating characteristic of the proposed neural network model. (b) The confusion matrices of the proposed neural network model.

6.3.2 Enhanced FitPropRate vs. Original FitProbRate

A comparison regarding the number of operations between the enhanced and original versions of *FitPropRate* is conducted to test the exponentiality of a sequence of intervals. Table 6.3 and Figure 6.14 show that when the number of intervals is equal to ten, both techniques perform similarly. However, when the number of intervals increases, the proposed technique performs much better than the original one since the enhanced version only uses the previous ten intervals. In the case where the number of intervals is one thousand, the original *FitPropRate* needs three thousand operations, whereas the proposed technique only needs ten operations. Since the complexity of A-D test is $n \log n$, testing a smaller number of intervals reduces the number of operations needed to know which type of distribution it is, such as exponential distribution.

Table 6.3. Number of operations ($O(n \log n)$) for one transmitting node.

Number of Intervals	**FitProbRate**	**Proposed Technique**
10	10	10
20	26	10
50	85	10
100	200	10
200	460	10
500	1349	10
1000	3000	10

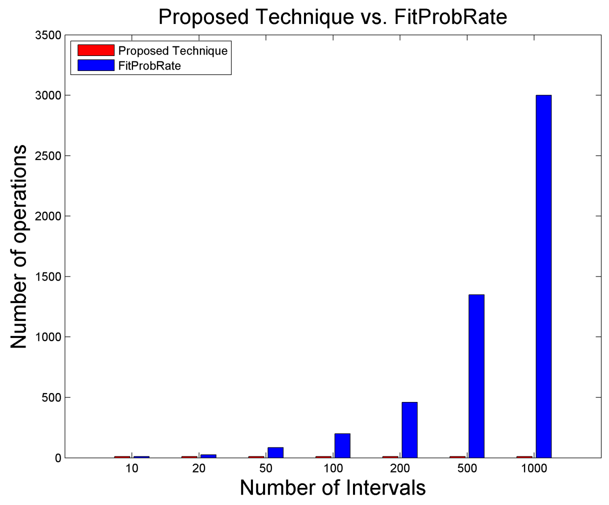


Figure 6.14. Number of intervals vs. number of operations for one transmitting node.

In Table 6.4 and Figure 6.15, the number of transmitting nodes is selected to be five instead of one as in the previous comparison. Results also show that the proposed technique reduces the number of operations needed whenever the number of intervals increases when compared to the original *FitPRobRate*. In the case where the number of intervals is one thousand, the original *FitPropRate* needs fifteen thousand operations, whereas the proposed technique only needs fifty operations.

Table 6.4. Number of operations ($O(n \log n)$) for five transmitting nodes.

Number of Intervals	FitProbRate	Proposed Technique
10	50	50
20	130	50
50	425	50
100	1000	50
200	2301	50
500	6747	50
1000	15000	50

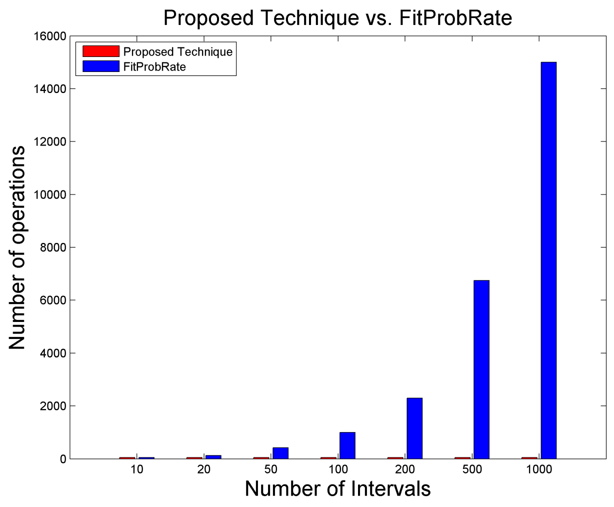
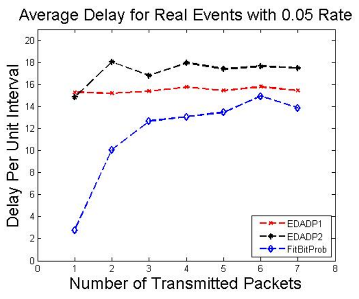


Figure 6.15. Number of intervals vs. number of operations for five transmitting nodes.

