Sensors **2015**, *15*, 5820-5864; doi:10.3390/s150305820

OPEN ACCESS

sensors

ISSN 1424-8220

www.mdpi.com/journal/sensors

Article

Fortified Anonymous Communication Protocol for Location Privacy in WSN: A Modular Approach

Abdel-Shakour Abuzneid [1,*], Tarek Sobh [1], Miad Faezipour [1], Ausif Mahmood [1] and John James [2]

[1] Computer Science and Engineering Department, University of Bridgeport, Bridgeport, CT 06604, USA; E-Mails: sobh@bridgeport.edu (T.S.); mfaezipo@bridgeport.edu (M.F.); Mahmood@bridgeport.edu (A.M.)

[2] Department of Electrical Engineering & Computer Science, United States Military Academy, West Point, NY 10996, USA; E-Mail: john.james@usma.edu

* Author to whom correspondence should be addressed; E-Mail: abuzneid@bridgeport.edu; Tel.: +1-203-576-4113; Fax: +1-203-576-4401.

Academic Editor: Leonhard M. Reindl

Received: 19 November 2014 / Accepted: 2 March 2015 / Published: 10 March 2015

Abstract: Wireless sensor network (WSN) consists of many hosts called sensors. These sensors can sense a phenomenon (motion, temperature, humidity, average, max, min, *etc.*) and represent what they sense in a form of data. There are many applications for WSNs including object tracking and monitoring where in most of the cases these objects need protection. In these applications, data *privacy* itself might not be as important as the privacy of source location. In addition to the source location privacy, sink location privacy should also be provided. Providing an efficient end-to-end privacy solution would be a challenging task to achieve due to the open nature of the WSN. The key schemes needed for end-to-end location privacy are anonymity, observability, capture likelihood, and safety period. We extend this work to allow for countermeasures against multi-local and global adversaries. We present a network model protected against a sophisticated threat model: passive /active and local/multi-local/global attacks. This work provides a solution for end-to-end anonymity and location privacy as well. We will introduce a framework called fortified anonymous communication (FAC) protocol for WSN.

Keywords: WSN; anonymity; privacy; source location privacy; sink privacy; contextual privacy; routing privacy; temporal privacy; traffic privacy; observability; safety period

1. Introduction

Wireless sensor networks (WSNs) consist of many hosts called sensor nodes (SNs). A wireless sensor device is a simple autonomous host device. It can sense a phenomenon, convert the sensed information into a form of data, process the data and then transmit the data to a sink or a base-station (BS) for further usage or analysis. The sensor host is very limited in terms of storage, cache memory, processing and computing power, communication capabilities and battery lifetime [1–4]. There are many different applications adopting sensor nodes. However, this work focus on monitoring and tracking applications where sensor nodes monitor a certain area and track the presence of a certain object of interest such as an animal in the wildlife, a patient or a doctor in a hospital, or a fellow soldier or a vehicle in the battlefield. When the sensor node senses the object, it reports data to the sink (or to multiple sinks) either directly or through other neighboring sensors. One of the most common applications discussed in source location privacy (SLP) literature is the *panda monitoring game* [4,5]. When a sensor node detects a panda in a certain area, it should report via a message transmitted through intermediate nodes to the sink. In order to protect the panda from hunters or adversaries (ADVs), we need to implement in place an efficient source location privacy scheme (SLP). In such a scenario, location privacy is much more important than the confidentiality of the sensed data itself. Source location privacy is even more important in military, homeland security, and law enforcement, in addition to many other civilian applications [6]. In addition, base-station location privacy (BLP) is very crucial for every WSN since it aggregates all the data.

2. Problem Statement

Privacy in WSN is typically categorized into two categories: *data privacy* and *context privacy* [7–10]. The data privacy includes *data aggregation* and *data query*. The context privacy includes *routing privacy*, *identity privacy*, *location privacy* and *timing privacy*. In this work, we shall focus on using anonymity to provide *location privacy*, which includes two subcategories, source location privacy and base-station location privacy. One of the first works to classify context privacy was done by Kamat *et al.* [11,12], where they addressed the panda hunter game. They claim that the routing scheme is responsible for hiding source location of a subject. They have used two metrics to measure SLP: *Safety period* and *capture likelihood*. Safety period is the number of messages a source sends before it is captured. The capture likelihood is the probability that an adversary can capture the source within a certain period. There are generally two ways to locate a source using passive attacks: *Traffic analysis* [7,13] and *packet tracing* [7,14,15]. The traffic analysis can determine the source or sink locations by analyzing the traffic. Packet tracing can also be used to find the source location since adversaries may use radio-frequency localization techniques to perform a hop-by-hop trace. The adversary can move quickly during packet trace. It could be used to trace mobile nodes due to its fast response compared to traffic analysis [7,14]. We provide a framework that can be tested against other solutions using the following metrics: (i) *Security*: the probability that the adversary successfully identifies the source, the intermediary SNs or the sink; (ii) *energy cost*; (iii) *storage and memory cost*; (iv) *delivery time*; (v) *safety period*: how long it takes the adversary to capture the first sensor node in the network. Our proposed framework provides a modular system that could be configured for a variety

of network models and for a variety of threat models. The rest of this paper is organized as follows: in Section 3, we give some background and literature survey. In Section 4, we will explain the suggested system model, network model, threat model and the traffic model. In Section 5, we will introduce the anonymity module. In Section 6, we will discuss the module of data authentication and integrity. In Section 7, we will discuss temporal privacy. In Section 8, we will have a thorough security analysis. In Section 9, we will have performance analysis and evaluation. In Section 10, we will summarize our work and suggest some additional development to the framework in the future work.

3. Background and Literature Survey

There are many solutions that have been presented to solve the problems of SLP and BLP. Li *et al.* [8] discussed some of the solutions for SLP but they did not aim to create a survey. The comprehensive survey for SLP was presented in the work by Conti *et al*. [4], where they categorized the solutions into eleven groups. They have discussed many solutions and compared them in terms of power consumption, the attack/threat model, view of the network, exposed information, and efficiency in providing SLP. They also discussed some issues that each solution exhibits.

Anonymity is an old issue that was discussed for mobile networks, Ad Hoc networks and Internets. Recently, it has become a concern for WSNs. We have identified solutions for location privacy using anonymity in WSN. We have included them chronologically in Table 1.

An important solution against a global adversary introduced by Chen *et al.* [16] called efficient anonymous communication (EAC) provides sender, link and sink anonymity. We know that anonymity is not enough to achieve fortified end-to-end privacy. There are some solutions based on fake data sources where SNs send out fake packets to other nodes within the network. Some literature call them dummy packets. A fake packet does not contain any real information about any real events but it helps to obfuscate the real traffic and to divert the adversary by mimicking the presence of a fake source. The literature shows reasonable solutions using fake sources. Some of them are designed to handle a local adversary and some of them are suitable for a global adversary. Some of the literature presume a certain routing scheme, topology, network and threat models. Ouyang *et al.* [17] introduced three different solutions to handle the global adversary problem. The first solution is the globally optimal algorithm (GOA). Each SN has a pseudo random number generator that defines the interval time. The second solution by Ouyang *et al.* [17], is the heuristic greedy algorithm (HGA) where SNs follow the same procedure as in GOA except that the SN does not know the complete topology, but it only has the information of its location and the seeds of its neighbors. The third solution by Ouyang *et al.* [17] is the probabilistic algorithm (PBA) where nodes still follow the procedure of HGA, except that they do not send fake messages at the end of every interval. It uses probability p to decide whether to send a fake message or not. The value of p will reduce the communication overhead at the expense of SLP.

We can enhance SLP and BLP by having temporal privacy against the hop-by-hop trace attack or timing analysis attack [4]. There are some literature addressed this using issuing packet delay techniques. Hong *et al.* [18] introduced probabilistic reshaping (*PRESH*) to counter the adversary that uses timing analysis techniques and also introduced and upgraded PRESH to be extended probabilistic reshaping (*exPRESH*) to counter such a scenario. The SN will delay the packet in its buffer again up to D time. Kamat *et al.* [19] introduced rate controlled adaptive delaying (*RCAD*).

Table 1. Solutions for location privacy using anonymity [4]. SAS, Simple Anonymity Scheme; CAS, Cryptographic Anonymity Scheme; HIR, Hashing-Based ID Randomization; RHIR, Reverse HIR; APR, Anonymous Path Routing; ACS, Anonymous Communications Scheme; DCARPS, Destination Controlled Anonymous Routing Protocol for Sensor Nets; MAQ, Max Query Aggregation; PhID, Phantom ID; EAC, Efficient Anonymous Communication.

No.	Scheme	View of the Adversary	Anonymity Technique	Passive Attacks	Active Attacks
1	SAS & CAS [20]	Global	Pseudonyms	Eavesdropping, SN compromise, limited traffic analysis	-
2	HIR & RHIR [21]	Global	Pseudonyms	Eavesdropping, SN compromise	-
3	APR [22]	Local	Pseudonyms	Eavesdropping, hops-tracing	SN compromise
4	DCARPS & Global DCARPS [6]	Global	Pseudonyms	Eavesdropping, hops-tracing	-
5	ACS [23]	Local	Pseudonyms	Rate monitoring, time correlation, identity analysis, hops-trace	-
6	MQA [24]	Global	Aggregation	Eavesdropping, hops-tracing	Packet injection
7	PhID [4,25]	Local	Pseudonyms	Eavesdropping, traffic analysis	-
8	EAC [16]	Global	Pseudonyms	Eavesdropping, traffic analysis	DoS, SN compromise, Traffic injection

In this work, we shall enhance EAC, the efficient anonymous communicating protocol [16,26]. An extension to EAC called *Enhanced Communication Protocol for Anonymity and Location Privacy in WSN* (E^2AC) was presented in [27]. We will call our scheme FAC, *a fortified anonymous communication protocol for WSN*. EAC does not handle the pseudonyms synchronization very well. There are many situations where the system will get unsynchronized. It also could not handle multi-colluding adversaries and lacks a mechanism for time correlation attack. Most of the other solutions do not handle global or multi-colluding adversaries. Each of the different solutions focus on certain selected attack scenarios. Our work is aimed to be comprehensive. We propose a solution against anonymity attacks, temporal attacks, transmission rate-analysis attacks, and statistical attacks, which altogether will provide a fortified source and sink location privacy.

4. System, Network, Threat and Traffic Models

In this work, a framework for end-to-end location privacy using anonymity and temporal privacy is presented. The framework provides the following security elements: Sender anonymity, receiver anonymity, link anonymity, SLP, BLP, data privacy, safety period, and energy preservation. We use sink and BS alternatively throughout this work. The system would be fed with inputs such as the nature of the adversaries in the network, the residual energy in the SNs, the desired lifetime or safety period. We assume bi-directional links where two nodes are considered neighbors if and only if they can hear each other [6].The network considers one sink, which collects/aggregates sensed data (stimuli) from all the SNs. The sink works as an interface for WSN to the wired network [20]. Data packets generated by SNs are ultimately destined uplink to the sink and never destined to another SN. However, it could go through

a multi-hop path. Control packets can be sent from the sink, downlink, to the SNs by unicast or by broadcast messages. To enhance BLP, the sink acts like any other SN in the network while communicating with the SNs to make it absolutely indistinguishable. Most of the literature show that the operation of WSN network goes through two or more phases. However, generally speaking, the WSN runs in three phases: *Pre-deployment phase*, *setup phase*, and *communication phase*. We assume that the SNs have the ability to obfuscate the addresses at the MAC level header [20,28]. All sensors are time synchronized using *time synchronization protocol* [20].

The WSN will need a protocol for *network topology discovery* that allows the sink to view the global topology of the network without revealing the location of the sink [6]. The adversary nodes have very strong capabilities compared to the SNs. They are resource-rich; sufficient energy supply, computation/processing capabilities, and unlimited storage memory. An adversary could run both *passive* and *active* attacks. We presume *Kerchhoff's Principle* [29] for our framework, where the adversary knows everything about the system except the *keys* and *IDs*. The framework will be able to handle both passive and active attacks. We presume that only few compromised nodes could coexist at one time due to the implementation of intrusion detection system (IDS) [16,30–33]. We assume a global adversary, which can monitor the traffic of the entire network and can determine the node responsible for the initial transmission, as in Figure 1.

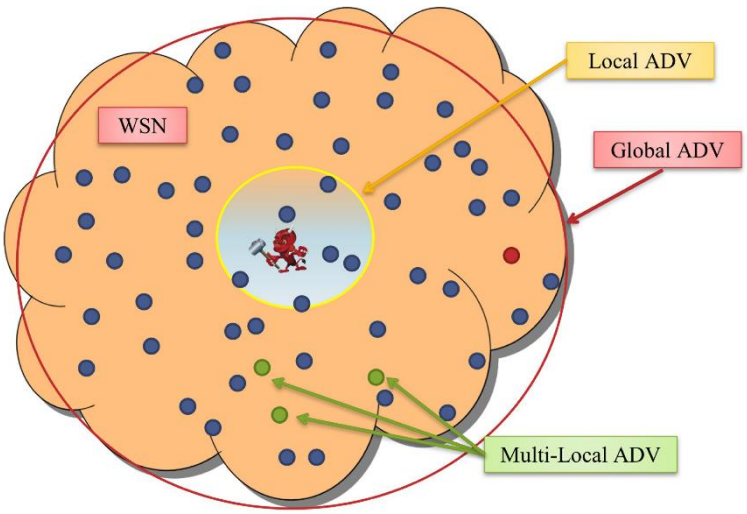


Figure 1. The view of adversary in WSN: local, global and multi-local.

Assuming a global adversary means: (a) the worst-case scenario for area coverage where colluding sensors can cooperate to cover the whole network area [34]; and (b) the worst-case scenario for timing where the coverage area of the adversary is not known to the privacy protocol at any time [34]. We also assume that the adversary is capable of observing transmissions over extended periods. It is, however, not able to break the encryption algorithms or the hash functions used for securing data during transmission. We presume *abundant* traffic where sensors detect and transmit many packets such as in the applications of environment monitoring. Such networks can resist global eavesdroppers easily comparted to *scarce* traffic networks due to the volume of transmissions that could happen at one time. The framework is built of many blocks of *functions* and *protocol*. We have adopted some of the solutions provided in the literature such as solutions for localization and time synchronization. Figure 2 provides a

list of all the blocks that we have provided solutions for and the blocks that we have adopted. The BS will be able to control the network by assigning the value of different parameters.

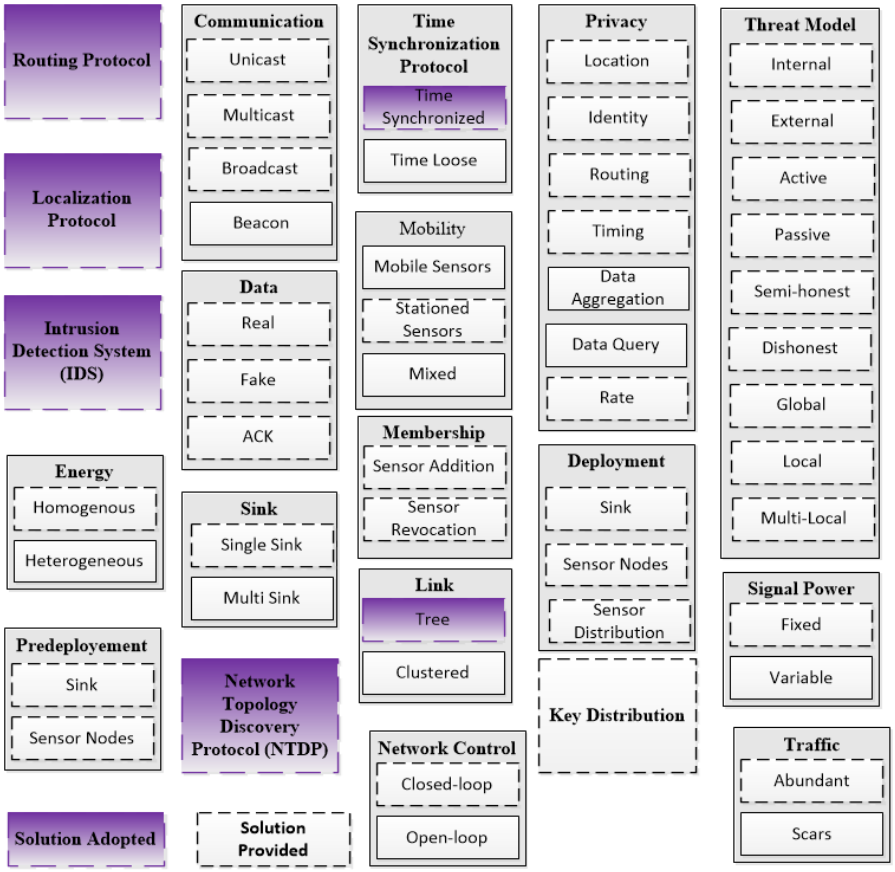


Figure 2. Adopted and provided modules for the framework.

5. Module I: Anonymity

The communication process is divided into three phases, namely: Pre-deployment phase, setup phase and communication phase.

5.1. Pre-Deployment Phase

Prior to actual distribution of the SNs in the field of application, the SNs need to be tested, fully charged, and preloaded with some parameters. We will use subcase letters i and j to describe source and intermediary nodes consecutively. We will use *BS* to describe the sink or the base station. Table 2 summarizes all the parameters and terms used in this work.

Table 2. Reference of important parameters and terms used by FAC.