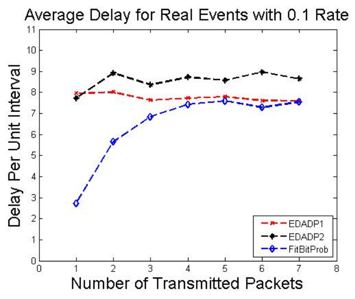
6.4 EDADP1 vs. EDADP2 vs. FitProbRate

6.4.1 Average Delay

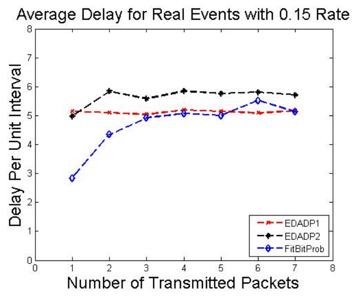
In Figure 6.16, the y-axis represents the delay per unit interval and the x-axis represents the number of transmitted real packets. In Figure 6.16c, the average delay of *EDADP1* is similar to *FitProbRate* when the number of real packets increases in the case of 0.15 transmission rate. However, *EDADP2* performance is the worst since it does not increase the transmission rate of all real packets. *EDADP2* only increases the transmission rate of the odd real packets. Unlike *EDADP1* that increases the transmission rate of all real packets and *FitProbRate* that transmits the real packets as soon as the entire sequence satisfies the *A-D* test. If the transmission rate is decreased to 0.1, *EDADP1* and *FitProbRate* perform similarly especially when the number of real packets is four or more with a slight advantage to *FitProbRate*, as shown in Figure 6.16b. This slightly less delay comes with a massive drawback regarding the anonymity level, as shown in Figure 6.17. *EDADP2* still provides higher delay when compared to the other techniques. When the transmission rate is 0.05 (Figure 6.16a), the performance of all three techniques dropped since the real packets are sent using longer intervals. However, *FitProbRate* still has a slightly better performance over the other two techniques due to the transmitting mechanism that is based on satisfying the *A-D* test, but this mechanism sacrifices the anonymity.



(a)



(b)



(c)

Figure 6.16. Average delay of real packets at (a) 0.05, (b) 0.1, and (c) 0.15 transmission rates.

6.4.2 Anonymity Level

In Figure 6.17, the y-axis is the anonymity level and the x-axis is the number of transmitted real packets. The ideal anonymity level is zero. *EDADP2* has the best performance at all transmission rates. *EDADP2* provides a higher anonymity level than *EDADP1*, especially when the number of real packets increases. The main reason is the utilized mechanism by *EDADP2* that increases the transmission rate of the odd real packets and uses the original transmission rate for the even ones, which confuses the neural network about the existence of the real event. *EDADP1* provides a high level of anonymity when the number of real packets is small and an acceptable level of anonymity when the number of real packets is large. The *EDADP1* has a lower level of anonymity when compared to *EDADP2* because *EDADP1* increases the transmission rate of all real packets. The performance of *FitProbRate* is poor and far from the ideal anonymity level at all transmission rates because it only uses

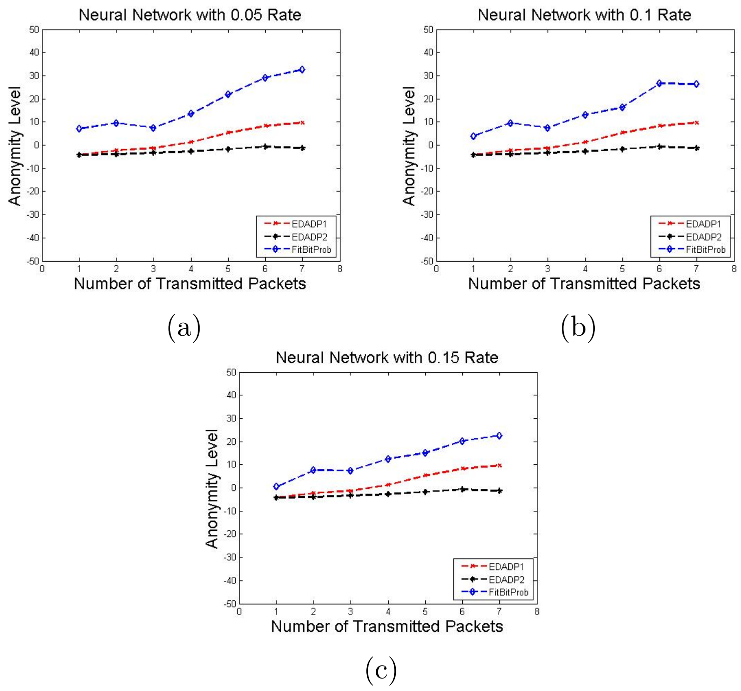
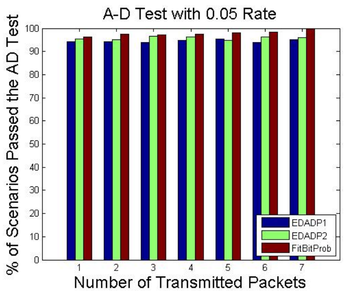


Figure 6.17. Anonymity level using the proposed neural network model at (a) 0.05, (b) 0.1, and (C) 0.15 transmission rates.

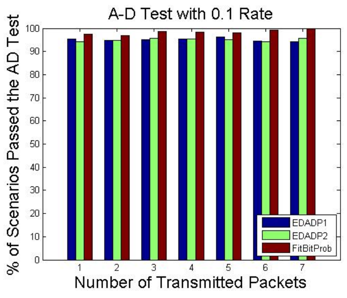
the *A-D* test to measure the anonymity level of a system. In contrast, *EDADP1* and *EDADP2* attempt to avoid *Rate Monitoring* attacks, *Time Correlation* attacks, and the *A-D* test, which results in a much higher level of anonymity.

6.4.3 A-D Test

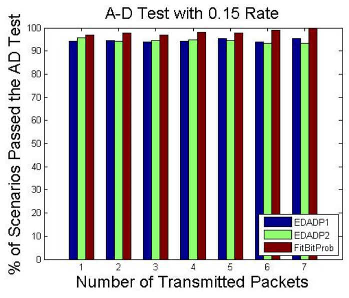
The *A-D* test is a useful metric that achieves a high level of anonymity. Nevertheless, it cannot be used as a stand-alone parameter. The *A-D* test should be merged with other metrics such as *Rate Monitoring* and *Time Correlation* to have a sufficient analysis of a system. In Figure 6.18, the y-axis represents the percentage of scenarios that passed the *A-D* test, whereas the x-axis represents the number of transmitted real packets. All techniques perform similarly in all transmission rates with a slightly better performance for *FitProbRate*. The reason for this outcome is that *FitProbRate* applies the *A-D* test on the entire sequence every time a real packet is detected. However, this unnoticeable difference comes with massive processing time, overhead, and power consumption, as shown in Figure 6.19.



(a)



(b)



(c)

Figure 6.18. Anderson-Darling test at (a) 0.05, (b) 0.1, and (c) 0.15 transmission rates.

6.4.4 Average Processing Time

Figure 6.19 illustrates the efficiency of the proposed techniques in regards to the average processing time, which can also indicate the overhead and power consumption of the *WSN*. The y-axis represents the processing delay in seconds, whereas the x-axis represents the number of transmitted real packets. *EDADP1* and *EDADP2* have significantly less processing time when compared to *FitProbRate* at all transmission rates. Additionally, the proposed techniques are not impacted by the number of real packets. The average processing time needed by *EDADP1* and *EDADP2* is stable regarding the number of real packets since the *A-D* test is not part of the proposed techniques. In contrast, *FitProbRate* has a higher processing time, especially when the number of real packets increases. This result was expected since *FitProbRate* applies the *A-D* test every time a real packet is detected, which leads to more processing delay, overhead, and power consumption.

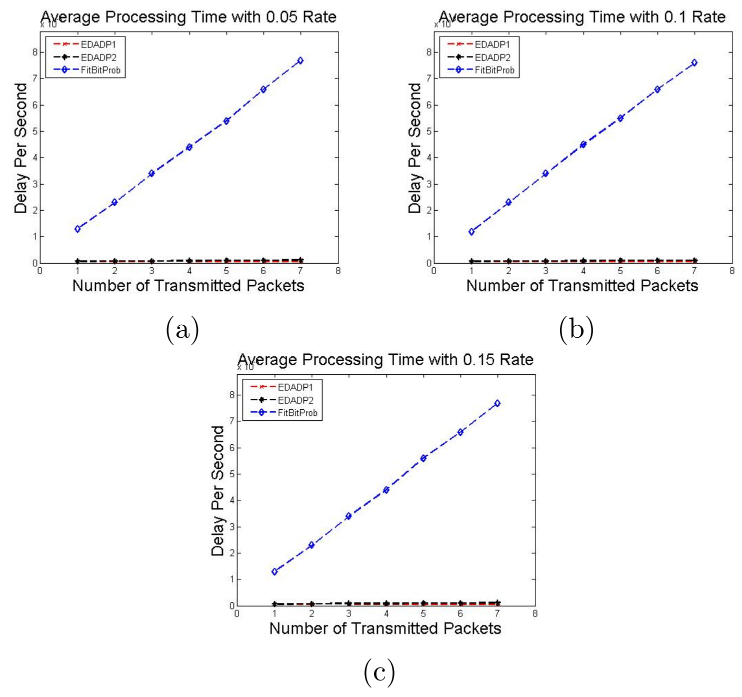


Figure 6.19. Average processing time at (a) 0.05, (b) 0.1, and (c) 0.15 transmission rates.

Figure 6.20 shows that *EDADP1* has a slight advantage over *EDADP2* regarding the average processing time because *EDADP1* does not have to track the sequence number of real packets. In other words, *EDADP2* has more average processing time because it needs to track the sequence number of each real packet to decide whether the packet is odd or even.

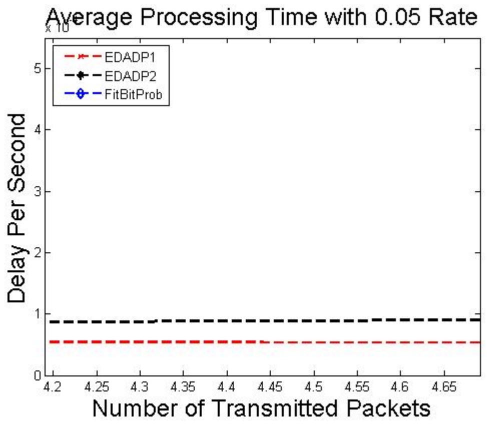
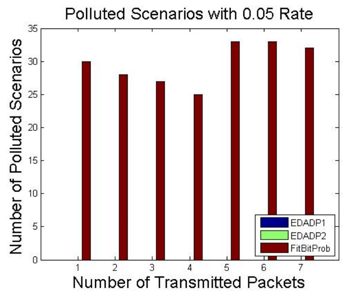


Figure 6.20. Average processing time of *EDADP1* and *EDADP2* at 0.05 transmission rate.