Notation	Definition	Source
ID_i	ID of sensor i	Preloaded
a_i	Random number shared between SN_i & BS	Preloaded
b_i	Random number shared between SN_i & neighbors	Preloaded
c_i	Random number shared between SN_i & neighbors	Preloaded

Table 2. *Cont.*

Notation	Definition	Source
H1	Hash function to create pseudonyms and the keys	Preloaded
H	Hash function to create data digest	Preloaded
$k_{i\leftrightarrow bs}$	Pair-wise key shared between SN_i & BS	Preloaded
kb_i	Broadcast key for SN_i	Preloaded
fkb_i	Fake broadcast key for SN_i	Preloaded
N	Number of SNs in WSN	Learned
N_i	Number of neighboring for SN_i	Calculated
$HC_{i\leftrightarrow bs}$	Hop-count between SN_i & BS	Learned
PID_i	Pseudonym ID shared between SN_i & BS	Calculated
$BPID_i$	Broadcast pseudonym ID	Calculated
$a_{i\leftrightarrow j}$	Random value shared between SN_i & SN_j	Calculated
$k_{i\leftrightarrow j}$	Pair-wise key shared between SN_i & SN_j	Calculated
$OHPID_{i\leftrightarrow j}$	Pseudonym ID shared between SN_i & SN_j	Calculated
$APID_i$	ACK pseudonym ID for SN_i	Calculated
$FBPID_i$	Fake broadcast pseudonym ID	Calculated
T_i	Table in SN_i for shared parameters	Calculated
TIME_STAMP	Time stamp	Calculated
SEQ_NO	Sequence number for a message	Calculated
TTL	Time to live	Calculated
MCG_LGTH	Message size	Calculated
$\boldsymbol{\Delta_{residual}}$	Residual energy	Calculated
$\oplus$	XOR Operation	Operation
$\|\|$	Concatenation operation	Operation

5.2. Setup Phase

It is typical to presume the WSN is considered secure for some short period after the deployment of sensors and before the steady communication phase. Zhu *et al.* [35] presented that WSN has a lower bound on the time interval (T_{min}) before the adversary is able to compromise a SN. During this time, the sensors can communicate and exchange all needed information safely. The sink needs to know the location of all the SNs participating in the WSN. Likewise, the SNs need to know their relative locations to the sink and to their neighbors. There are many *localization schemes* which, are proposed in the literature [16,20,36,37]. We presume the network will adopt one of the available efficient localization schemes. Localization allows each SN to know its smallest *hop-count* to the BS ($HC_{i\leftrightarrow bs}$).

5.2.1. Creating Pseudonyms

The key idea is to use pseudonyms instead of using real IDs for the SNs and the BS during communication. Therefore, one disposable pseudonym per one transmission is used. This way, the ADV cannot trace back to the source using multiple messages containing the real ID. There are five kinds of transmissions that could happen in the WSN: (i) Multi- hop transmission between SN_i and BS; (ii) transmission between two sensor neighbors i and j; (iii) broadcast sent by SN_i or BS; (iv) acknowledgement; and (v) fake broadcast. The process starts by creating a pseudonym ID for each SN_i, we call it for short (PID_i) which is computed using Equation (1):

$$PID_i = H_1(ID_i \oplus a_i) \quad (1)$$

The SN_i can calculate the broadcast pseudonym ID ($BPID_i$) according to Equation (2):

$$BPID_i = H_1(ID_i \oplus b_i) \quad (2)$$

The SN_i can calculate the fake broadcast pseudonym ID ($FBPID_i$) according to Equation (3):

$$FBPID_i = H_1(ID_i \oplus c_i) \quad (3)$$

SN_i should, by now, know its entire neighbor set (N_i). SN_i will send a broadcast discovery message ($M_{discovery}$), to exchange parameters with all one-hop neighbors. The format of the message is stated in Equation (4):

$$M_{discovery} = K_{dis}(TTL \,||\, ID_i \,||\, k_{i\leftrightarrow bs} \,||\, kb_i \,||\, fkb_i \,||\, a_i \,||\, b_i \,||\, c_i \,||\, \Delta_i \,||\, HC_{i\leftrightarrow bs}) \quad (4)$$

where TTL should be 1 for this transmission. K_{dis} is a shared common encryption key to secure the discovery message. SN_i will receive also a similar broadcast message from SN_j and from all other neighbors. Both SN_i and SN_j will calculate a new random value ($a_{i\leftrightarrow j}$) according to Equation (5):

$$a_{i\leftrightarrow j} = H_1(ID_i \oplus ID_j) \quad (5)$$

Both SN_i and SN_j will calculate also a new pair-wise key $k_{i\leftrightarrow j}$ according to Equation (6):

$$k_{i\leftrightarrow j} = H_1(k_{i\leftrightarrow bs} \oplus k_{j\leftrightarrow bs}) \quad (6)$$

SN_i also calculates broadcast pseudonym ID for SN_j ($BPID_j$) according to expression Equation (2) since SN_i has already received the values of ID_j and b_j through $M_{discovery}$. It also calculates the one-hop pseudonym ID ($OHPID_{i\leftrightarrow j}$) shared between SN_i and SN_j as expressed in Equation (7):

$$OHPID_{i\leftrightarrow j} = H_1(a_i \oplus a_j) \quad (7)$$

Finally, acknowledgement pseudonym ID for SN_i ($APID_i$) will be calculated according to Equation (8):

$$APID_i = H_1(ID_i) \quad (8)$$

SN_i will create a table (T_i) which contains the shared values with the neighbors as listed in Table 3. In conclusion, we have replaced the ID with *quintuple pseudonyms* to reference the SN during the communication.

Table 3. Shared values among sensor neighbors. If SN_i has N_i neighbors, then T_i will have N_i tuples.

Information in T_i Per Each Neighbor	Tuple for SN_j
Shared random number	$a_{i\leftrightarrow j}$
Shared broadcast random number	b_j
Shared fake broadcast random number	c_j
Shared broadcast key	$BPID_j$
Shared fake broadcast key	$FPID_j$
Shared one-hop key	$k_{i\leftrightarrow j}$
Current one-hop pseudonym ID	$OHPID_{i\leftrightarrow j}$
Link direction	$link_{i\rightarrow j}$
Residual energy level	Δ_j

5.2.2. Deleting Security Information

After storing all required pseudonyms, parameters and keys in T_i, it would be the time to delete all unnecessary information from SN_i memory for the purpose of security [27]. In addition, it will release some memory storage space [16,26]. Most importantly, SN_i will delete ID_i and $HC_{i\leftrightarrow bs}$, which could be critical information for the adversary. In addition, SN_i shall delete all discovery messages.

5.3. Communication Phase

During the communication phase, when sensing and sending data to the BS takes place, there are seven operations that continue until network lifetime ends. These operations are: (i) Sense and send a message to a neighbor; (ii) forward a message hop-by-hop; (iii) broadcast a real message; (iv) acknowledgement; (v) broadcast a fake message; (vi) SN removal; and (vii) SN addition. A SN will have three roles, in terms of data transmission, during the communication phase: (i) Role as a sensor; (ii) as a message forwarder; and (iii) as a broadcaster. In the following sections, we will use SN_i as a source node and SN_j as a neighbor to the source.

5.3.1. Transmission as a Sensor

When SN_i senses data, it needs to send a message hop-by-hop to the BS. The SN_i only recognizes itself by its (PID_i), and the BS will recognize the source of the message by its PID_i as well. Thus, the PID_i of the source needs to be included in the message until the BS receives it. Consequently, the PID of a sensor will be updated after every transmission. The SN_i needs to select one neighbor from N_i to forward the message to it. The selection process goes through a probabilistic protocol, which guarantees that SN_i does not use one neighbor all the time when forwarding its data; first, for routing privacy, and second for increasing the lifetime of the WSN. SN_i will form the message in the following format:

$$M_{i\rightarrow j} = OHPID_{i\leftrightarrow j} \,||\, E_{ki\rightarrow j}\,(PID_i \,||\, E_{ki\leftrightarrow bs}\,(D_i)) \quad (9)$$

where D_i includes the sensed data. Once SN_i knows that the message ($M_{i\rightarrow j}$) is delivered to the the neighbor, it needs to dispose of the current pseudonym PID_i and issue a new one for the next transmission as indicated in Equation (10):

$$PID_i = H_1(PID_i \oplus a_i) \quad (10)$$

In addition, both SN_i and SN_j will dispose of the current $OHPID_{i\rightarrow j}$ and issue a new one for the next communication between the two neighbors according to Equation (11):

$$OHPID_{i\rightarrow j} = H_1(OHPID_{i\rightarrow j} \oplus a_{i\rightarrow j}) \quad (11)$$

The message (M) will then be reformatted by the recipient SN_j and again forwarded to the next node, say SN_r, and so on, until it gets to the BS. If SN_j was the BS, then the BS uses the shared one-hop key between the sensor and the BS, to decrypt the data and to get the PID_i, which the BS can use to recognize the source SN_i. Only at this point of time, BS can update the value of PID_i of SN_i. It also reads the data (D_i) which the BS can decrypt using $k_{i\leftrightarrow bs}$.

5.3.2. Transmission as a Forwarder

When SN_i sends the message one-hop uplink to the neighbor SN_j, then SN_j needs to forward the message to another intermediary node. Upon receiving $M_{i\to j}$, SN_j will match $OHPID_{i\to j}$ in its table, T_j. If there is no match, then the message definitely is not addressed for SN_j and it will be dropped immediately. If it matches, then the message is decrypted using $k_{i\to j}$. The message will be forwarded to SN_r after (M) is reformatted as in Equation (12):

$$M_{j\to r} = OHPID_{j\leftrightarrow r} \,||\, E_{kj\to r}\,(PID_i \,||\, E_{ki\leftrightarrow bs}\,(D_i)) \quad (12)$$

Right after the data is *received* by SN_j and *forwarded* to the next one-hop SN_r, the SN_j updates the pseudonym $OHPID_{i\leftrightarrow j}$. SN_j now is ready to exchange another message with SN_i using the new pseudonym $OHPID_{i\leftrightarrow j}$. However, SN_j is not yet ready to send data to SN_r since SN_r does not update the $OHPID_{j\leftrightarrow r}$ until (D_i) is forwarded to the next hop, say NS_v. See Figure 3 for the sequence of transmissions for a message from SN_i to the BS.

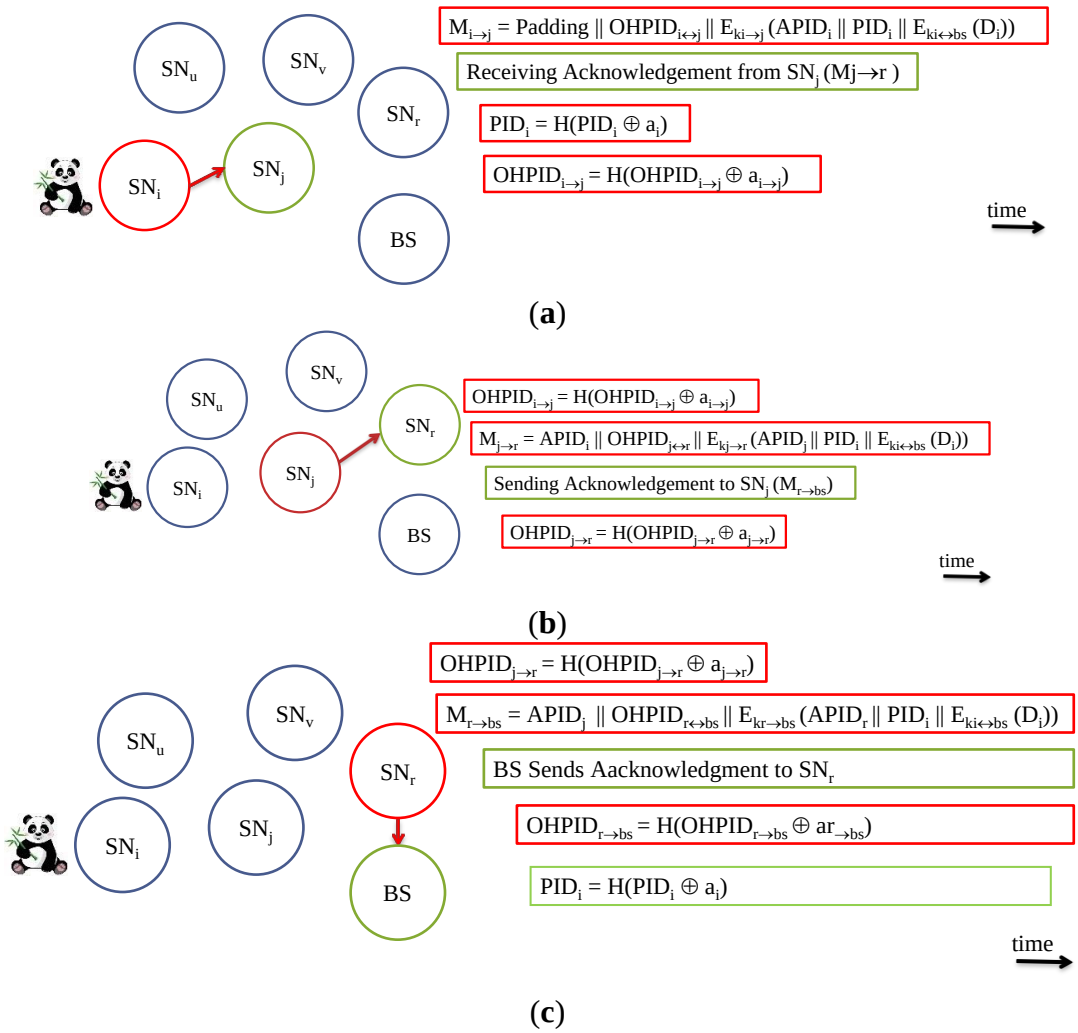


Figure 3. The sequence of a message transmission from SN_i to the BS. (**a**) SN_j receives a message from SN_i; (**b**) SN_j forwards the message to a neighbor SN_r; (**c**) BS receives the message and processes it.

5.3.3. Acknowledgement

As expected in data networks, message could be lost or could be corrupted. In either case, retransmission is required. Because SNs change PIDs after each transmission, synchronizing PIDs is crucial. Updating the pseudonyms depends on successful message delivery. Ideally, the source should update the pseudonyms only after making sure the BS receives the data. However, the lack of direct connection between the source and the BS makes it a bit complicated process.

The BS cannot send direct acknowledgement to the source if it is multiple hops away. We have to depend on multiple acknowledgements along the path between the source and the BS. SN_i needs to calculate the Acknowledgement pseudonym ID ($APID_i$) according to Equation (13):

$$APID_i = H1(APID_i \oplus b_i) \quad (13)$$

The message will be sent out to the neighbor with the current value for $APID_i$. Thus, we will rewrite $M_{i\to j}$ as it appears in Equation (14):

$$M_{i\to j} = Padding \,||\, OHPID_{i\leftrightarrow j} \,||\, E_{ki\to j}\,(APID_i \,||\, PID_i \,||\, E_{ki\leftrightarrow bs}\,(D_i)) \quad (14)$$

Padding is added to make sure all the one-hop messages have the same size to prevent *size correlation* attacks. When SN_j receives the message, it will reformat the message as in expression Equation (15) and then send it to SN_r:

$$M_{j\to r} = APID_i \,||\, OHPID_{j\leftrightarrow r} \,||\, E_{j\to r}\,(APID_j \,||\, PID_i \,||\, E_{ki\leftrightarrow bs}\,(D_i)) \quad (15)$$

The transmission of $M_{j\to r}$ should be heard by all the neighbors including both SN_i and SN_r. If SN_i hears the message and reads ($APID_i$), the SN_i knows that $M_{i\to j}$ was received correctly by SN_j. Only at this time, SN_i updates the value of $OHPID_{i\leftrightarrow j}$. PID_i will get updated, as well, since SN_i is the source of the message. This is exhibited in Figure 4. Here are two scenarios:

Scenario 1: The packet sent by SN_i is lost or got corrupted. In this case, SN_j considers nothing happened, so it will not forward any message onward. Meanwhile, SN_i will wait for (ζ) time to expire. It will send the message again with updated $APID_i$. Once the message is acknowledged according to the procedure explained earlier, then PID_i, $OHPID_i$ and $APID_i$ will be updated. If it is intermediary SN, only $OHPID_i$ and $APID_i$ is updated as exhibited in Figure 5.

Scenario 2: SNj receives the packet correctly; the new packet $M_{j\to r}$ is sent out which contains the acknowledgement ($APID_i$), and SN_j updated the value of $OHPID_{i\leftrightarrow j}$. However, SN_i does not hear the forwarded message $M_{j\to r}$ within time (ζ). At this moment SN_i does not know for sure if the message was delivered (*resembles scenario 1*), or the acknowledgement is lost. It has to account for the worst case. A copy of the message will be retransmitted to SN_j with the current $OHPID_i$ and updated $APID_i$. SN_j can recognize the message because of the value of old $OHPID_i$. After receiving the *retransmitted* message, it now sends a direct acknowledgement to SN_i as in Equation (16).

$$ACK_{i\leftarrow j} = APID_i \,||\, Padding \quad (16)$$

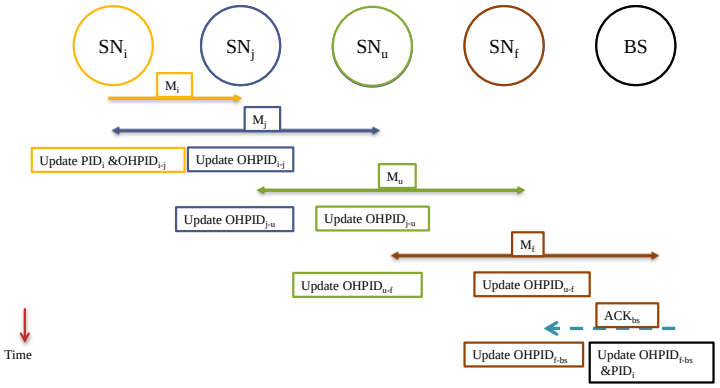


Figure 4. Using APID$_i$ for acknowledgement with no errors.

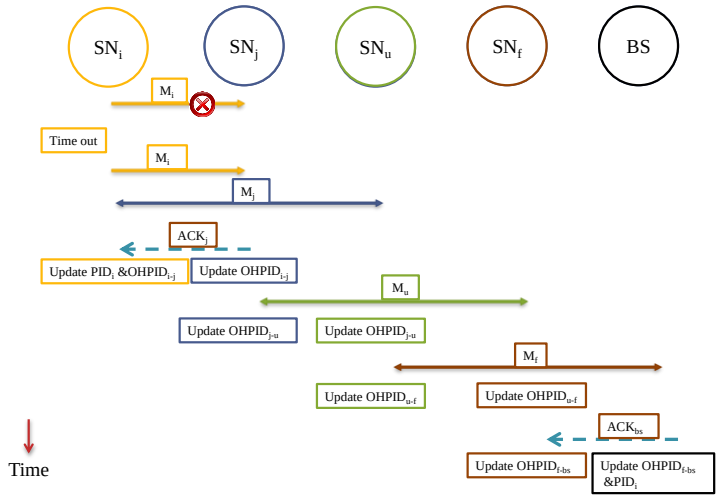


Figure 5. Acknowledgement for a message with errors.

Figure 6 shows the process. BS is treated similar to a normal SN, so it has to acknowledge every message it receives. After the message is delivered to the BS, and after the message is acknowledged, the PID$_i$ (of the source) will be updated on the BS tables while it has been already updated in the sensor itself after the first acknowledgement.