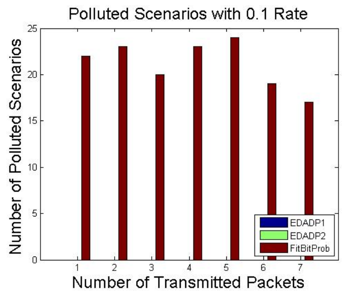
6.4.5 Polluted Scenarios

The polluted scenarios show the reliability of the proposed techniques. In Figure 6.21, the y-axis represents the number of polluted scenarios out of one thousand, whereas the x-axis is the number of transmitted real packets. The *FitProbRate* technique is based on probability and the *A-D* test. Therefore, in some situations, *FitProbRate* struggles to find the appropriate transmitting interval that satisfies the *A-D* test. In this type of situations, *FitProbRate* enters an infinite loop in trying to determine the time of the next interval, which causes the polluted scenario. In *FitProbRate*, the number of polluted scenarios ranges from 15 to 33 cases based on the selected transmission rate and the number of transmitted real packets. In the proposed techniques, *EDADP1* and *EDADP2* achieved zero polluted scenarios because of their mechanism that does not allow any polluted scenarios to be created since the *A-D* test is not part of the proposed techniques.

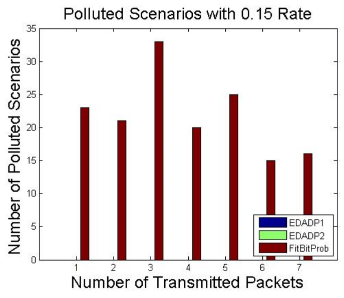
This metric shows the high reliability of *EDADP1* and *EDADP2* when compared to the *FitProbRate* scheme.



(a)



(b)



(c)

Figure 6.21. Polluted scenarios at (a) 0.05, (b) 0.1, and (c) 0.15 transmission rates.

CONCLUSION

Based on the experiments and simulation results of this work, the null (H_0) hypothesis cannot be rejected, which means that the adversary is confused about the existence of the real event when the *WSN* is injected with dummy traffic. Given the lack of techniques that provide high anonymity to the source node, this work presents *SLP* techniques to improve the performance of *WSNs* while keeping a high level of anonymity against a global adversary model that is capable of monitoring the entire traffic in the network. Six different techniques, *DUD*, *DAD*, *CAD*, *EDAD*, *EDADP1*, and *EDADP2* were developed to provide a low transmission rate while still maintaining anonymity. *WSN* applications that need privacy can select among the proposed techniques based on the required level of anonymity with respect to delay, delivery ratio, and overhead.

Previous literature does not provide validation models of the performance in terms of anonymity. Therefore, all proposed techniques were tested under comprehensive analytical models (*Visualization*, *Neural Network*, and *Steganography*) to confirm that they provide a high level of anonymity. The simulations indicate that *CAD* provides the best performance in terms of average delay and delivery ratio while guaranteeing the delivery of the event within a certain delay constraint. However, *EDAD* outperforms other techniques with respect to overhead. The well-known *FitProbRate* technique was tested under the proposed neural network model. The results demonstrate that it performs poorly, especially when the number of real packets increases.

An enhanced version of *FitProbRate* was also developed and compared to the original one. The results show that the proposed technique greatly reduces the number of operations required to determine if a specific sequence follows an exponential distribution. Moreover, a comprehensive comparison between *EDADP1*, *EDADP2*, and *FitProbRate* was conducted that focuses on the average delay, anonymity level, average processing time, Anderson-Darling test, and polluted scenarios. Results indicate that *EDADP1* and *EDADP2* outperform the *FitProbRate* scheme overall. The results confirm the high level of anonymity, efficient average processing time, and reasonable average delay produced by the proposed techniques.

In the future work, other classifiers will be considered for the global adversary model such as Deep Neural Networks and Support Vector Machine (SVM). A more sophisticated global adversary model that is capable of utilizing parallelism will also be examined. In addition, an implementation of different routing protocols to show the impact of these on the proposed techniques' performance. Lastly, testing the scalability of the proposed techniques by using larger networks will be attempted.

REFERENCES

[1] V. P. Illiano and E. C. Lupu, "Detecting malicious data injections in wireless sensor networks: A survey," *ACM Computing Surveys (CSUR)*, vol. 48, no. 2, p. 24, 2015.

[2] A. Salleh, K. Mamat, and M. Y. Darus, "Integration of wireless sensor network and web of things: Security perspective," in *the 8th IEEE Control and System Graduate Research Colloquium (ICSGRC)*, 2017, pp. 138–143.

[3] W. Dargie and C. Poellabauer, *Fundamentals of wireless sensor networks: theory and practice.* John Wiley & Sons, 2010.

[4] M. Conti, J. Willemsen, and B. Crispo, "Providing source location privacy in wireless sensor networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1238–1280, 2013.

[5] Z. A. Eu, H.-P. Tan, and W. K. Seah, "Design and performance analysis of mac schemes for wireless sensor networks powered by ambient energy harvesting," *Ad Hoc Networks*, vol. 9, no. 3, pp. 300–323, 2011.