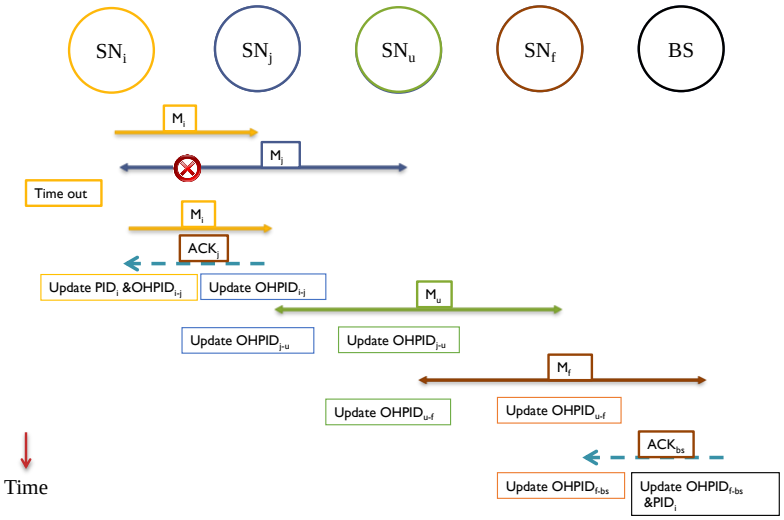


Figure 6. Handling lost acknowledgement.

Both the SN_i and the BS will be ready to exchange a new message. As long the new message does not reach to the BS before the old PID_i is updated, the system will remain synchronized. This way, we have a possible window of one message only. We propose implementing a *sliding window* mechanism as exhibited in Figure 7 [27]. For each sensor, we can have a window of (W) slots.

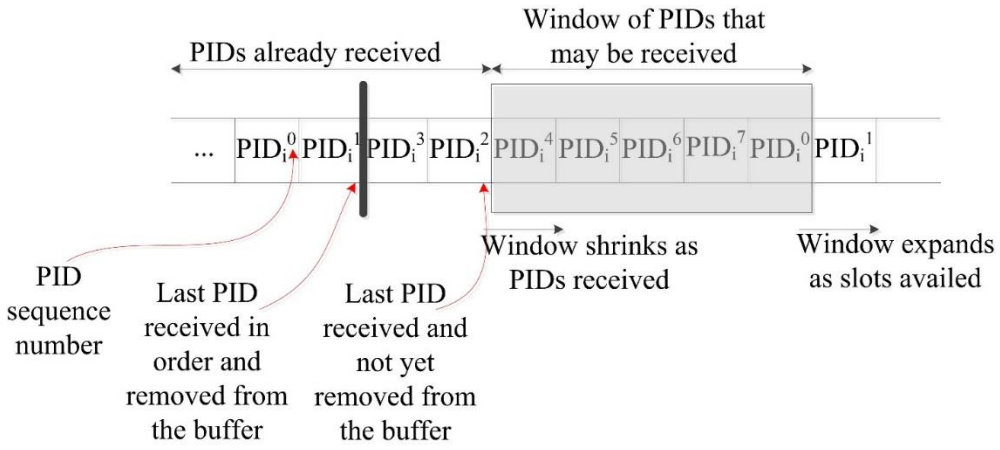


Figure 7. Sliding window for received PIDs [27].

5.3.4. Transmission as a Broadcaster

Typically, the BS is required to broadcast a message for control and management purposes. Likewise, a sensor might need to broadcast a message to the BS or to the neighbors for network setup, maintenance and other management issues. The framework requires keeping all the messages indistinguishable throughout the network, so all the messages need to have the same size. Each SN is preloaded with a broadcast key (kb_i) and assigned broadcast pseudonym ($BPID_i$). The broadcast message sent by SN_i is formatted as in Equation (17):

$$M_b = \text{Padding} \,||\, BPID_i \,||\, E_{kbi}(D_b) \quad (17)$$

All the neighbors will receive the broadcast message from a source SNi. SN_i and the recipients will update $BPID_i$ according to Equation (18).

$$BPID_i = H_1(BPID_i \oplus b_i) \quad (18)$$

Upon receiving the broadcast message (M_b), SN_j decrypts the message using (kb_i) stored in the table (T_j). It then encrypts it again using (kb_j) and broadcasts (M_b) to its one-hop neighbors set (N_j) as in Equation (19):

$$M_b = BPID_i \,||\, BPID_j \,||\, E_{kbj}(D_b) \quad (19)$$

When the BS receives a broadcast message, it is ultimately the destination, so intuitively it does not need to broadcast the message again. Our proposed framework assumes that the BS behaves similar to a normal sensor. To maintain this pre-course, we require the BS to broadcast the message again for acknowledgement purpose. Thus, we introduce the limited broadcast where the BS will be able to broadcast to only one hop (TTL = 1).

5.3.5. Limited Broadcast Messages

A sensor inside network maze can only recognize the neighboring sensors and the BS. When SN broadcasts a message uplink (towards the BS), then all neighbors should hear it. The neighbor should broadcast the message again if and only if the message comes from a SN with a bigger hop-count (HC). This will conserve a lot of unnecessary traffic and energy dissipation. The broadcast message will contain (TTL = HC). The value will keep decreasing by one until it gets to the BS. In contrast, the downlink broadcast messages (by the BS to the SNs) should have (TTL = 0) where the intermediary sensors would rebroadcast the message if and only if it comes from a neighbor with a smaller (HC). A special case when (TTL = 1) where the message will be broadcasted to one-hop neighbors only. FAC also may adopted a more sophisticated optimized flooding algorithm for wireless multi-hop network, such as CDS-based algorithms [38,39].

5.3.6. Fake Broadcast Message

The sensors need to send fake messages to prevent *time correlation*, *rate analysis* and *statistical analysis*. A fake message is technically a one-hop broadcast message. However, to prevent correlation, the message needs to behave similar to real messages. Therefore, the message needs to be encrypted and have similar size as the real message to make it completely *indistinguishable*. Since it has to carry a dummy data, it will contain the *residual energy* (Δ) of the issuing sensor. This information will be extracted by the recipient neighbors and saved in the related tuple in the table (T). The fake broadcast message sent by SN_i is as in Equation (20):

$$M_f = \text{Padding} \,||\, FPID_i \,||\, E_{kfi}(\Delta_i) \quad (20)$$

All the neighbors will receive the fake broadcast message from SNi. SN_i and the recipients will then update $FPID_i$ according to Equation (21):

$$FPID_i = H_1(FPID_i \oplus c_i) \quad (21)$$

There is no need to worry about the pseudonyms synchronization since the main purpose of the fake messages is to show activity in idle sensors to obfuscate real messages.

5.4. SN Removal

There are many reasons why a sensor should be removed from WSN. For instance, when the battery of a sensor is about to deplete, it should refrain from participation. This would protect against data loss and maintain the pseudonyms synchronized. In some other cases, WSN use IDS [40,41] to protect against active attacks, so once a sensor is captured, it must be banished from the network. Procedurally, if SN_i opts to be removed, it will send a message to the BS as in Equation (22):

$$M_{i\rightarrow j} = OHPID_{i\leftrightarrow j} \,||\, E_{ki\rightarrow j}\,(PID_i \,||\, E_{ki\leftrightarrow bs}\,(D_{remove})) \quad (22)$$

where (D_{remove}) is a command to banish the sensor. The tuple of the SN_i in the BS tables will be disabled permanently. In addition, SN_i will send a broadcast message to the neighbors as in Equation (23):

$$M_b = \text{Padding} \,||\, BPID_i \,||\, E_{kbi}(D_{remove}) \quad (23)$$

Once the neighbors get the message (D_{remove}), they will delete the tuple related to SN_i from the table (T) and banish the sensor. The BS for sensor removal could use the same process.

5.5. SN Addition

To add a new sensor to the network, the sensor will be preloaded with the required parameters: ID_i, a_i, b_i, c_i, H1, $k_{i\leftrightarrow bs}$ and kb_i, and fkb_i. Right after deployment, the sensor calculates the shared parameters with its neighbors. The BS should be trusted to run the process. The BS will send special key (k_{add}) to all the neighbors. SN_i will be preloaded with the same key as well. SN_i and the neighbors will use this special key to authenticate with each other. Initially, the BS sends the following message to the one-hop neighbors of the new sensor as in Equation (24):

$$M_b = \text{Padding} \,||\, BPID_{bs} \,||\, E_{kb\text{-}bs}(D_{add}) \quad (24)$$

where (D_{add}) is expressed in Equation (25):

$$D_{add} = hc \,||\, k_{add} \quad (25)$$

The initial value for hc is *zero*. It will be incremented every time the message is forwarded.

5.6. Contribution of Anonymity Module

Other works have provided anonymity using pseudonyms and aggregation to provide SN anonymity while very few provided BS anonymity. Our anonymity module has contributed with an innovative approach by using 100% anonymous communication. We have provided to have anonymous real, fake, acknowledgment, unicast and broadcast message transmission. Moreover, we have provided anonymous transmission for the BS by providing limited onion encryption. Compromising a SN in some other works would lead to the discovery of the pseudonyms, which are, related the SN, which could help the adversary to carry further attacks. In our module, capturing a SN will not lead to pseudonyms' leakage. The module will fight against local, multi-local and global adversary. Although, some solutions claimed fighting global anonymity, keeping the pseudonyms synchronized was not possible. We have provided a complete mechanism for synchronization, secure sensor addition and removal. The module will fight both passive and active attacks. A complete anonymity and security analysis is be provided in Section 8. Section 9, explains how the solution remains light compared to the other works.

6. Module II: Data Authentication and Integrity

The data is encrypted before transmission to protect against passive attacks such as eavesdropping. For active attacks, such as data and transaction falsification, message authentication is required. The two important security aspects to achieve: (i) Verify that the content of the message is not altered and; (ii) the source is authentic. We could achieve authentication by either using a message authentication code (*MAC*), or one-way hash function (*OWH*). MAC would require the sender (SN_i) and receiver (BS) to share a secret key. The authentication code is calculated as *MAC = F (k, D)*. *DES* or other algorithms can be used to generate the code. The OWH also accepts a variable size message (*D*) as input and produces a fixed sized digest *MD = H (D)* as output. Examples for OWH are *SHA*, *MD5*, *Whirlpool and* HMAC. The advantage of OWH over MAC is the fact that it does not use encryption, which is quite

slow. Comes in the middle, HMAC which is a MAC derived from OWH such as *SHA-1*. It could be expressed as: MD = HMAC (K, D).

If we opt to use HMAC as an example, the ($M_{i\rightarrow j}$) will be rewritten as in Equation (26):

$$M_{i\rightarrow j} = APID_i \,||\, OHPID_{i\leftrightarrow j} \,||\, E_{ki\rightarrow j}\,(APID_i \,||\, PID_i \,||\, E_{ki\leftrightarrow bs}\,(D_i)) \,||\, \mathbf{HMAC_{ki\leftrightarrow bs}\,(PID_i \,||\, D_i)} \quad (26)$$

The key ($k_{i\leftrightarrow bs}$) is shared between SN_i and the BS. The message could be authenticated with MD using OWH as in Equation (27):

$$M_{i\rightarrow j} = APID_i \,||\, OHPID_{i\leftrightarrow j} \,||\, E_{ki\rightarrow j}\,(APID_i \,||\, PID_i \,||\, E_{ki\leftrightarrow bs}\,(D_i) \,||\, \mathbf{H(PID_i \,||\, E_{ki\leftrightarrow bs}\,(D_i))}) \quad (27)$$

As it is transparent from expression Equation (27), we need more processing time and therefore more power consumption because we have encrypted a sizable packet. There is a tradeoff between higher security and energy conservation. The first approach is more appropriate. Authentication for the broadcast messages is done as in Equation (28):

$$M_b = Padding \,||\, BPID_i \,||\, E_{kbi}(D_b) \,||\, HMAC_{kbi}(D_b)) \quad (28)$$

Alternatively, it can be achieved using Equation (29):

$$M_b = Padding \,||\, BPID_i \,||\, E_{kbi}(D_b \,||\, H(kb_i \,||\, D_b)) \quad (29)$$

The message could contain other important information such as *sequence number* (similar to the well-known *HDLC* and *TCP* protocols) and *time stamp*. The receiver uses the sequence number to verify the order of messages. Time stamp is used to check the delay threshold. Both checks will enhance protection against various active attacks. The message core data (D_i) could have the following format:

$$D_i = SEQ\_NO \,||\, TIME\_STAMP \,||\, MSG\_LGTH \,||\, SENSED\_DATA \quad (30)$$

Providing authentication to protect against active attacks is crucial in any communication. The innovation of our authentication module is by providing message authentication for every transmission in the network without limiting it to real messages unlike many other works proposed. The adversary can utilize any captured transmission to launch attacks against the network, which could include real, fake and acknowledgement messages. Our module can use MAC, OWHF and HMAC according to the security needs of the WSN. The network can adjust the parameters according to the security situation using adaptive framework. Integrating the authentication module with the anonymity module without hindering the performance of either one is a necessity, which we have achieved in this work.

7. Module III: Temporal Privacy

WSN could suffer from time correlation attacks [11,13,14,23,42] by observing the time between correlative packets sent and received in a certain neighborhood. The adversary can trace forward and backward the messages until they reach to the BS or to the source. Hence, hiding temporal information is crucial for both anonymity and location privacy. Using routing schemes to protect against time correlation attacks is found to be efficient to certain extent where local adversary usually has limited mobility and partial view of the network traffic. However, routing based schemes do not work for global adversary where the traffic of the whole network can be easily monitored with a full spatial view and the adversaries can collude together to promptly detect the origin and time information of the event [18,34]. A mechanism is required to divert attention of the adversary when there is event-driven transmissions,

especially with the presence of global adversary [43]. Figure 8 exhibits a probabilistic distribution for the fake messages.

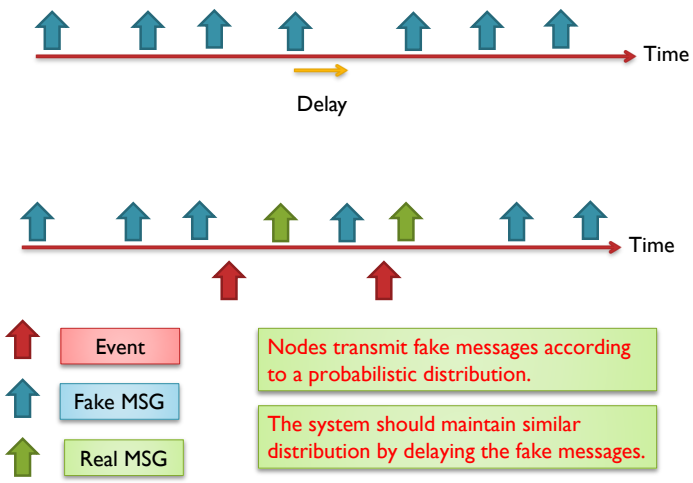


Figure 8. Nodes transmit fake messages according to a probabilistic distribution. When real messages are sent, the system should maintain the required distribution by delaying some fake messages [34,44].

The distribution of events changes which could be a reason for the adversary to detect the event timing and thereafter the source of the event. The message distribution (both real and fake) needs to be adjusted to prevent time correlation. In some applications, such as monitoring and surveillance, we cannot guarantee a certain event distribution. The literature talk about three ways to maintain an obfuscated message distribution: (i) By issuing message delays; and (ii) by issuing fake messages; and (ii) by using both delays and fake messages. Using delays works well against local adversary but might not be suitable for time sensitive networks. In contrast, using fake messages is required to protect against multi-local and global adversary, however, it is very expensive in terms of energy dissipation. Furthermore, adversary with good statistical analysis can easily detect the message distribution if the scheme is not designed carefully [34]. Some work in the literature clearly differentiates between two terms: the *event* (of transmission) and the *interval* (of transmission). If every interval has only one transmission, then event and interval are the same, however, this might not be the case when we have multiple transmissions during one interval. So, the anonymity level depends on the capability of the adversary to distinguish between real and fake transmissions. This means, given multiple transmissions by a SN, the adversary must be unable to distinguish, with significant confidence, between transmissions carry real data and transmissions carry fake data. Alomair *et al.* [34] suggested that transmission "indistinguishability" is not enough. They claim that indistinguishability is achieved when adversary monitoring the network over multiple time intervals, in which some intervals contain real event transmissions and others do not, is unable to determine, with significant confidence, which of the intervals contain the real traffic. If intervals are indistinguishable, the individual transmissions within the interval should also be indistinguishable.

We should have a mechanism to quantify anonymity while it is used, in the literature, in different ways. However, in our work, anonymity means how to prevent the adversary from knowing the source of the message. In other words, the adversary could know that a particular sensor sent a message at one time, but it should not know that sensor is the source of the message. By delaying the real messages and

by issuing multiple messages at one interval would mislead the adversary. As an example, for one transmission and one adversary, where the adversary can guess either the message is real or fake without any anonymity measurement taken, it should be 0.5 (either fake or real). Let us presume ψ donates one adversary strategy for breaching the anonymity of the system among a set of strategies. Let us presume P_r is the probability that the adversary succeeds using strategyψ. The anonymity A as defined in [34] with the existence of a strategyψ, is presented in Equation (31):

$$A_\psi: = 1 - P_r, \text{ where } 0 \leq P_r \leq 1 \quad (31)$$

If we presume that Σ represents all possible strategies for the adversary to breach the anonymity of the WSN, the accumulated anonymity will be as in Equation (32):

$$A: = \min (A_\psi), \text{ where } \psi \in \Sigma \quad (32)$$

It is very important to increase anonymity for every individual SN in the network especially with the presence of multi-local or global adversaries. Presence of colluding adversaries could cause the anonymity to drop exponentially [34]. Take Figure 9 as an example, where WSN has a moving Panda from point "a", to "b", to "c", then finally to "d" where each location has a SN to report the Panda's movement. If the anonymity of each sensor is A = 0.8, then the anonymity at node "b" is $A = 0.8^2 = 0.64$ and at point "d" is $A = 0.8^4 = 0.41$. Having global adversary makes it super necessary to design a strong anonymity model which can resist the time correlation attack [42].

In this work, we assume the worst case for time correlation attacks which is a global or laptop-class adversary attacks [17]. Having an anonymity scheme to protect against the global adversary will be very expensive solution in terms of energy preservation and thus the lifetime of the network. In the following two subsections, we propose two schemes, the simple global anti temporal (SGAT) and the energy controlled anti temporal (ECAT).