[6] D. K. Noh and J. Hur, "Using a dynamic backbone for efficient data delivery in solar-powered wsns," *Journal of Network and Computer Applications*, vol. 35, no. 4, pp. 1277–1284, 2012.

[7] S. B. Lande and S. Z. Kawale, "Energy efficient routing protocol for wireless sensor networks," in *the 8th International Conference on Computational Intelligence and Communication Networks (CICN)*, Dec 2016, pp. 77–81.

[8] K. Pongaliur and L. Xiao, "Sensor node source privacy and packet recovery under eavesdropping and node compromise attacks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 4, p. 50, 2013.

[9] K. Hayawi, A. Mortezaei, and M. V. Tripunitara, "The limits of the trade-off between query-anonymity and communication-cost in wireless sensor networks," in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, 2015, pp. 337–348.

[10] J. B. Hughes, P. Lazaridis, I. Glover, and A. Ball, "Opportunities for transmission power control protocols in wireless sensor networks," in *the 23rd International Conference on Automation and Computing (ICAC)*, Sept 2017, pp. 1–5.

[11] A. Tyagi, J. Kushwah, and M. Bhalla, "Threats to security of wireless sensor networks," in *the 7th IEEE International Conference on Cloud Computing, Data Science & Engineering-Confluence*, 2017, pp. 402–405.

[12] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, 2002, pp. 88–97.

[13] I. F. Akyildiz and M. C. Vuran, *Wireless sensor networks.* John Wiley & Sons, 2010, vol. 4.

[14] A. Praveena and S. Smys, "Efficient cryptographic approach for data security in wireless sensor networks using mes vu," in *the 10th IEEE International Conference on Intelligent Systems and Control (ISCO)*, 2016, pp. 1–6.

[15] P. Kamat, W. Xu, W. Trappe, and Y. Zhang, “Temporal privacy in wireless sensor networks: Theory and practice,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 4, p. 28, 2009.

[16] J. Celestine, K. Vallepalli, T. Vinayaraj, J. Almotir, and A. Abuzneid, “An energy efficient flooding protocol for enhanced security in wireless sensor networks,” in *IEEE Systems, Applications and Technology Conference (LISAT), Long Island*, 2015, pp. 1–6.

[17] W. Tan, K. Xu, and D. Wang, “An anti-tracking source-location privacy protection protocol in wsns based on path extension,” *IEEE internet of things journal*, vol. 1, no. 5, pp. 461–471, 2014.

[18] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: a survey,” *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[19] T. Arampatzis, J. Lygeros, and S. Manesis, “A survey of applications of wireless sensors and wireless sensor networks,” in *Proceedings of the IEEE International Symposium on Intelligent Control and Automation*, 2005, pp. 719–724.

[20] A. Milenković, C. Otto, and E. Jovanov, “Wireless sensor networks for personal health monitoring: Issues and an implementation,” *Computer communications*, vol. 29, no. 13, pp. 2521–2533, 2006.

[21] J. Yick, B. Mukherjee, and D. Ghosal, “Wireless sensor network survey,” *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[22] S. Pirbhulal, H. Zhang, W. Wu, and Y.-T. Zhang, “A comparative study of fuzzy vault based security methods for wirless body sensor networks,” in *the 10th IEEE International Conference on Sensing Technology (ICST)*, 2016, pp. 1–6.

[23] P. Belsis and G. Pantziou, "Protecting anonymity in wireless medical monitoring environments," in *Proceedings of the 4th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, 2011, p. 55.

[24] J. B. Hughes, P. Lazaridis, I. Glover, and A. Ball, "A survey of link quality properties related to transmission power control protocols in wireless sensor networks," in *the 23rd IEEE International Conference on Automation and Computing (ICAC)*, 2017, pp. 1–5.

[25] G.-A. L. Zodi, G. P. Hancke, G. P. Hancke, and A. B. Bagula, "Enhanced centroid localization of wireless sensor nodes using linear and neighbor weighting mechanisms," in *Proceedings of the 9th ACM International Conference on Ubiquitous Information Management and Communication*, 2015, p. 43.

[26] F. Alfayez, "A wireless sensor network system for border security and crossing detection," Ph.D. dissertation, Manchester Metropolitan University, 2015.

[27] T. Azzabi, H. Farhat, and N. Sahli, "A survey on wireless sensor networks security issues and military specificities," in *IEEE International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, 2017, pp. 66–72.

[28] P. Pancholi and A. S. Yadav, "Energy efficient density-based k clusters for wireless sensor networks," in *the 7th IEEE Power India International Conference (PIICON)*, 2016, pp. 1–6.

[29] K. K. Gagneja, "Secure communication scheme for wireless sensor networks to maintain anonymity," in *IEEE International Conference on Computing, Networking and Communications (ICNC)*, 2015, pp. 1142–1147.

[30] R. D. Shinganjude and D. P. Theng, "Inspecting the ways of source anonymity in wireless sensor network," in *the 4th International Conference on Communication Systems and Network Technologies (CSNT)*, 2014, pp. 705–707.

[31] R. V. Steiner and E. Lupu, “Attestation in wireless sensor networks: A survey,” *ACM Computing Surveys (CSUR)*, vol. 49, no. 3, p. 51, 2016.