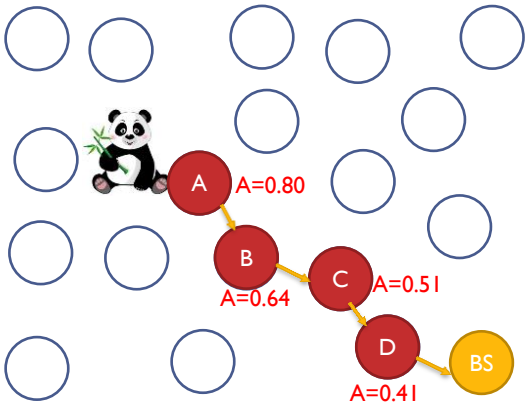


Figure 9. Having multiple colluding nodes will reduce system anonymity exponentially [34,42,44].

7.1. Simple Global Anti Temporal Scheme (SGAT)

When an event-driven message is sent out, the adversary can trace back the message to the SN or forward to the BS. Sending few other transmissions in the network within the range of the adversary confuses it and prevents the adversary from having known path to follow. In this work, we presume the lifetime of the network (Ω) is divided into a number of intervals (*I*) and each interval time is (ω), where:

$$\Omega = I \times \omega_i \quad (33)$$

The value of Ω can be predicted as a range between a minimum value (worst case) Ω_{min} and a maximum value (best case) Ω_{max}. It all depends on how real/fake transmissions are facilitated. The SNs will send either a fake or a real message during one interval. The message is sent at the end of each interval or it is adjusted to be sent during the interval to create some variable delays through the route to the BS, which would confuse the adversary more and would prevent it from gaining useful knowledge about the network based on time correlation. SN_i that has sensed the event or received the real data from another SN_j, will send the real message (M_r) through a hop-by-hop path to the BS, and some other nodes will send fake messages (M_f) during the same time to disrupt the adversary. There are two questions: *How long the message will be held in the SN after it is sensed?* Simplistically, M_r and M_f are sent at the end of the interval *I*. The time from arrival of the data to the end of the period time (τ_w) expressed in Equation (34):

$$\tau_w = \omega_i - t_a \quad \text{where: } t_0 \leq t_a \leq t_s \leq \omega_i \quad (34)$$

where: t_0 is the beginning of the interval I_i, t_a is the arrival time, $t_0 \leq t_a$.

Ideally, the message will be sent immediately after it is sensed or received which makes $\tau_w = 0$. Theoretically, τ_w could be a value: $0 \leq \tau_w \leq \omega_i$ as exhibited in Figure 10.

How many SNs in the network will send fake message during one interval time and which ones? Simplistically, every SN in the network, which is in the range of the adversary and in the range of source SN, should send a fake message while SNs that have real messages will send the real messages only.

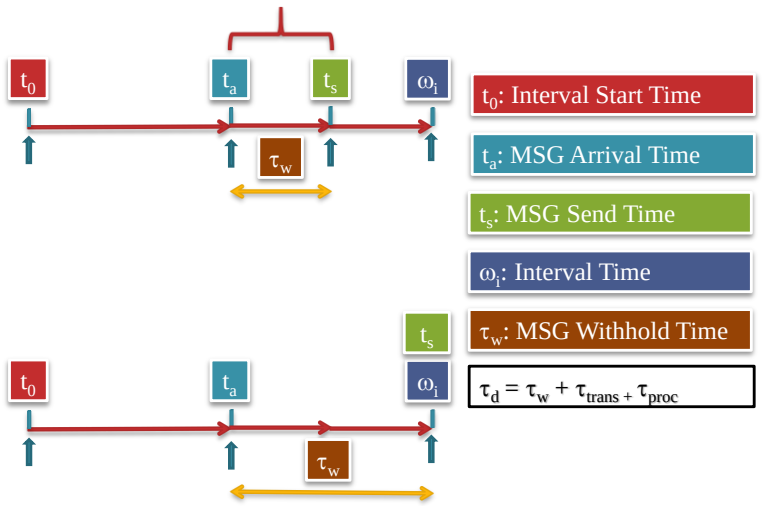


Figure 10. Timing for receiving a real message and then sending it out during the interval period assigned to the sensor node. The total delay will include the processing time, transmission time and the withhold time [42].

There are many technical issues regarding the determination of the optimal configuration for both questions mentioned earlier. For instance, it is not possible for the neighboring nodes to know in advance when a SN is going to sense an event. It is a completely unpredictable random-distribution for the events. The need to transmit fake messages becomes even much more crucial if we do not have busy-network. Therefore, all SNs with no real messages need to send fake messages during the interval I_i, in the worst

case, or only selected nodes according to a probabilistic protocol. Having high number of fake message transmissions will reduce the lifetime of the network in favor of privacy. Doing the reverse will jeopardize the privacy of the sensor nodes. The adversary could learn the mechanism of sending real and fake messages at the end of the interval. However, it is not very dangerous if the network sends enough fake messages at the same time. Having variable withhold time (τ_w) is useful for privacy and for reducing the average network delay. The delivery time (τ_d) presuming that the message is always sent at the end of the interval I_i is:

$$\tau_d = \tau_w + \tau_{trans} + \tau_{proc} \quad (35)$$

where: τ_d is delivery time, τ_{trans} is transmission time, τ_{proc} is processing time.

If we presume τ_{proc} is much smaller than τ_{trans}, then τ_d can be rewritten as the following:

$$\tau_d = \tau_w + \tau_{trans} \quad (36)$$

If the message needs to go through (U) hops to the BS, and if we assume that the transmission only happens at the end of the Interval I_i, then t_s, equals to ω_i, and the total delivery time ($\tau_{d\text{-}total}$) can be calculated according to the expression below [42]:

$$\tau_{d-total} = \sum_{u=1}^{U} \tau_{w_u} + \tau_{trans_u} \quad (37)$$

Having t_s equals to ω_i; *i.e.*, sending message at the end of the interval, will increase the delay of the delivery presuming that every τ_{trans} is equal [42]. Thus, optimizing $\tau_{d\text{-}total}$ is a function of τ_w according to Equation (38):

$$\tau_{d-total} = f(\tau_w) = \sum_{u=1}^{U} \tau_{w_u} \quad (38)$$

Each SN will be informed during the setup phase about ω_i for the lifetime of the network. The BS also can alter this value by broadcast when the conditions of the WSN changes (closed-loop control). The value of ω_i should be calculated to achieve at least the minimum expected lifetime span Ω_{min} without jeopardizing the privacy and data integrity. Thus:

$$\Omega_{high\text{-}th} \geq \Omega_i \geq \Omega_{low\text{-}th} \quad (39)$$

where $\Omega_{high\text{-}th}$ is the highest possible value for Ω_i and, $\Omega_{low\text{-}th}$ is the lowest expected value for Ω_i. When SN does not have a real message to send before the end of the interval period I_i, it will send a fake message according to the procedure explained in the anonymity module. When SN has a real message, it will send it to one node from the neighborhood set (N_i).

7.1.1. Security Analysis

The adversary sees every SN sending a message at a fixed data rate at any one time. It also cannot distinguish any message from the rest of the messages in the network since none has similar ID. If we have N nodes in the WSN, the probability that one adversary can locate the sending node is:

$$P_r = \frac{1}{N} \quad (40)$$

We can calculate anonymity as:

$$\mathbf{A} = \mathbf{1} - \boldsymbol{P_r} \tag{41}$$

7.1.2. Delivery Time

Message follows hop-by-hop path until it gets to the BS as exhibited in Figure 11. In this scheme, the message waits until the end of the interval. The delay will be calculated according to the expression below [42]:

$$\tau_{d-total} = \tau_{w_0} + (\mathrm{HC} - 1) * \omega_i + \tau_{trans_u} \tag{42}$$

It axiomatic that most delay accumulates from holding the message until the end of the interval periods.

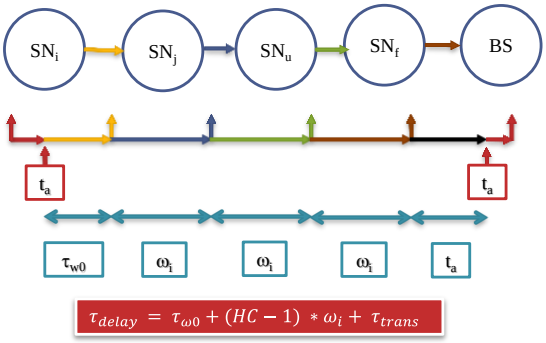
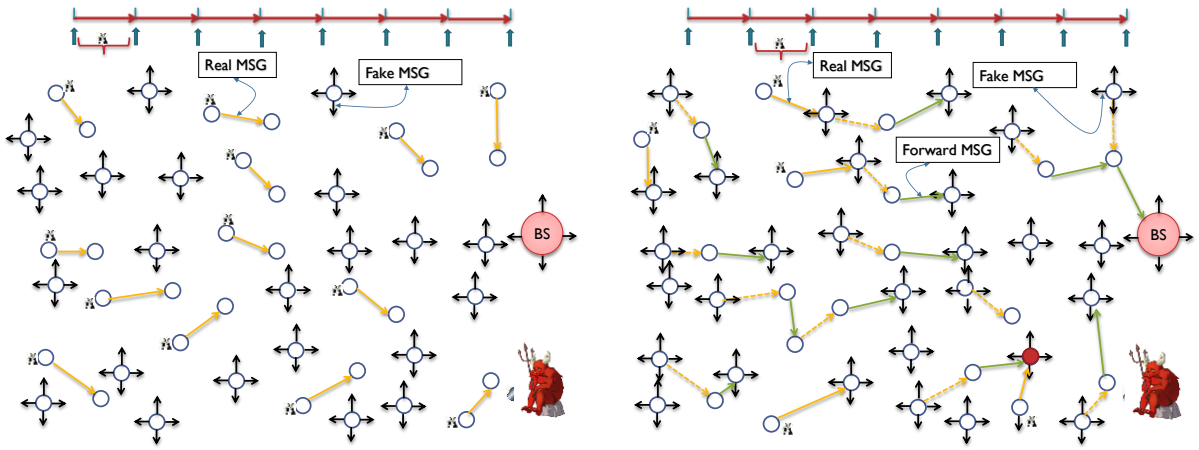


Figure 11. Total delay required to send a message from source to the BS through (*U*) hopes [42].

7.1.3. Energy Cost

We presume in this scheme that every node would send one message at the end of each interval. The message could be either real or fake. If we have (N) nodes in the WSN, then we expect (N) messages during each interval I_i. The energy spent for transmission is almost constant since we have fixed size messages. However, we can evaluate how expensive it would be to use fake messages for privacy enhancement. If we have (Q) percent of the nodes send real messages at each interval, then we are wasting (1-Q) percent of the energy and of the bandwidth.

We can adjust the amount of energy consumed by increasing the interval period ω_i. However, increasing ω_i, would increase the delays. If a SN receives multiple messages in one interval, then it will *queue* the messages for transmission. Because the SN needs to wait until the end of the interval, it could arrange the messages in a queue and send them randomly at the end of interval. This should also increase the privacy and security of the data. It could also select a different forward node for each message. In conclusion, SGAT is energy-expensive due to sending fake/real messages by every node per each interval of time. However, SGAT provides the maximum message *entropy*. Figure 12 exhibits the network transmissions for two consecutive intervals.



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

Figure 12. Anonymity with fake messages. (**a**) Sensors sense events and send real messages while the rest send fake messages. With Fake MSG's: $P_r = \frac{1}{N} = \frac{1}{48} = 2\%$, Without Fake MSG's: $P_r = \frac{1}{n} = \frac{1}{11} = 9\%$; (**b**) Sensors send or forward real messages will not send fake messages. The more the network gets busy the less fake messages are transmitted.

7.2. Energy Controlled Anti Temporal Scheme (ECAT)

There are three major drawbacks in *SGAT*: (i) Having fixed interval time ω_i while it is possible to adjust the value for a better traffic and energy control; (ii) not considering the residual energy as a metric for selecting the forward hop; (iii) high rate of traffic due to fake messages.

7.2.1. Changing ω_i from Fixed to Variable

Having a fixed interval time, ω_i could be a glitch for network performance. If ω_i is set to be a large value, then the delay will be high which could be a serious problem in some time sensitive applications. If ω_i is set to be a small value, then a huge amount of fake messages will be sent at the end of each interval, which will reduce the lifetime of SNs and accordingly the lifetime of the WSN. We propose that we have variable ω_i as presented in [17]. Every node will be calculating its ω_i using a pseudo-random number generator (PRNG). We suggest a uniform distribution algorithm such as multiplicative congruential algorithm [45,46], which is the basis for many of the random number generators in use today. Lehmer's generators [47] involve three integer parameters, *r*, *s*, and *m*, and an initial value, x_0, called the seed. A sequence is generated by the following modified formula:

$$X_{k+1} = b \times ((r \cdot X_k + s) \bmod m + f) \quad (43)$$

The result of the modified PRNG will be a sequence of integer values between ($b \times f$) and ($b \times (m + f - 1)$). Each SN needs to be preloaded with the seed $\times 0$, r, s, m, b and f values. The seed range is 0 to ($m - 1$) and it is uniformly assigned to the sensor nodes. If b = 2, f = 1 and m = 4 then sequence of four intervals will be: $\omega_i \in [2,4,6,8]$ time-units as exhibited in Figure 13. We could have up to (m!) different sequences that are uniformly distributed on the SNs. For instance, we can have Equation (24) different sequences for our example and if we have Equation (48) nodes in the network, so each sequence should be provided to two nodes only.

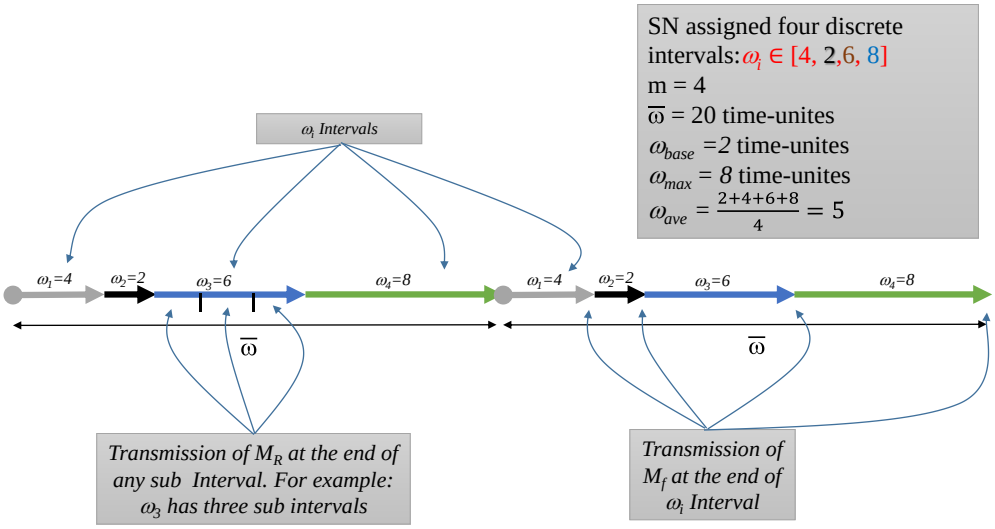


Figure 13. The generation of the sequence of intervals assigned for each sensor. The sequence keeps repeating for the sensor. Fake messages transmitted only at the end of the interval. However, real message could be transmitted at any subinterval.

Each node will be dynamically assigned an interval value, which needs to change after each transmission. Taking the example above, the first $\frac{1}{m}$ th (more or less) of the sensors will send data after 2 time-units. Then, the second $\frac{1}{m}$ th will transmit after 4 time-units, and so on. At any point of time, the adversary will be faced by enough transmissions in the network that it could divert its attention far away from the SNs sending real data. By having (m) interval values where each SN will be generating one ω_i using the PRNG, we have reduced the average interval time from ω_{max} to ω_{ave}. That is explained in the inequality Equation (44):

$$\omega_{max} > \omega_{ave} = \frac{T_1 + \cdots + T_m}{m} > \omega_{min} \quad (44)$$

Considering the earlier example, we have ω_{max} = 8, ω_{min} = 2 and ω_{ave} = 5. That is: we have reduced the delay interval by 37.5%. If m = 8, then delay reduced by 44%. The transmission of real and fake messages is exhibited in Figure 14.

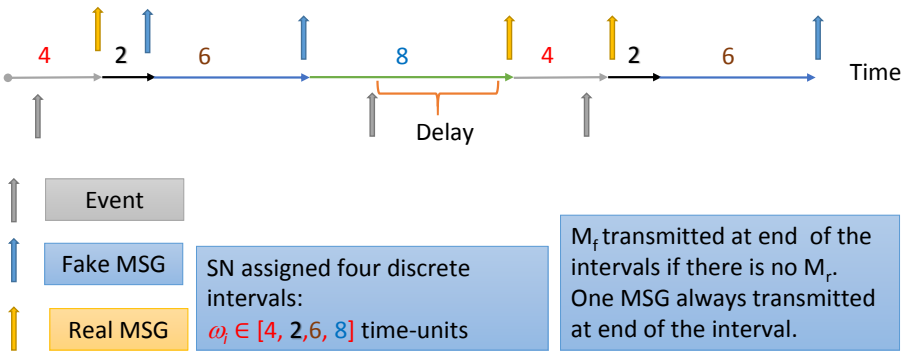


Figure 14. SN is assigned a sequence of intervals, which repeat until sensor lifetime ends. At the end of each interval, the sensor sends a fake message if it does not have an event to report. This should cause different delay times depending on the event relative arrival time and the length of the interval.

7.2.2. Reducing the Amount of Fake Messages and Delay for Real Messages

We have created a mechanism for dynamic interval allocation. Let us call $\overline{\omega}$ the *big interval* which has *subintervals* ω_i. It still makes sense to send fake messages at the end of each interval. However, having the real message wait until the end of the interval time, as exhibited in Figure 15, is not commendable because it increases the delay at each node. Let us presume the current subinterval ω_i is the maximum, which is 8 time-units according to the example discussed earlier. Let us presume that the message was sensed at 2 time-units and it is ready to be sent at 4 time-units. Following the SGAT rules, it still needs to wait for another 4 time-units to be sent out! However, we know for sure that many other nodes have different ω_i subinterval values. Thus, during the time subintervals 2, 4, 6, and 8 there is enough traffic in the network. We propose that when the data is ready, the SN_i should send the data during the next subinterval slot within the current interval ω_i. Consequently, if we are at interval $\omega_{max} = 8$ which has four subintervals at (2, 4, 6, and 8), and for our example, at 6 time-units the data can be sent out. This way, we save about 2 time-units delay while we can guarantee that the adversary will not be able to infer the source of transmission because we have enough traffic distributed in the network.

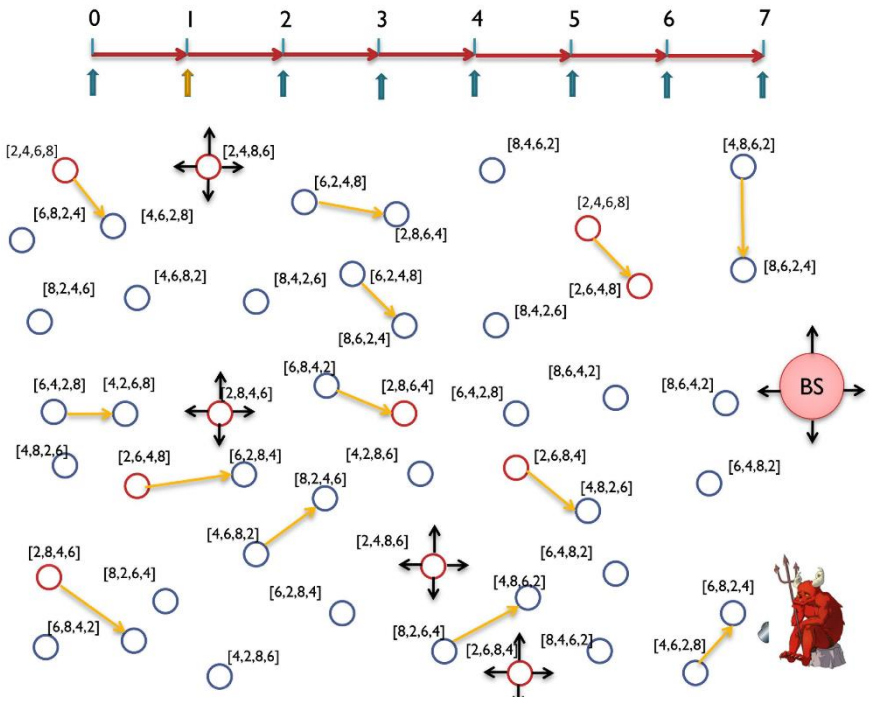


Figure 15. Any node assigned subinterval $\omega_i = 2$ will send a fake message if it does not have a real message to report. All the nodes in the network can send real messages within any subinterval.

If we select higher values for $\overline{\omega}$, then we can further reduce the number of fake messages transmitting at one subinterval, however, we are increasing the average delay as well. Selecting a value for $\overline{\omega}$ could be a tool to adjust security *versus* energy conservation. We have improved the fake message efficacy (*FME*) [27,42] which could be calculated as indicated below:

$$\mathrm{FME} = \frac{\sum_{i=1}^{i=m} \frac{\overline{\omega}}{SI_i} - m}{\sum_{i=1}^{i=m} \frac{\overline{\omega}}{SI_i}} \quad (45)$$

where $\overline{\omega}$ is the big interval value, (SI_i) the subinterval value, (m) the total number of subintervals. For example, if SN assigned a sequence $\omega_i \in [4, 6, 2, 8]$, fake messages could be sent at the following subintervals [4, 10, 12, 20, 24, 30, 32, ...], and the real messages could be sent at the following sub intervals [2, 4, 6, 8, 10, 12, ...]. By substituting $\omega_i = 8$, $SI_i = 2$ and m = 4, FME will be 60%. Figure 14 exhibits the transmission of fake and real messages for two consecutive subintervals.

7.2.3. Energy Conservation by Forwarding Messages to Energy-Rich SNs

When a node senses data or receives data that needs to be forwarded to the BS, it has to select the next one-hop node from the neighborhood set N_i. During the setup phase, each SN_i has information about the hop-distance for each of the neighbors stored in its table T_i. Typically, there are three sets: one set where the hop-distance is less than its own (uplink set), a set where the hop-distance equals to itself (equal-link set) and a set where the hop-distance is larger than itself (downlink set). Choosing a node randomly or by round robin from the uplink set will be ideal in terms of delays since it will give the shortest path to the BS. However, that will cause the nodes in this set to consume more energy compared to the other two sets of the neighbors. After each transmission, the SN consumes some energy. The residual energy for SN_i will be calculated as below:

$$\Delta_{residual} = \frac{\Delta_{current}}{\Delta_{initial}} \quad (46)$$

Each node will calculate its residual energy and share it with its one-hop neighbors. When the node sends fake messages, it will send its residual energy with it. The neighbor SN_j will store the value in its T_j for each of its neighboring nodes. This way, any sensor node will have some information about the residual energy level for its immediate neighbors. Figure 16 exhibits the mechanism for selecting the forward node.

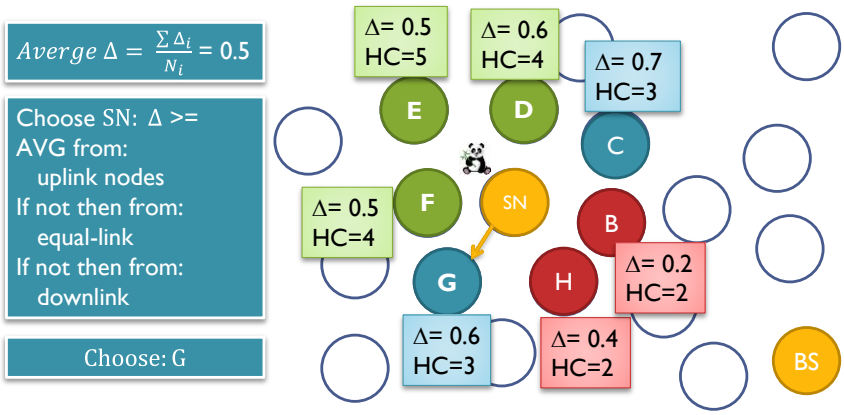


Figure 16. How to choose the forwarding node according to the energy levels of the neighbors. The sensor calculates the average energy levels for all the neighbors. Then it will select a neighbor, which has energy level higher than the calculated average energy, from uplink nodes if it is available. If not, then from equal-link nodes and then from downlink nodes.

7.2.4. Handling Rate Attack

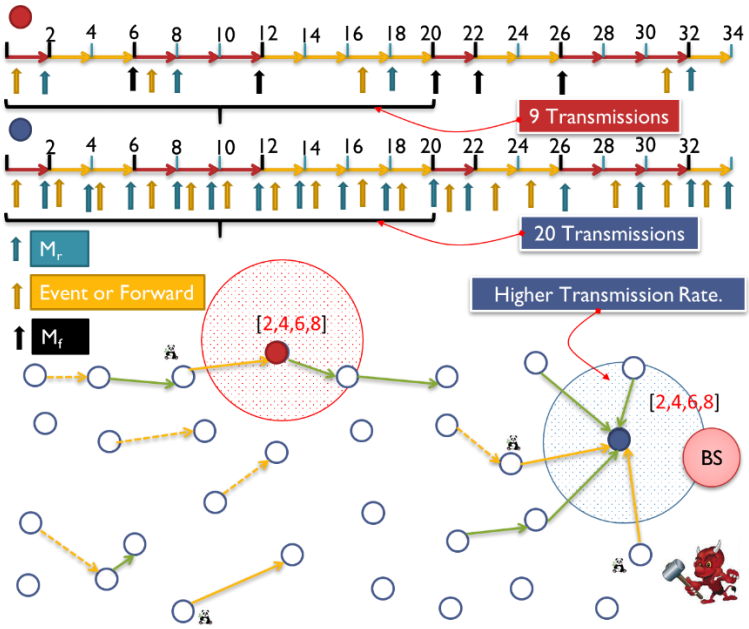


Figure 17. Higher transmission rate next to the BS. The figure exhibits about 20 transmissions near by the BS while one other area has 9 transmissions for the same period. This could be a bed for rate attack where the ADV can locate the BS [42].

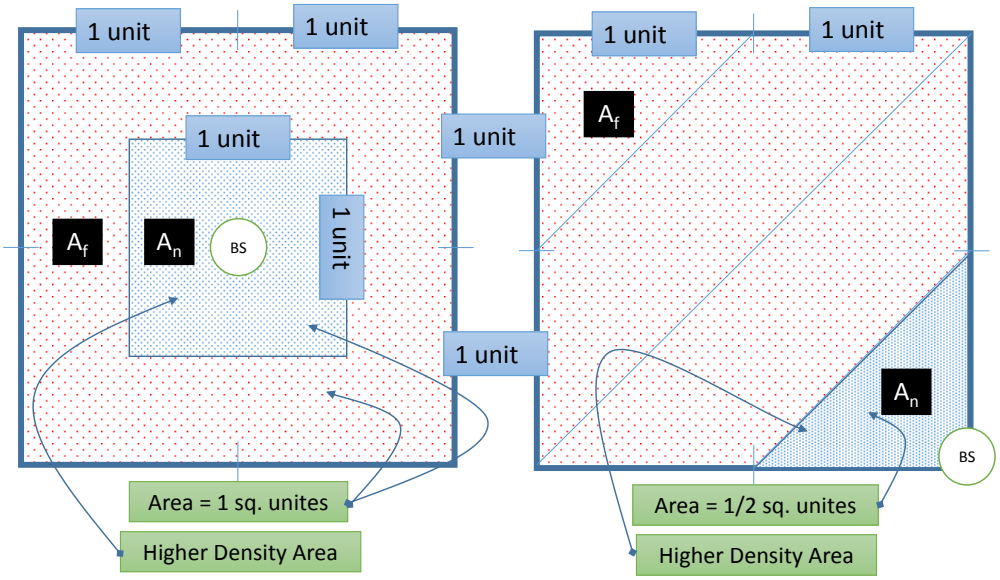


Figure 18. The area coverage of a central sink is higher than a peripherally sink. To balance the higher data rate nearby the sink, we acquire a higher density sensor distribution.

One issue that WSN with one sink could suffer from is having higher transmission rate next to the BS where messages ultimately need to reach out to the BS as the final destination. In contrast, periphery sensors far from the BS could have light transmission rates. Figure 17 illustrates the issue. This could jeopardize the location privacy of the BS. One solution is to have multiple sinks distributed in the network. This contradicts with the pre assumptions we set for our framework so we will not address this solution in this work. The framework needs to maintain similar average rate among all the sensors. This

could be achieved by increasing the number of fake messages transmitted by less busy nodes, which means increasing the bandwidth usage and the power consumption. We need also to reduce the fake messages sent by busy nodes or delay the real messages to maintain similar rates. The latter is achieved automatically since the sensors do not send fake messages when they have real messages. However, this could be better tuned for average busy nodes as well. Having balanced rate in the WSN could help to maintain balanced average lifetime for the nodes in the network. Presuming that all the nodes are heterogeneous in terms of energy would mean that busy nodes would be depleted sooner that could create an empty coverage area or a buffer zone between the sink and the peripheral SNs. This makes it a double fold problem. The first approach is to select a suitable location for the BS in the network map. Most of the literature shows a side location for the BS. It is maybe because it is more suitable for the applications in hand where the BS is connected to the backbone network in a reachable area and sensors are unattended in out of reach areas. Figure 18 exhibits that the coverage area of a central BS is much better than a side BS. The density of nodes closer to the BS should be higher. The range of transmission for sensors in higher density areas may need adjustment to control energy dissipation. We could have multiple density areas around the BS where the density is reduced, as it gets distant from the BS. Figure 18 exhibits only two density areas for simplicity. If the storage of the sensor is not big enough, which is unrealistic case with increasing storage technology in the sensors, then the sensor does not need to include all the neighbors in the tables. The network will be divided into two areas, near (A_n) and far (A_f). The framework will set average transmission rate (ATR) thresholds, R_{max} and R_{min}. Sensors in A_n will be loaded with R_{max} where the sensors need to queue messages to maintain the threshold. In reverse, sensors in A_f will be loaded with R_{min} to maintain the lower threshold by sending more fake messages as needed. The sensor will calculate its average transmission rate over a period of time $T_{atr,}$ which is preset by the framework.

7.3. Contribution of Temporal and Rate Privacy Module

Our innovation in this module is by providing both temporal and rate privacy. Many works have provided solutions for temporal privacy by either using delays or fake messages, but few has addressed the rate privacy as an independent threat to the WSN. In our module, we have used delay and fake messages to provide an efficient solution for such attacks. We took in consideration, the need to reduce the delays in the real-time applications and the necessity to reduce the energy dissipation. In addition, very few works has addressed the rate privacy for the BS presuming it is physically protected. In this framework, we always considered the BS as a normal sensor node, which requires privacy. Section 9 provides a thorough analysis and simulation for the delays, entropy and energy. The three modules of anonymity, authentication, temporal/rate privacy altogether will provide source, link and sink location privacy.

8. Anonymity and Security Analysis

We need to analyze FAC for both passive and active adversary attacks. The adversary (ADV) model has a global view of the network. ADV could target the source, intermediary and BS nodes. Usually, ADV starts by monitoring transmission somewhere in the network and then attempts to acquire sources (downlink direction) or BS (uplink direction). Passive attack is ordinarily the base for active attack. Once

ADV determines the identity and location of a source or the BS, it consequently can launch various active attacks against certain nodes or disrupt the operation of the entire WSN. The main strength of passive ADV is the fact that neither SNs nor the BS will know about their existence. Nonetheless, active attacks can be detected if the framework instruments reasonable IDS. Any comprehensive solution for location privacy should protect against anonymity attacks, temporal attacks and rate discovery attacks. We believe that routing privacy is useful only against local advisory and once the WSN faced with a global or a multi-local adversary, routing privacy is not crucial. Thus, we have chosen short-path routing technique for this work. Any other routing protocols should be utilized to reduce delays and energy consumption.

8.1. Security against Passive Attacks

SNs use disposable pseudonyms to identify each other instead of using real IDs. No real ID stored in the sensor and no pseudonym is used more than once. Data is encrypted all the way from the source to the BS using shared pair-wise keys. For *eavesdropping and content analysis*, ADV can intercept messages without being able to read them because data is encrypted all the way to the BS. The only information ADV can get from the captured messages is the pseudonyms: OHPID, BPID or FPID, which are all temporary IDs and have no use except to calculate a new set of pseudonyms. Fortunately, the ADV cannot get from the captured messages, important parameters $a_{i\leftrightarrow j}$,b_i or c_i which are all required to calculated new pseudonyms. Source PIDs are all encrypted during transmission. For *hop-by-hop trace*, ADV can track a stream of messages from one node to another by overhearing the messages. The ADV will be challenged with many real and fake transmissions throughout the WSN. Furthermore, each node will retransmit the messages through different routes. For *size-correlation*, ADV will be able to understand relationship between incoming and outgoing messages by analyzing sizes of the messages. This attack does not work for our framework since all the messages have commensurate size. For *identity correlation*, ADV cannot relate overheard identities to their nodes. It is not possible since SNs use different pseudonyms every time a message is transmitted. For *rate monitoring*, ADV tries to collect some statistical information about transmission rates. For instance, WSN will have a higher transmission rate nearby the sink. This is handled by issuing fake messages to maintain a similar transmission rate. For *angle-of-arrival (AoA)*, ADV uses special hardware to determine the signal direction. The framework did not account for specific countermeasure; however, it becomes a more serious issue with mobile SNs. Furthermore, *AoA* would not perform well in our framework because of the uniform message distribution by using real and fake messages. For *received-signal-strength (RSS)*, ADV uses special hardware to measure signal strength to calculate distance to the source. This is not an issue for our framework since every transmitter has fixed transmission power and SNs are immobile.

8.2. Security against Active Attacks

In principle, we assume ADV knows encryption protocols used by the framework; however, the framework needs to hide *encryption keys* and *IDs*. Active attacks can be categorized into *soft* and *hard*. For soft-active attacks, ADV tries to compromise SNs to get some information related to security of the sensors such as *keys* and *IDs*. Consequently, it will monitor all messages traversing through the compromised nodes to discover the source and the BS locations. ADV hides its presence by acting

passively (soft) but once it captures privacy information, it reports the information to an external executer to do further damages (such as killing the Panda in the Panda game). For that, it is harder for the IDS to detect the attack. In hard-active attacks, ADV captures SNs and invasively forge messages, sent replay messages *etc*. Moreover, ADV could load powerful devices with the captured credentials to launch more catastrophic attacks. Hard-active attacks could be detected by IDS; however, it could depend very much on the sophistication of the IDS used. With that, it remains very challenging to countermeasure hard-active attacks. In the following two subsections, we will analyze the security of our framework against active attacks.

8.2.1. Soft-Active Attacks

If ADV *physically* compromises SN_i, then it captures two sets of information:

(i) Information related to the node itself: the current *PID*, the parameters used to calculate the pseudonyms, the hash functions, the keys and other information as listed in Table 2.
(ii) Information related to the neighbors as listed in Table 3.

The ADV would have all it needs to issue new valid pseudonyms and to send messages out to neighbors. Let us look closely at few scenarios:

Scenario 1: If ADV physically compromises SN_i, and if SN_j and $SN_r \in N_i$, so SN_i knows some information about both SN_j and SN_r. However, it cannot calculate important information such as $a_{j\leftrightarrow r}$ which is required for one-hop communication between SN_j and SN_r [16], because SN_i would need ID_j and ID_r which are both deleted permanently at the end of the setup phase. If SN_i hears a message, it cannot determine, with high confidence, the sender among neighbors while communicating with each other. If SN_i receives message from sources $\notin N_i$, then it would not be able to determine the source.

Scenario 2: If ADV *physically* compromises multiple SNs, let us call it set N_{cs}, and collects number of messages, let us call it set N_{cm}. Then, the number of compromised PIDs equal to N_{cm} since each message has unique PID. If the source $SN_i \notin N_{cs}$, then ADV cannot know the source node [16,27].

Scenario 3: If the message sent, by source SN_i as in *scenario*, 2 passes thought $SN_j \in N_{cs}$ or even through multiple compromised nodes, it will not be able to correlate the captured PID_i with SN_i.

Scenario 4: If a message sent by source SN_i and $\forall$ SN $\in N_i$ is also $\in N_{cs}$ (all neighbors are compromised), then ADV will be able to know that SN_i is the source. It is unrealistic situation to have many compromised nodes in one area. However, this proves single or few compromised nodes cannot locate the identity of the source. In addition, a compromised node does not actually need to locate the sources within its range since it can detect the objects of interest (Panda) knowing that the ultimate goal of the adversary is to capture the *object* not the sensor reporting the object.

In summary, while we cannot prevent physical capturing of sensors, we need to make sure capturing sensors do not have destructive effects on other sensors. It is clear that our anonymity model protects against the *avalanche* or the *domino effect* behavior once one or few sensors are physically captured.