[32] M. Guerrero-Zapata, R. Zilan, J. M. Barceló-Ordinas, K. Bicakci, and B. Tavli, “The future of security in wireless multimedia sensor networks,” *Telecommunication Systems*, vol. 45, no. 1, pp. 77–91, 2010.

[33] Y. Wang, G. Attebury, and B. Ramamurthy, “A survey of security issues in wireless sensor networks,” 2006.

[34] C. Gu, M. Bradbury, A. Jhumka, and M. Leeke, “Assessing the performance of phantom routing on source location privacy in wireless sensor networks,” in *the 21st IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2015, pp. 99–108.

[35] A. Gurjar and A. B. Patil, “Cluster based anonymization for source location privacy in wireless sensor network,” in *IEEE International Conference on Communication Systems and Network Technologies (CSNT)*, 2013, pp. 248–251.

[36] A.-s. Abuzneid, T. Sobh, and M. Faezipour, “An enhanced communication protocol for anonymity and location privacy in wsn,” in *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2015, pp. 91–96.

[37] M. Bradbury, M. Leeke, and A. Jhumka, “A dynamic fake source algorithm for source location privacy in wireless sensor networks,” in *IEEE Trustcom/BigDataSE/ISPA*, vol. 1, 2015, pp. 531–538.

[38] L. Yao, L. Kang, P. Shang, and G. Wu, “Protecting the sink location privacy in wireless sensor networks,” *Personal and ubiquitous computing*, vol. 17, no. 5, pp. 883–893, 2013.

[39] I. J. Habeeb and R. A. Muhajjar, "Secured wireless sensor network using improved key management," in *Proceedings of the 5th ACM International Conference on Network, Communication and Computing*, 2016, pp. 302–305.

[40] M. Chaudhari and S. Dharawath, "Toward a statistical framework for source anonymity in sensor network using quantitative measures," in *IEEE International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, 2015, pp. 1–5.

[41] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, 2002, pp. 41–47.

[42] L. Zhou, C. Wan, J. Huang, B. Pei, and C. Chen, "The location privacy of wireless sensor networks: Attacks and countermeasures," in *the 9th IEEE International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*, 2014, pp. 64–71.

[43] M. M. Mahmoud and X. Shen, "A cloud-based scheme for protecting source-location privacy against hotspot-locating attack in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 10, pp. 1805–1818, 2012.

[44] S. S. Tabrizi and D. Ibrahim, "Security of the internet of things: An overview," in *Proceedings of the ACM International Conference on Communication and Information Systems*, 2016, pp. 146–150.

[45] P. K. Roy, J. P. Singh, P. Kumar *et al.*, "An efficient privacy preserving protocol for source location privacy in wireless sensor networks," in *IEEE International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, 2016, pp. 1093–1097.

[46] R. Manjula and R. Datta, "An energy-efficient routing technique for privacy preservation of assets monitored with wsn," in *IEEE Students' Technology Symposium (TechSym)*, 2014, pp. 325–330.

[47] M. Guo, X. Jin, N. Pissinou, S. Zanlongo, B. Carbunar, and S. S. Iyengar, "In-network trajectory privacy preservation," *ACM Computing Surveys (CSUR)*, vol. 48, no. 2, p. 23, 2015.

[48] I. Tomić and J. A. McCann, "A survey of potential security issues in existing wireless sensor network protocols," *IEEE Internet of Things Journal*, 2017.

[49] Z. Xiaoling, H. Donghui, H. Zhengfeng, and D. Liang, "A location privacy preserving solution to resist passive and active attacks in vanet," *China Communications*, vol. 11, no. 9, pp. 60–67, 2014.

[50] T. Li, J. Ma, C. Sun, D. Wei, and N. Xi, "Pvad: Privacy-preserving verification for secure routing in ad hoc networks," in *International Conference on Networking and Network Applications (NaNA)*, Oct 2017, pp. 5–10.

[51] P. Spachos, D. Toumpakaris, and D. Hatzinakos, "Angle-based dynamic routing scheme for source location privacy in wireless sensor networks," in *the 79th IEEE Vehicular Technology Conference (VTC Spring)*, 2014, pp. 1–5.

[52] B. Alomair, A. Clark, J. Cuellar, and R. Poovendran, "Toward a statistical framework for source anonymity in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 2, pp. 248–260, 2013.

[53] A.-S. Abuzneid, T. Sobh, M. Faezipour, A. Mahmood, and J. James, "Fortified anonymous communication protocol for location privacy in wsn: a modular approach," *Sensors*, vol. 15, no. 3, pp. 5820–5864, 2015.

[54] M. Shao, Y. Yang, S. Zhu, and G. Cao, "Towards statistically strong source anonymity for sensor networks," in *the 27th IEEE Conference on Computer Communications (INFOCOM)*, 2008, pp. 51–55.

[55] W. Xiao, H. Zhang, Q. Wen, and W. Li, "Passive rfid-supported source location privacy preservation against global eavesdroppers in wsn," in *the 5th IEEE International Conference on Broadband Network & Multimedia Technology (IC-BNMT)*, 2013, pp. 289–293.