8.2.2. Hard-Active Attacks

If ADV physically compromises SN_i then it can launch denial of service attacks (DoS), which is an effort to temporarily or indefinitely suspend transmission in the network. It consumes the resources such as bandwidth, memory, storage, and processor time. When ADV compromises SNs, it would be able to send massive valid messages to consume system resources. The ADV will be able also to launch replay attacks where ADV gets credentials of the some sensors and attempts to mimic the sensors to send messages to other neighbors. The other attacks such as, forging attack, packet alternation, packet dropping and packet injection are all only possible to physically captured nodes. However, it cannot propagate easily behind neighbors. Nothing could be worse than having physically captured nodes where ADV has full control over the sensors. Good IDS can detect such attacks and respond by removing the compromised nodes immediately. The most danger tactic of hard-active attacks is to prevent the real messages from following normal paths to the BS and force the messages to traverse through certain routes. Our main contribution to handle this attack is to put in place a seamless and efficient protocol to add and remove SNs while WSN in action.

8.3. Sink Security

ADV can learn that a sensor has received a message in two ways: (i) When the sensor retransmits the message, which was tracked beforehand to another sensor; (ii) the ADV is able to make a correlation between the captured ID and the physical recipient sensor. The adversary cannot locate the BS location by compromising only one neighboring sensor because each transmission uses a different pseudonym. It actually will need to compromise multiple colluding sensors along the path to the BS or many neighbors of the BS. While we cannot prevent having many physically fallen sensors, our framework's goal is to delay the capturing of the BS if there are many colluding captured sensors in the WSN. A very interesting *scenario* is *exhibited* in Figure 19. Let us presume $SN_r \in N_{cs}$. It issues a message with D_{bomb} such that: $APID_r \,||\, OHPID_{r\leftrightarrow u} \,||\, E_{r\rightarrow u}\,(APID_r \,||\, PID_r \,||\, E_{kr\leftrightarrow bs}\,(\boldsymbol{D_{bomb}}))$. IF ADV compromise multiple nodes along the path to the BS where each sensor decrypts the data to read this *signature* at every hop: $(\boldsymbol{PID_r} \,||\, \boldsymbol{E_{kr\leftrightarrow bs}}\,(\boldsymbol{D_{bomb}})$. Providing the colluding sensors, in the path to the BS, read similar signature while it knows by design that every message should be directed uplink to the BS, the ADV could follow through to the BS. Having multiple compromised paths (with compromised sensors) reading the same pattern will give adversary more clues. Compromised nodes can even collude to force the real messages to route through fixed suspected areas in effort to focus the capturing process, which becomes a function of: (i) the size of the network; (ii) The traffic density; (iii) the number of compromised nodes. To solve this issue, we have to wipe out the signature before each transmission. Thus, every message will be forwarded to the next hop as below:

$$M_{u\rightarrow x} = APID_u \,||\, OHPID_{u\leftrightarrow x} \,||\, E_{u\rightarrow x}\,(APID_u \,||\, PID_r \,||\, \mathbf{PID_u} \,||\, \mathbf{E_{ku\leftrightarrow bs}}\,(E_{kr\leftrightarrow bs}\,(D_i))) \quad (47)$$

We have added a multiple levels of encryption, which will be done at every hop using the shared key between the hop and the BS. In addition, PID of the hop will be added in sequence so the BS can do the decryption in sequence. This solution increases the size of the message proportionally to the number of hops. We suggest having the onion encryption done for a distance of few hops, O_h. So, if

O_h = 2, then we have only two extra encryptions. In addition, we need to account for O_h PID's added to the message.

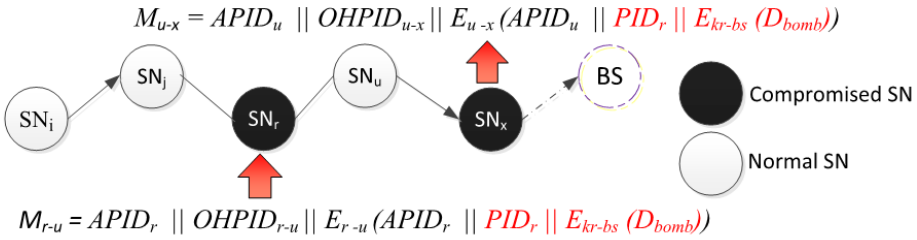


Figure 19. Hard-active Attack tries to get the BS by inserting a signature in the transmitted message.

We have implemented using Matlab a WSN of 100 SNs uniformly distributed over 30 × 30 area where the average distance between the SNs 3.7 as in Figure 12. The nodes are homogeneous in terms of initial energy. The WSN adopted one BS located at the side of the network. The SNs were preloaded with all the initial pseudonyms, so the simulation started right at the communication phase.

Sensors issue real messages according to a normal distribution using SGAT. To simulate how the network behaves to protect the BS, the simulation inserted some random compromised sensors. The compromised sensors sent some bomb messages as exhibited in Figure 19 and colluded to track the location of the BS. We have protected the BS by using the onion encryption so, we have simulated for O_h equals to 1, 2 and 3. The adversary succeeds when it identifiers all the nodes forming the curve around the BS; SNs have 1, 2 and 3 hc from the BS, consecutively. Figure 20 exhibits the number of transmissions required before the adversary can succeed. It is clear that with higher value for O_h, the network will be able to send more messages before the BS is compromised. Having a higher number of compromised nodes in the WSN will make it faster to capture the BS, as well.

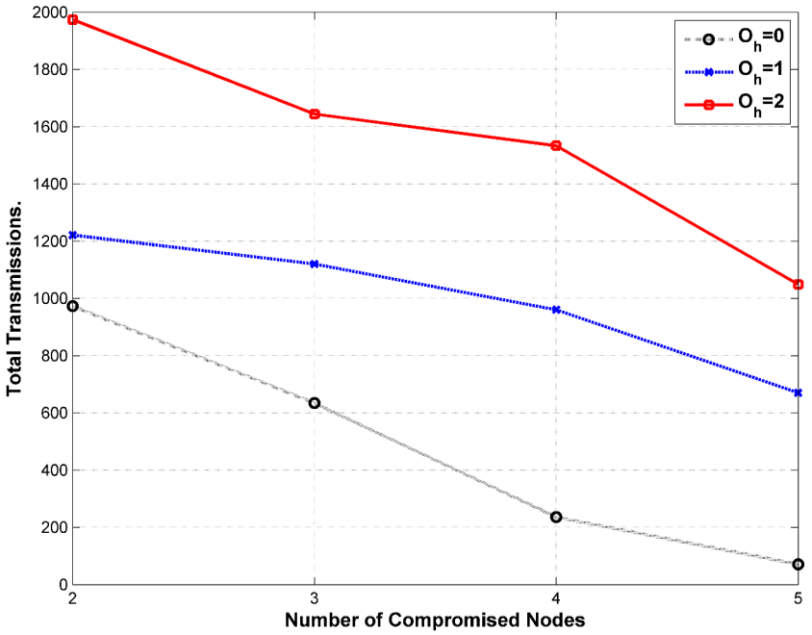


Figure 20. Protecting the BS by having onion encryption. Increasing O_h and decreasing the number of compromised nodes will increase the total number of messages successfully transmitted to the BS before it is captured.

8.4. Link Anonymity

Link anonymity is to prevent the ADV from knowing the relationship between the sender and the receiver. If a message leaves a sender and subsequently leaves the recipient without change, the ADV would know the relationship between the two nodes. This is secured in our framework since every message is completely changed after each retransmission including the IDs. In addition, it maintains fixed size. Applying different delays and different next-hop direction should also increase the privacy of the link. Furthermore, the adversary cannot know if the link carries real or fake data.

8.5. Timing Privacy

By using fake messages at variable interval times and message delays, it becomes super hard for the ADV to correlated messages being transmitted over the network as exhibited in Figure 12.

8.6. Routing Privacy

Although short-path routing is used in this framework, choosing the next hop is done according to certain *probabilistic* algorithm, which accounts for the residual energy of the sensors and the usage frequency to increase the route privacy, as exhibited in Figure 16. ADV cannot relate routes to nodes due to the triple anonymity. Even if two messages for one sensor follow the exact same route, ADV will see them as if they are two different routes since each hop along the route carries messages with different PIDs.

8.7. Data Privacy

All the data is encrypted before transmission and encrypted again at every hop along the route to the BS. A message digest will authenticate data. The only time data is not protected when the sensors are physically compromised. The compromised nodes are able to inject data in the network. If ADV uses the compromised nodes actively, a good IDS can detect the falsified data. The framework provides a secure facility to remove compromised sensors and to add valid sensors, if needed, to the WSN.

8.8. SLP and BLP

SLP and BLP are achieved at first by having the triple anonymity (*source*, *BS*, *link*) which was argued earlier. ADV cannot infer any information from the intercepted messages. Passive attacks will not endanger the location privacy. However, strong active attacks could hinder the location privacy without having good IDS. Secondly, we have provided a solution for temporal privacy using ECAT. Thirdly, we have provided a solution for rate attacks. The three security measures will work hand in hand to provide location privacy.

9. Performance Evaluation

In this section, we evaluate the performance of the FAC framework, including delays, energy dissipation, data rate privacy, storage, processing, computational, and communication costs.

9.1. Delay

In SGAT, sensors transmit/forward the data at the end of the interval, which would cause a huge delay considering the volume of messages that each sensor needs to transmit during the network real-time operation. In addition, the messages traverse through multiple hops until it gets to the BS, which makes the accumulated delays significant. The other alternative scheme is having the sensors select one of the following subintervals (ω) randomly to forward the message. This also will cause some unnecessary delays although it could help in hiding the temporal behavior of the sensors. ECAT scheme divides ($\overline{\omega}$) into subintervals (ω), so the transmission will happen at the first available subinterval when the message is ready. We have simulated a smaller network to the one descripted in Section 8.3 for the transmission delays. It includes 48 SNs only with ω distribution as presented in Figure 15. We have three simulations using SGAT, ECAT, and random delays. Figure 21 shows that delay per one-transmission increases throughout the network as the number of transmitted messages increases which could cause unjustifiable delays especially in the real time applications. Figure 22 also shows the average delays for the three schemes. It shows that using ECAT has improved delays by 64% compared to SGAT. The total delay for one message from a source to a destination (BS) is calculated according to expression Equation (37). It is a function of the distance from the BS (hc) which we technically have no control over after sensors deployment. In addition, it is a function of the chosen ($\overline{\omega}$) and (ω) values for the system. The larger the (ω), the more delays accumulated. We have simulated the same network using ECAT for the total delay as exhibited in Figure 23. It shows that the delay rises as the *hc* increases and as the size of the intervals widens. We conclude of these simulations that the performance of ECAT is better than SGAT while it continues to provide a good temporal privacy. Using a fixed delay will reduces the delays slightly but it provides a very week temporal privacy.

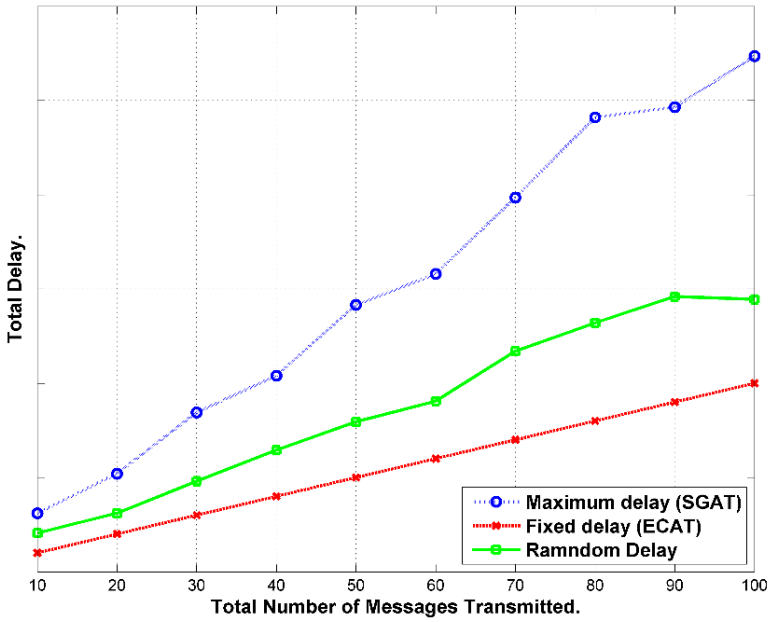


Figure 21. Total accumulated delay per one node increases as number of messages increases in the WSN. ECAT sends the message only one subinterval after the message arrival. SGAT sends the message at the end of the big interval $\overline{\omega}$. In between, the approach of selecting one of the following subinterval randomly to send the message. Clearer image?

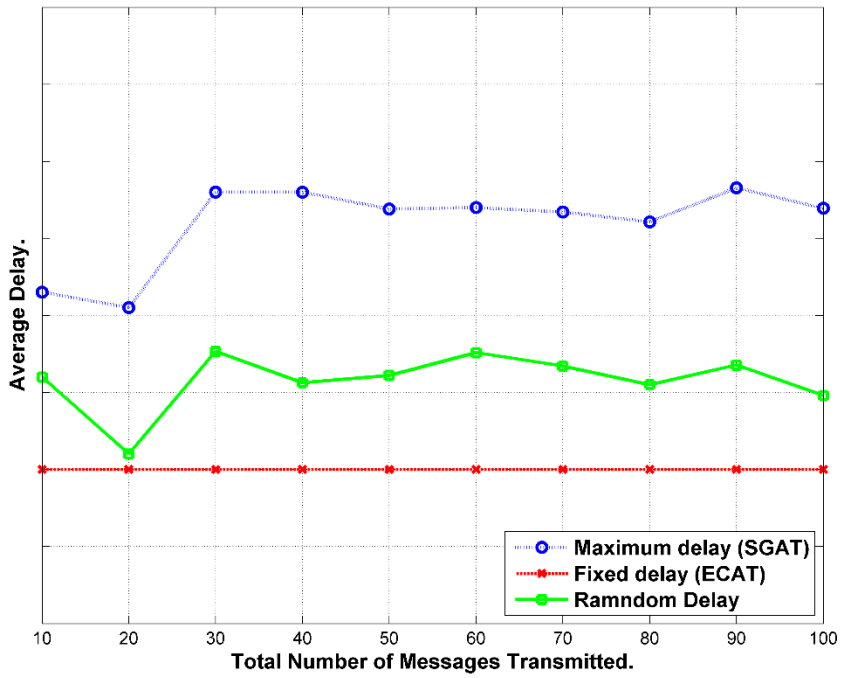


Figure 22. Average accumulated delay per one node. ECAT shows the good performance of a minimum average delay since it sends the message only one subinterval after the message arrival.

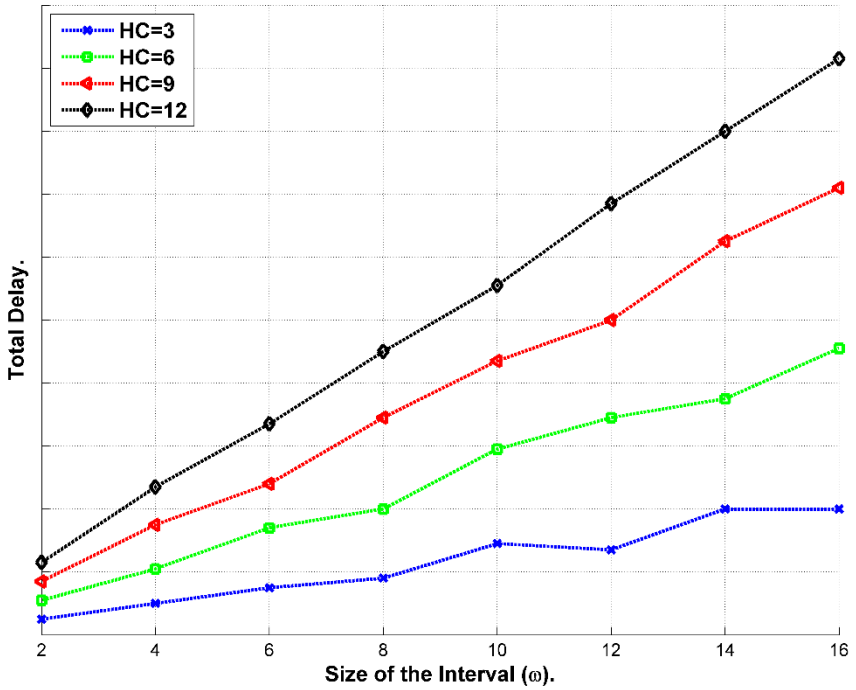


Figure 23. The accumulated delay is a function of the hop count (hc) and the size of ω.

9.2. Energy Cost

In our work, we will assume a simple energy dissipation model [27,42,48,49]. The radio dissipates Ɛ nJ/bit for both transmission and reception by the sensors circuitry. Moreover, it consumes ε nJ/bit/m^2 for

the transmitter amplifier to achieve an acceptable signal to noise ratio. Therefore, to transmit *k* bits for *r* distance, the total transmission energy dissipation will be:

$$E_t = k \times \mathcal{E} + k \times r^2 \times \varepsilon \quad (48)$$

In addition, the receiver would consume for reception of k-bit message:

$$E_r = k \times \mathcal{E} \quad (49)$$

SGAT assumes that every node would send one message at the end of each interval where the message could be either real or fake. If we have *N* nodes in the network, then we expect *N* messages during each interval. The energy spent for transmission or reception is similar per one message because we have unified-size messages to prevent size correlation attacks by the adversary. If we have *p* percent of the nodes issue or forward real data at each interval, then $1 - p$ percent of the energy and the bandwidth is wasted on fake messages. We can adjust the amount of energy consumed by increasing the interval time (ω). However, increasing (ω), would increase the delay. The consumption of transmitting fake messages is a double fold since the transmitter will consume E_t for every message and all the neighbors (N_i) will consume ($N_i \times E_r$). When the transmission range increases, N_i increases. The total energy consumed in the network to send real and fake messages in one interval [48]:

$$E_R = (N)(k \times \mathcal{E} + k \times r^2 \times \varepsilon) + (N \times N_i)(k \times \mathcal{E}) \quad (50)$$

ECAT has improved the energy dissipation by reducing the amount of fake messages transmitted while maintaining the required temporal security. The number of total messages transmitted per interval has reduced from 100% to a certain percentage (*p*). We have simulated the WSN in Figure 15 as presented in Section 9.1 using SCAT to calculate the energy dissipation. Figure 24 exhibits the total energy dissipation per one message considering the transmitter, the recipients and the range of transmission. The size of the messages is 8000 bits, $\mathcal{E}$ is 50 nJ/bit and ε is 100 pJ/bit/m^2. The simulation shows that the energy dissipation due to the increase of sensor range is marginal compared to the increase in energy dissipation due to the increase of neighbors (N_i). However, increasing the range could increase N_i if the WSN has uniform sensor distribution. We have also simulated the network to see how the transmission of fake messages has improved using ECAT. Figure 25 exhibits the simulation of 40 subintervals (ω). The graph shows the maximum possible fake message at each subinterval (ω). For instance, the total fake messages during ω = 10 is 16 messages while during ω = 32 is 20 messages. The mean of fake transmissions is 19.5 (compared to 48 messages in SGAT). The average fake messages for the completely simulated period is 19.5 messages which shows about 59% reduction of possible fake messages comparted to SGAT.