[56] H. Wang, B. Sheng, and Q. Li, "Privacy-aware routing in sensor networks," *Computer Networks*, vol. 53, no. 9, pp. 1512–1529, 2009.

[57] R. Di Pietro and A. Viejo, "Location privacy and resilience in wireless sensor networks querying," *Computer Communications*, vol. 34, no. 3, pp. 515–523, 2011.

[58] J.-H. Park, Y.-H. Jung, H. Ko, J.-J. Kim, and M.-S. Jun, "A privacy technique for providing anonymity to sensor nodes in a sensor network," *Ubiquitous Computing and Multimedia Applications*, pp. 327–335, 2011.

[59] Y. Fan, J. Chen, X. Lin, and X. Shen, "Preventing traffic explosion and achieving source unobservability in multi-hop wireless networks using network coding," in *IEEE Global Telecommunications Conference (GLOBECOM 2010)*, 2010, pp. 1–5.

[60] Y. Fan, Y. Jiang, H. Zhu, and X. Shen, "An efficient privacy-preserving scheme against traffic analysis attacks in network coding," in *IEEE INFOCOM*, 2009, pp. 2213–2221.

[61] J. F. Laikin, M. Bradbury, C. Gu, and M. Leeke, "Towards fake sources for source location privacy in wireless sensor networks with multiple sources," in

IEEE International Conference on Communication Systems (ICCS), 2016, pp. 1–6.

[62] A. Jhumka, M. Leeke, and S. Shrestha, "On the use of fake sources for source location privacy: Trade-offs between energy and privacy," *The Computer Journal*, vol. 54, no. 6, pp. 860–874, 2011.

[63] X. Zha, K. Zheng, and D. Zhang, "Anti-pollution source location privacy preserving scheme in wireless sensor networks," in *the 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2016, pp. 1–8.

[64] Y. Pant and H. Bhadauria, "Performance study of routing protocols in wireless sensor network," in *the 8th IEEE International Conference on Computational Intelligence and Communication Networks (CICN)*, 2016, pp. 134–138.

[65] A. Proaño and L. Lazos, "Perfect contextual information privacy in wsns under-colluding eavesdroppers," in *Proceedings of the 6th ACM conference on Security and privacy in wireless and mobile networks*, 2013, pp. 89–94.

[66] K. Mehta, D. Liu, and M. Wright, "Location privacy in sensor networks against a global eavesdropper," in *IEEE International Conference on Network Protocols (ICNP)*, 2007, pp. 314–323.

[67] Y. Yang, M. Shao, S. Zhu, B. Urgaonkar, and G. Cao, "Towards event source unobservability with minimum network traffic in sensor networks," in *Proceedings of the 1st ACM conference on Wireless network security*, 2008, pp. 77–88.

[68] X. Niu, C. Wei, W. Feng, and Q. Chen, "Osap: Optimal-cluster-based source anonymity protocol in delay-sensitive wireless sensor networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2014, pp. 2880–2885.

[69] C. M. George and M. Kumar, "Cluster based location privacy in wireless sensor networks against a universal adversary," in *IEEE International Conference on Information Communication and Embedded Systems (ICICES)*, 2013, pp. 288–293.

[70] Y. Ouyang, Z. Le, D. Liu, J. Ford, and F. Makedon, "Source location privacy against laptop-class attacks in sensor networks," in *Proceedings of the 4th ACM international conference on Security and privacy in communication netowrks*, 2008, p. 5.

[71] A. Abbasi, A. Khonsari, and M. S. Talebi, "Source location anonymity for sensor networks," in *the 6th IEEE Consumer Communications and Networking Conference (CCNC)*, 2009, pp. 1–5.

[72] K. Bicakci, H. Gultekin, B. Tavli, and I. E. Bagci, "Maximizing lifetime of event-unobservable wireless sensor networks," *Computer Standards & Interfaces*, vol. 33, no. 4, pp. 401–410, 2011.

[73] W. Yang and W. Zhu, "Protecting source location privacy in wireless sensor networks with data aggregation," *Ubiquitous Intelligence and Computing*, pp. 252–266, 2010.

[74] Y. Yang, J. Zhou, R. H. Deng, and F. Bao, "Better security enforcement in trusted computing enabled heterogeneous wireless sensor networks," *Security and Communication Networks*, vol. 4, no. 1, pp. 11–22, 2011.

[75] R. Lu, X. Lin, H. Zhu, and X. Shen, "Tesp2: Timed efficient source privacy preservation scheme for wireless sensor networks," in *IEEE International Conference on Communications (ICC)*, 2010, pp. 1–6.

[76] J.-P. Sheu, J.-R. Jiang, and C. Tu, "Anonymous path routing in wireless sensor networks," in *IEEE International Conference on Communications (ICC)*, 2008, pp. 2728–2734.

[77] J.-R. Jiang, J.-P. Sheu, C. Tu, J.-W. Wu *et al.*, "An anonymous path routing (apr) protocol for wireless sensor networks." *J. Inf. Sci. Eng.*, vol. 27, no. 2, pp. 657–680, 2011.

[78] G. Suarez-Tangil, E. Palomar, B. Ramos, and A. Ribagorda, "An experimental comparison of source location privacy methods for power optimization in wsns," in *Proceedings of the 3rd WSEAS international conference on Advances in sensors, signals and materials*, 2010, pp. 79–84.