The number of fake message will be reduced further as the network gets busy transmitting real messages since a sensor node do not send a fake message at a subinterval where it has a real message to convey. We have simulated the same network with 70% probability of event occurrence. Figure 26 shows that the average fake messages has reduced to 5.85 messages, which is almost 88% reduction from SGAT. This also will reduce the energy consumption significantly.

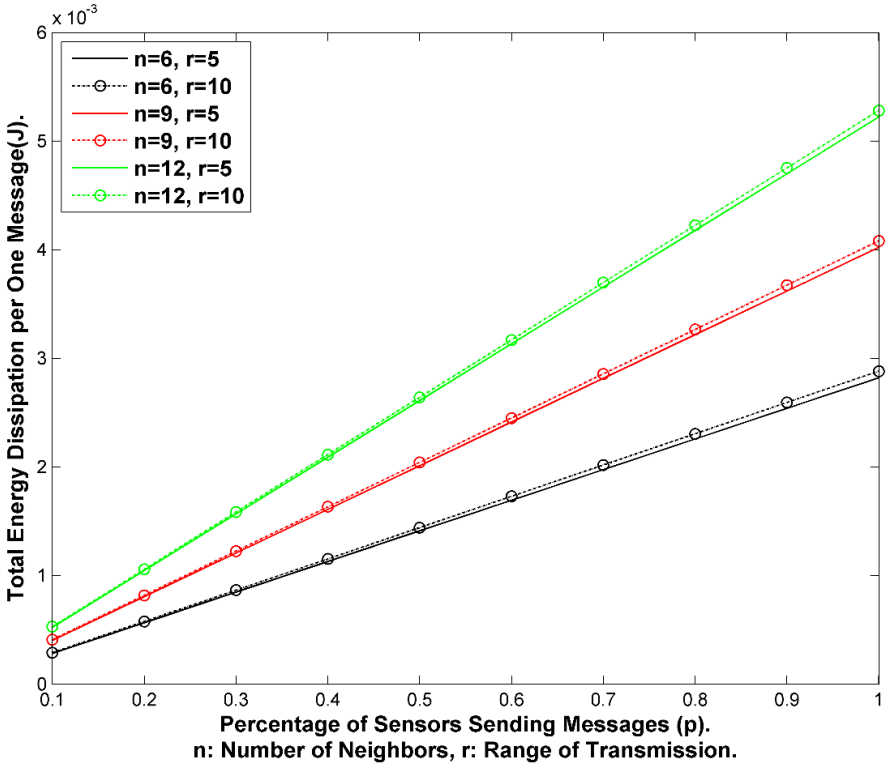


Figure 24. The energy dissipation increases as the number of neighbors and the sensor transmission range increases. Simulation assumed message size of 8000 bits, Ɛ is 50 nJ/bit and ε is 100 pJ/bit/m^2.

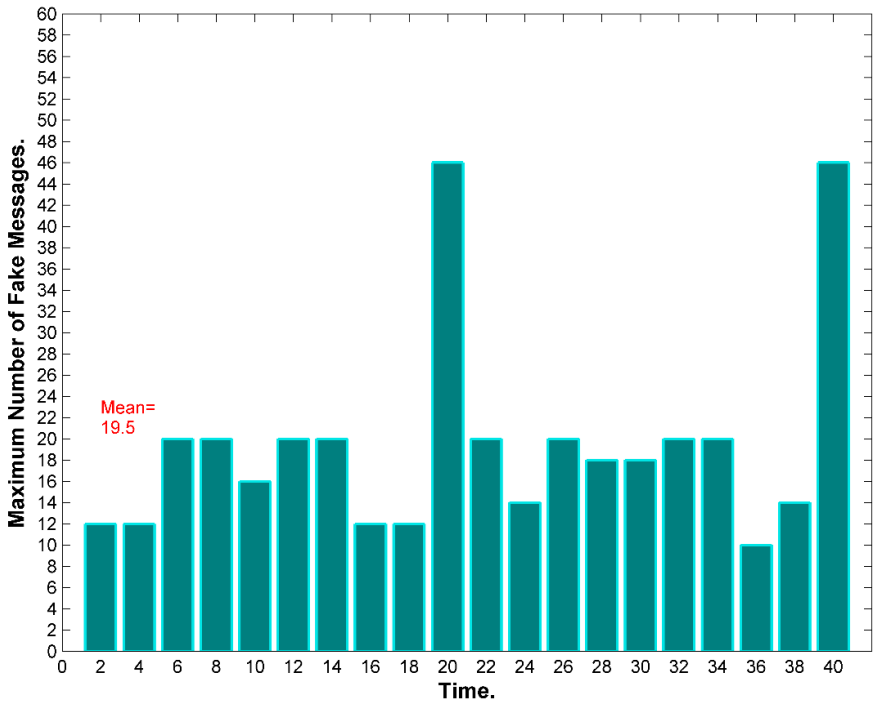


Figure 25. A simulation for the maximum possible fake messages per subinterval using ECAT. In SGAT, this number should be 48, which is one message per one sensor. ECAT has reduced it significantly. The mean in this simulation is 19.5, which is about 59% reduction of fake messages.

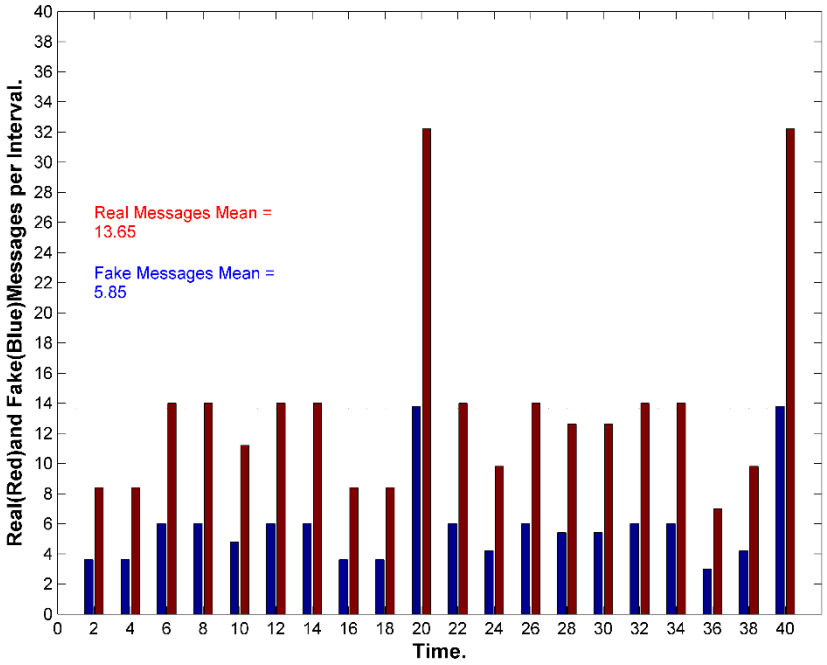


Figure 26. The average fake messages in a busy network with 70% of the slots occupied by real messages. The reduction of fake messages transmission in ECAT is about 88% compared to SGAT.

The most expensive operation for energy consumption is transmission of *bits* from one node to another. We use two stages for air communication in our framework, (i) In the setup phase and; (ii) in the communication phase. The data transmission during setup phase is minimal. During communication phase, data will be forwarded hop-by-hop to the BS. Every packet is equally sized to prevent time and size correlation. We have introduced a probabilistic fake message transmission scheme which none of the other protocols adopted. Real messages are sent at the end of each subinterval time to prevent delays.

The cost per message at one interval time is:

$$\text{Average Message Cost} = \frac{R + (N - R)P_r + A}{R} \quad (51)$$

where R is the total number of SNs sending real messages at one subinterval time, P_r the probability of sending fake message by SNs, and A is the average number of acknowledgements in one interval. None of the other schemes addressed the issue of rate analysis attacks, which is one of the easiest attacks any adversary can use. Using fake messages is an expensive solution. However, we have designed FAC to be adaptive to the network traffic situation by using a closed-loop system. The sink can always increase or decrease the amount of fake messages used according to the reports it is getting about the system security. The threshold values of R_{min} and R_{max} are also adjustable according to the network situation.

9.3. Transmission Rate Privacy

To handle this issue, we have adopted two threshold values: R_{min} and R_{max} where the sensor needs to keep its message transmission rate between these two values. The sensor needs to send real message at the end of subinterval time slot. If it does not have a real message, then it needs to send a fake message only if that time slot is scheduled to send a fake message according to ECAT protocol, otherwise no

transmission will happen and the slot remains idle. Ideally, the sensor has a real message at the subinterval so it does not need to waste a slot by sending a fake message. The sensor can use this facility to control the threshold data rate. For instance, if the rate is high (such as in areas nearby the BS), it can replace fake messages with real messages which is a double fold beneficial. If all the fake messages are already replaced and still there is real messages above the threshold, then the sensor is required to queue the messages and delay the transmission to maintain same average message rate between R_{min} and R_{max}. In contrast, if the message rate is low (as in the periphery sensors), then the sensor can transmit more fake messages during idle slots. We have simulated the network in Figure 15 for ECAT and assigned the R_{min} to be *thirteen* messages for two consecutive ($\overline{\omega}$) intervals (a total of 20 subintervals). We have assigned the R_{max} to be 13 messages during this period, which is seven less than the total number of subintervals. That means we allow up to 13 real and fake messages during these two consecutive ($\overline{\omega}$) intervals. Figure 27 exhibits the output of the simulation for four different individual transmissions. For instance, the first transmission shows, 14 real messages (blue bar), 2 fake messages (light blue bar), and *four* idle slots (green bar). The total real and fake messages is 16 (*orange bar*) which is above the assigned threshold, *thirteen*, by *three* messages, which is expressed by the brown bar.

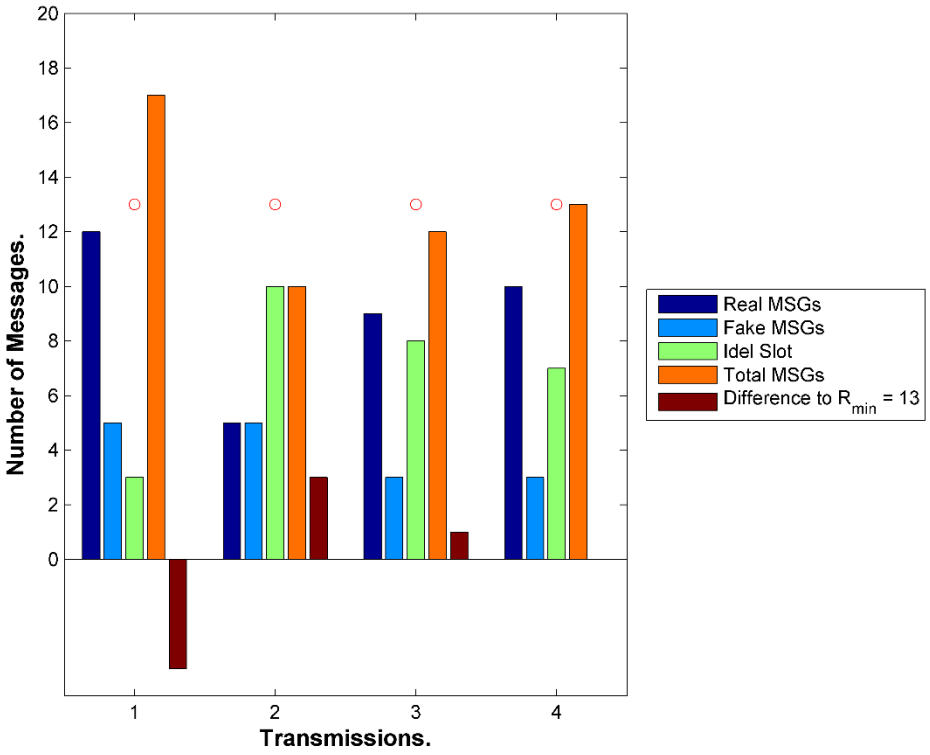


Figure 27. The simulation shows the total real messages, fake messages and idle slots. The threshold rate is *thirteen* messages over T_{atr} period. For instance, the first transmission shows total of *sixteen* messages, which is *three* above the threshold value. The sensor will cancel three fake messages. The third set shows the number of transmissions at the threshold.

The SN will cancel *three* fake messages out of the *four*. In some worse cases, the sensors would need to queue the messages for next slots. For instance, if there is 15 real messages during this time, the system send only 13 messages and queue 2 messages for the next period. Ultimately, the overall message rate during T_{atr} will be within the assigned thresholds.

We have simulated this approach extensively for (500), (1000), (1500) and (2000) transmissions as exhibited in Figure 28. The first, third and fifth bar sets show the total amount of fake messages needed to be replaced with real messages to maintain R_{min} for the thresholds of th = 10, th = 11 and th = 12 consecutively. For instance, a threshold of 10 means that the maximum number of messages transmitted should be 10 (out of 20 in our simulation). The second, fourth and sixth bar sets also show the number of messages which needed to be queued for the three consecutive threshold values. Therefore, if we have real messages above the number of scheduled fake messages, then we have to queue the messages for the next period of T_{atr}. Overall, this simulation exhibits a great preference since we always would like to reduce the amount of fake messages and keep the bandwidth busy with real messages whenever it is possible. In addition, the simulation exhibits very small messages need to be queued (delayed). It shows as we increase the threshold value the less fake messages replacement or delays is required. In summation, reducing the fake messages and keeping the delayed message minimal is the goal, which ECAT clearly achieves.

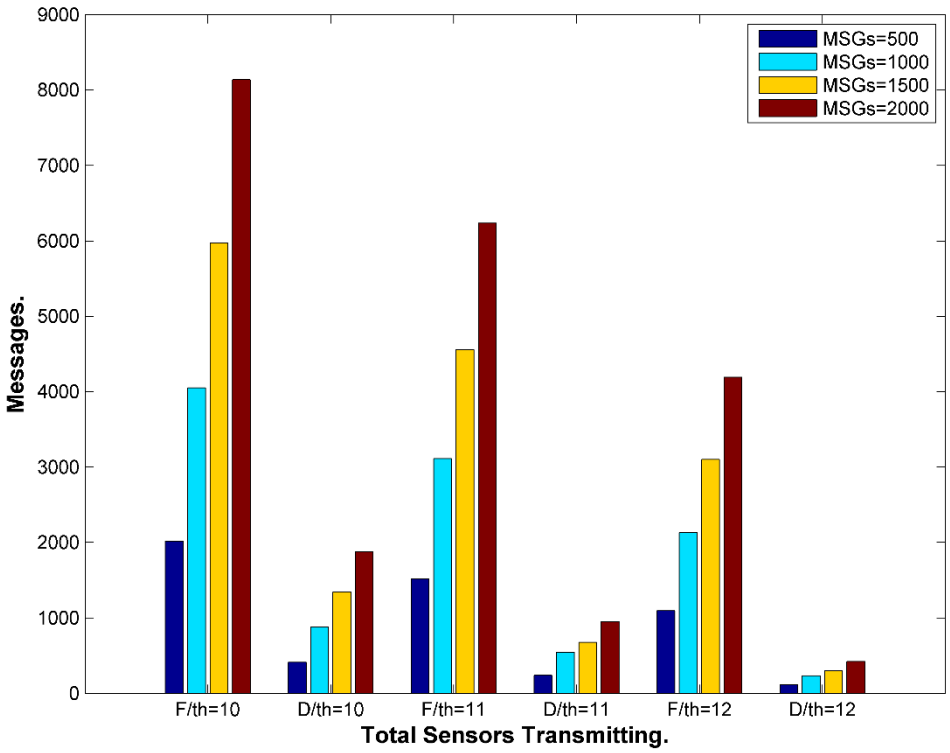


Figure 28. Simulation for busy network with different R_{min} threshold values th = 10, 11 and 12. F stands for the number of fake messages has been reduced and D stand for the number of delayed messages to maintain R_{min}. The simulation shows that the number of delayed messages is minimal while it decreases as the threshold increases.

9.4. Storage Evaluation

There are two sets of information stored in a SN_i: (i) information related to the sensor itself such as random numbers: (a_i, b_i, c_i), pseudonyms: (PID_i, $BPID_i$, $FPID_i$, $APID_i$), keys: ($ki \leftrightarrow bs$, bki, fbki); (ii) information related to neighbors which include, random numbers: ($a_{i \leftrightarrow j}$, b_j, c_j), pseudonyms: ($OHPID_{i \leftrightarrow j}$, $BPID_j$, $FPID_j$), keys: ($k_{i \leftrightarrow j}$), Misc: ($link_{i \leftrightarrow j}$, Δ_j).

If we presume that the keys, the random numbers, the pseudonyms and the hash functions are all *n* bits long in average, and the required bits for miscellaneous data altogether is *two* bytes, and the average number of neighbors N_{ave}, then the total storage memory required is:

$$\text{Storage} = 10n + (7n + 16) \times N_{ave} \quad (52)$$

Chen *et al.* [16] indicated the storage for *SAS,* CAS, *APR, DCARPS and EAC.* We also calculated the storage for *PhID*, *ACS*, HIR and RHIR. All are listed in Table 4. The size of storage increases proportionally when the size of *n* increases. The most common hashing functions which are considered very secure are: *MD4* [50] which uses 128-bits digest, *SHA-1* [50] which uses 160-bits digest, and *Whirlpool* [50] which uses 512-bits digest [50].

Table 4. Performance Comparison. *N* is the total number of sensors; N_{ave} is the average number of neighbors; *k* (only for RHIR) is number of stored hash values where the SN stores *k* hash values per one neighbor which are calculated in advance at setup phase.

No.	Scheme	Storage Cost (bits)	Computation Cost
1	SAS	$2nN + 4nN_{ave} + 16$	No hashing operations
2	CAS	$6n + 7nN_{ave} + 16$	Two hashing orations and two encryptions
3	HIR	$2n + 2nN_{ave}$	One hashing function
4	RHIR	$2n + 2nN_{ave} + nkN_{ave}$	No hashing functions
5	APR	$9n + 7nN_{ave} + 2N - 2N_{ave} - 2$	Six hashing functions
6	DCARPS	3n	No hashing functions
7	ACS	$5nN_{ave}$	Two hashing functions
8	PhID	$(3n + 2) \times N_{ave}$	Four hashing function
9	EAC	$6n + 6nN_{ave} + 2$	Four hashing operations
10	FAC	$10n + (7n + 16) \times N_{ave}$	Four hashing operations & O_h encryptions

9.5. Processing and Computational Evaluation

Hash functions are used to calculate the pseudonyms and symmetric cryptography is used to encrypt the messages. Because we need to calculate three pseudonyms and one acknowledgement pseudonym after each transmission, using encryption to create pseudonyms was avoided since it requires more processing power compared to hash functions. When a SN senses data, it needs to calculate four OWH for PID, OHPID, and APID at the sender and OHPID at the receiver. If the system opts for data authentication, then another hash function is needed. The source node needs only one encryption for the data if $O_h = 0$, however, it needs O_h more encryptions if onion fashion is used. Each intermediary node needs one decryption operation and then another encryption to issue the new message. Chen *et al.* [16] indicates that SAS does not use hashing or encryption to create pseudonyms because it uses already created pseudonyms from a space. The other scheme by Chen *et al.* [16], CAS, uses two hashing operations and two encryption operations. APR uses at least six hashing functions. DCARPS uses constant IDs, so no hashing functions or encryptions for creating IDs. EAC has four hashing operations. It is clear that our framework needs a bit extra processing power due to the higher privacy and security we have achieved. None of the other schemes can achieve privacy against global threats and active adversary attacks. The power consumption due to the additional encryption operations is marginal compared to the power consumption caused by data transmission. Figure 29 exhibits different storage

size for different privacy schemes, which are discussed throughout this work. It shows that the increase in storage is *linear* and relatively comparable to the other protocols. The size of the storage would increase when the number of neighbors increases. Each SN has limited flash memory size, which could confine the maximum number of neighbors that a sensor can fit. As an example, *TelosB* mote [16,20] has 1 MB external flash memory. Thus, if one neighbor node requires 1.2 k bits of storage memory, then TelosB could fit more than 800 neighbors, which is very much more than what is needed in practical networks. Although FAC shows a bit of increase in the storage required to store the pseudonyms but it is the only one, among the discussed protocols in this work, provides a steady and functional anonymity and location privacy under strong global and active attack. In addition, the current technology provides sensors with sizable storage memory, which makes it not an issue at all.

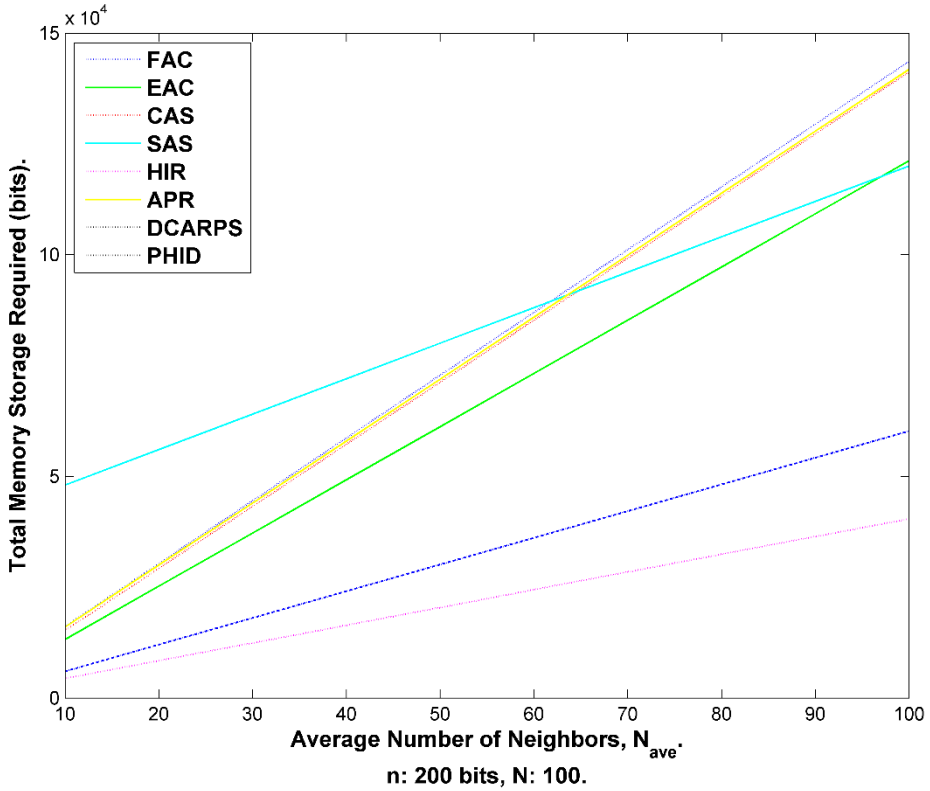


Figure 29. Size of storage memory using different privacy schemes.

10. Conclusions and Future Work

FAC is a modular framework that provides source, link and sink anonymity. It also provides temporal privacy and rate privacy. None of the previous related-work have a comprehensive solution for anonymity and location privacy. The three modules provided in FAC are made work together to prevent any statistical analysis attacks. The quadruple privacy (anonymity, temporal, rate, statistical) has provided a fortified SLP and BLP. FAC has addressed both local and global adversary. We have used a complex anonymity module where pseudonyms to replace real IDs are used. To provide temporal privacy both delays and fake messages are used. The use of fake messages was adjusted to manage the energy consumption. Two schemes are introduced, SGAT and ECAT. FAC is able to handle both homogenous and heterogeneous sensor nodes. FAC is both energy-aware and delay-aware. We have

demonstrated that FAC can withstand passive and active attacks by presenting scenarios and provided solutions. The memory cost was mathematically analyzed for the framework. The computational complexity for encryptions and hash functions was analyzed. To provide security for the BS against colluding active attacks, we have introduced onion encryptions. We have simulated the performance of the framework. The future work would include enhancement on the fake messages probabilistic scheme. In addition, we will implement FAC for different routing protocols such as clustered networks. We would plug FAC in some civil and military applications for further analysis, development and improvement.

Author Contributions

Abdel-shakour Abuzneid has conducted the research work and simulations. The rest of the authors contributed to framing the intellectual merit of the proposed contribution to the state-of-the-art. The article was written by Abdel-shakour Abuzneid.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Eu, Z.A.; Tan, H.-P.; Seah, W.K.G. Design and Performance analysis of MAC Schemes for Wireless Sensor Networks Powered by Ambient Energy Harvesting. *Ad Hoc Netw.* **2011**, *9*, 300–323.
2. Noh, D.K.; Hur, J. Using a dynamic backbone for efficient data delivery in solar-powered WSNs. *J. Netw. Comput. Appl*. **2012**, *35*, 1277–1284.
3. Li, Y.; Ren, J. Providing Source-Location Privacy in Wireless Sensor Networks. In *Wireless Algorithms, Systems, and Applications*; Liu, B., Bestavros, A., Du, D.-Z., Wang, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5682, pp. 338–347.
4. Conti, M.; Willemsen, J.; Crispo, B. Providing Source Location Privacy in Wireless Sensor Networks: A Survey. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 1238–1280.
5. Abbasi, A.; Khonsari, A.; Talebi, M.S. Source Location Anonymity for Sensor Networks. In Proceediongs of the 6th IEEE Consumer Communications and Networking Conference, Las Vegas, NV, USA, 10–13 January 2009.
6. Nezhad, A.A.; Miri, A.; Makrakis, D. Location privacy and anonymity preserving routing for wireless sensor networks. *Comput. Netw*. **2008**, *52*, 3433–3452.
7. Yao, L.; Kang, L.; Shang, P.; Wu, G. Protecting the sink location privacy in wireless sensor networks. *Pers. Ubiquitous Comput*. **2013**, *17*, 883–893.
8. Li, N.; Zhang, N.; Das, S.K.; Thuraisingham, B. Privacy preservation in wireless sensor networks: A state-of-the-art survey. *Ad Hoc Netw.* **2009**, *7*, 1501–1514.
9. Yan, Z.; Zhang, P.; Vasilakos, A.V. A survey on trust management for Internet of Things. *J. Netw. Comput. Appl.* **2014**, *42*, 120–134.
10. Jing, Q.; Vasilakos, A.; Wan, J.; Lu, J.; Qiu, D. Security of the Internet of Things: Perspectives and challenges. *Wirel. Netw.* **2014**, *20*, 2481–2501.

11. Kamat, P.; Zhang, Y.; Trappe, W.; Ozturk, C. Enhancing Source-Location Privacy in Sensor Network Routing. In Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS), Columbus, OH, USA, 10 June 2005.
12. Ozturk, C.; Zhang, Y.; Trappe, W.; Ott, M. Source-Location Privacy for Networks of Energy-Constrained Sensors. In Proceediongs of the Second IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, Vienna, Austria, 11–12 May 2004.
13. Jing, D.; Han, R.; Mishra, S. Countermeasures against Traffic Analysis Attacks in Wireless Sensor Networks. In Proceediongs of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks, Washington, DC, USA, 5–9 September 2005.
14. Ying, J.; Shigang, C.; Zhan, Z.; Liang, Z. A novel scheme for protecting receiver's location privacy in wireless sensor networks. *IEEE Trans. Wirel. Commun.* **2008**, *7*, 3769–3779.
15. Xinfeng, L.; Xiaoyuan, W.; Nan, Z.; Zhiguo, W.; Ming, G. Enhanced Location Privacy Protection of Base Station in Wireless Sensor Networks. In Proceedings of the 5th International Conference on Mobile Ad-hoc and Sensor Networks MSN, Fujian, China, 14–16 December 2009.
16. Chen, J.; Du, X.; Fang, B. An efficient anonymous communication protocol for wireless sensor networks. *Wirel. Commun. Mob. Comput.* 2012, *12*, 1302–1312.
17. Ouyang, Y.; Le, Z.; Liu, D.; Ford, J.; Makedon, F. Source Location Privacy against Laptop-Class Attacks in Sensor Networks. In Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks, Istanbul, Turkey, 22 September 2008; pp. 1–10.
18. Xiaoyan, H.; Pu, W.; Jiejun, K.; Qunwei, Z.; Jun, L. Effective Probabilistic Approach Protecting Sensor Traffic. In Proceedings of the IEEE Military Communications Conference (MILCOM), Atlantic City, NJ, USA, 17–20 October 2005.
19. Kamat, P.; Xu, W.; Trappe, W.; Zhang, Y. Temporal privacy in wireless sensor networks: Theory and practice. *ACM Trans. Sen. Netw.* **2009**, *5*, 1–24.
20. Xue, G.; Misra, S. Efficient anonymity schemes for clustered wireless sensor networks. *Int. J. Sens. Netw.* **2006**, *1*, 50–63.
21. Yi, O.; Zhengyi, L.; Yurong, X.; Triandopoulos, N.; Sheng, Z.; Ford, J.; Makedon, F. Providing Anonymity in Wireless Sensor Networks. In Proceedings of the IEEE International Conference on Pervasive Services, Istanbul, Turkey, 15–20 July 2007.
22. Jang-Ping, S.; Jehm-Ruey, J.; Ching, T. Anonymous Path Routing in Wireless Sensor Networks. In Proceedings of the IEEE International Conference on Communications (ICC), Beijing, China, 19–23 May 2008.
23. Xi, L.; Xu, J.; Myong-Soon, P. Location Privacy against Traffic Analysis Attacks in Wireless Sensor Networks. In Proceedings of the International Conference on Information Science and Applications (ICISA), Seoul, Korea, 21–23 April 2010.
24. Di Pietro, R.; Viejo, A. Location privacy and resilience in wireless sensor networks querying. *Comput. Commun.* **2011**, *34*, 515–523.
25. Park, J.-H.; Jung, Y.-H.; Ko, H.; Kim, J.-J.; Jun, M.-S. A Privacy Technique for Providing Anonymity to Sensor Nodes in a Sensor Network. In *Ubiquitous Computing and Multimedia Applications*; Kim, T.-H., Adeli, H., Robles, R.J., Balitanas, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 327–335.

26. Juan, C.; Hongli, Z.; Binxing, F.; Xiaojiang, D.; Lihua, Y.; Xiangzhan, Y. Towards Efficient Anonymous Communications in Sensor Networks. In Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM), Houston, TX, USA, 5–9 December 2011.
27. Abuzneid, A.; Sobh, T.; Faezipour, M. An Enhanced Communication Protocol for Anonymity and Location Privacy in WSN. In Proceedings of the IEEE Wireless Communications and Networking Conference, New Orleans, LA, USA, country, 9–12 March 2015.
28. Kong, J.; Hong, X. *ANODR:* Anonymous on Demand Routing with Untraceable Routes for Mobile ad-hoc Networks. In Proceedings of the 4th ACM International Symposium on Mobile ad hoc Networking & Computing, Annapolis, MA, USA, 1–3 June 2003; pp. 291–302.
29. Kerckhoffs' Principle. Available from: http://en.citizendium.org/wiki/Kerckhoffs%27_Principle (accessed on 1 March 2015).
30. Yanchao, Z.; Wei, L.; Wenjing, L.; Yuguang, F. Location-based compromise-tolerant security mechanisms for wireless sensor networks. *IEEE J. Sel. Areas Commun.* **2006**, *24*, 247–260.
31. Lu, R.; Lin, X.; Zhang, C.; Zhu, H.; Ho, P.; Shen, X. AICN: An Efficient Algorithm to Identify Compromised Nodes in Wireless Sensor Network. In Proceedings of the IEEE International Conference on Communications (ICC), Beijing, China, 19–23 May 2008.
32. Song, H.; Xie, L.; Zhu, S.; Cao, G. Sensor Node Compromise Detection: the Location Perspective. In Proceedings of the 2007 International Conference on Wireless Communications and Mobile Computing, Honolulu, HI, USA, 12 August 2007; pp. 242–247.
33. Tao, L.; Min, S.; Alam, M. Compromised Sensor Nodes Detection: A Quantitative Approach. In Proceedings of the 28th International Conference on Distributed Computing Systems Workshops (ICDCS), Beijing, China, 17–20 June 2008.
34. Alomair, B.; Clark, A.; Cuellar, J.; Poovendran, R. Toward a Statistical Framework for Source Anonymity in Sensor Networks. *IEEE Trans. Mob. Comput*. **2013**, *12*, 248–260.
35. Zhu, S.; Setia, S.; Jajodia, S. LEAP+: Efficient security mechanisms for large-scale distributed sensor networks. *ACM Trans. Sen. Netw*. **2006**, *2*, 500–528.
36. Mabrouki, I.; Belghith, A. E-SeRLoc: An Enhanced Serloc Localization Algorithm with Reduced Computational Complexity. In Proceedings of the 9th International Wireless Communications and Mobile Computing Conference (IWCMC), Sardinia, Italy, 1–5 July 2013.
37. Lazos, L.; Poovendran, R. SeRLoc: Secure Range-Independent Localization for Wireless Sensor Networks. In Proceedings of the 3rd ACM Workshop on Wireless Security, 1 October 2004; ACM: Philadelphia, PA, USA, 2004; pp. 21–30
38. Zheng, W.; Gao, S.; Qiu, L.; Zhang, W. A CDS-based Topology Control Algorithm in Energy Efficient Clustering. In Proceedings of the 31st Chinese Control Conference (CCC), Hefei, China, 25–27 July 2012.
39. Hongwei, D.; Weili, W.; Qiang, Y.; Deying, L.; Wonjun, L.; Xuepeng, X. CDS-Based Virtual Backbone Construction with Guaranteed Routing Cost in Wireless Sensor Networks. *IEEE Trans. Parallel Distrib. Syst*. **2013**, *24*, 652–661.
40. Abduvaliyev, A.; Pathan, A.S.K.; Jianying, Z.; Roman, R.; Wai-Choong, W. On the Vital Areas of Intrusion Detection Systems in Wireless Sensor Networks. *IEEE Commun. Surv. Tutor*. **2013**, *15*, 1223–1237.

41. YangXia, L.; Ye, G. A Survey on Intrusion Detection of Wireless Sensor Network. In Proceedings of the 2nd International Conference on Information Science and Engineering (ICISE), Hangzhou, China, 4–6 December 2010.
42. Abuzneid, A.; Sobh, T.; Faezipour, M. Temporal Privacy Scheme for End-to-End Location Privacy in Wireless Sensor Networks. In IEEE Electrical, Electronics, Signals, Communiction & Optimization, Visakhapatnam, Andhra Pradesh, India, 24–25 January 2015; pp. 2476–2481.
43. Mehta, K.; Donggang, L.; Wright, M. Location Privacy in Sensor Networks Against a Global Eavesdropper. In Proceedings of the IEEE International Conference on Network Protocols (ICNP), Beijing, China, 16–19 October 2007.
44. Alomair, B.; Clark, A.; Cuellar, J.; Poovendran, R. Statistical Framework for Source Anonymity in Sensor Networks. In Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM), Miami, FL, USA, 6–10 December 2010.
45. Wu, P.-C. Multiplicative, congruential random-number generators with multiplier. *ACM Trans. Math. Softw*. **1997**, *23*, 255–265.
46. Deng, L.-Y.; Rousseau, C.; Yuan, Y. Generalized Lehmer-Tausworthe Random Number Generators. In Proceedings of the 30th Annual Southeast Regional Conference, Raleigh, NC, USA, 8 April 1992; pp. 108–115.
47. Payne, W.H.; Rabung, J.R.; Bogyo, T.P. Coding the Lehmer pseudo-random number generator. *ACM Commun.* **1969**, *12*, 85–86.
48. Heinzelman, W.R.; Chandrakasan, A.; Balakrishnan, H. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Hawaii, HI, USA, 4–7 January 2000.
49. Abuhelaleh, M.A.; Mismar, T.M.; Abuzneid, A.A. Armor-LEACH—Energy Efficient, Secure Wireless Networks Communication. In Proceedings of 17th International Conference on Computer Communications and Networks (ICCCN), St. Thomas, USVI, USA, 3–7 August 2008.
50. Stallings, W. *Network Security Essentials Applications and Standards*, 5th ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2014.