[79] S. Kokalj-Filipović, F. Le Fessant, and P. Spasojević, "The quality of source location protection in globally attacked sensor networks," in *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 2011, pp. 44–49.

[80] K. Mehta, D. Liu, and M. Wright, "Protecting location privacy in sensor networks against a global eavesdropper," *IEEE Transactions on Mobile Computing*, vol. 11, no. 2, pp. 320–336, 2012.

[81] R. Doomun, T. Hayajneh, P. Krishnamurthy, and D. Tipper, "Secloud: Source and destination seclusion using clouds for wireless ad hoc networks," in *IEEE Symposium on Computers and Communications (ISCC)*, 2009, pp. 361–367.

[82] S. Ortolani, M. Conti, B. Crispo, and R. Di Pietro, "Events privacy in wsns: A new model and its application," in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2011, pp. 1–9.

[83] J. Deng, R. Han, and S. Mishra, "Decorrelating wireless sensor network traffic to inhibit traffic analysis attacks," *Pervasive and Mobile Computing*, vol. 2, no. 2, pp. 159–186, 2006.

[84] D. Huang, "Traffic analysis-based unlinkability measure for ieee 802.11 b-based communication systems," in *Proceedings of the 5th ACM workshop on Wireless security*, 2006, pp. 65–74.

[85] E. Erdemir and T. E. Tuncer, "Path planning and localization for mobile anchor based wireless sensor networks," in *the 25th IEEE European Signal Processing Conference (EUSIPCO)*, 2017, pp. 131–135.

[86] Y. Xiong, N. Wu, and H. Wang, "On the performance limits of cooperative localization in wireless sensor networks with strong sensor position uncertainty," *IEEE Communications Letters*, 2017.

[87] J. M. Pak, C. K. Ahn, Y. S. Shmaliy, and M. T. Lim, "Improving reliability of particle filter-based localization in wireless sensor networks via hybrid particle/fir filtering," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 5, pp. 1089–1098, 2015.

[88] R. Choudhary, A. Deepak, and S. Kumar, "A study of energy efficient transmission protocol in wirless sensor network," in *the 13th IEEE International Conference on Wireless and Optical Communications Networks (WOCN)*, 2016, pp. 1–4.

[89] S. Katzenbeisser and F. Petitcolas, *Information hiding techniques for steganography and digital watermarking.* Artech house, 2000.

[90] J. Fridrich, *Steganography in digital media: principles, algorithms, and applications.* Cambridge University Press, 2009.

[91] M. A. Stephens, "Edf statistics for goodness of fit and some comparisons," *Journal of the American statistical Association*, vol. 69, no. 347, pp. 730–737, 1974.

[92] M. A. Stephens, "Asymptotic results for goodness-of-fit statistics with unknown parameters," *The Annals of Statistics*, pp. 357–369, 1976.

[93] M. A. Stephens, "Goodness of fit for the extreme value distribution," *Biometrika*, vol. 64, no. 3, pp. 583–588, 1977.

[94] M. A. Stephens, "Goodness of fit tests with special reference to tests for exponentiality," Stanford University, Department of Statistics, CA, Tech. Rep., 1978.

[95] M. A. Stephens, "Tests of fit for the logistic distribution based on the empirical distribution function," *Biometrika*, vol. 66, no. 3, pp. 591–595, 1979.

[96] F. J. O'Reilly and M. A. Stephens, "Characterizations and goodness of fit tests," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 353–360, 1982.

[97] B. Musznicki and P. Zwierzykowski, "Survey of simulators for wireless sensor networks," *International Journal of Grid and Distributed Computing*, vol. 5, no. 3, pp. 23–50, 2012.

[98] K. Fall and K. Varadhan, "The ns manual (formerly ns notes and documentation)," *The VINT project*, vol. 47, 2005.

[99] Q. I. Ali, "Simulation framework of wireless sensor network (wsn) using matlab/simulink software," *Edited by Vasilios N. Katsikis*, vol. 263, 2012.

[100] X. Xian, W. Shi, and H. Huang, "Comparison of omnet++ and other simulator for wsn simulation," in *the 3rd IEEE Conference on Industrial Electronics and Applications*, June 2008, pp. 1439–1443.

[101] G. F. Lucio, M. Paredes-Farrera, E. Jammeh, M. Fleury, and M. J. Reed, "Opnet modeler and ns-2: Comparing the accuracy of network simulators for packet-level analysis using a network testbed," *WSEAS Transactions on Computers*, vol. 2, no. 3, pp. 700–707, 2003.

[102] A. Varga *et al.*, "The omnet++ discrete event simulation system," in *Proceedings of the European simulation multiconference (ESM2001)*, vol. 9, no. S 185. sn, 2001, p. 65.

[103] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st ICST international conference on Simulation tools and techniques for communications, networks and systems & workshops*, 2008, p. 60.

[104] A. Nayyar and R. Singh, "A comprehensive review of simulation tools for wireless sensor networks (wsns)," *Journal of Wireless Networking and Communications*, vol. 5, no. 1, pp. 19–47, 2015.