WAS Science Nature
WASSN

ISSN: 2766 - 7715
WAS PRESS

Article

Evaluation of a Link Layer Secure Protocols for WSNs

Ahmed M. Mabrouk [1,*]

[1] Department of Electronics and communications Engineering, Higher Institute of Engineering, Elshorouk Academy, Cairo, Egypt; a.mabrouk@sha.edu.eg

*Correspondence: a.mabrouk@sha.edu.eg

Received: 15-10-2020; Accepted: 18-11-2020; Published: 15-12-2020

Abstract: Communication energy is the main source of energy consumption for WSNs. all encryption protocols proposed for WSNs in the literature aim to reduce the security-related communication overhead in different ways to save communication energy. In other words, all proposed energy-efficient protocols affect security strength factors. However, ensuring communications security in WSNs indeed is critical. WSNs suffer from limited bandwidth, memory, and processing capabilities. They operate on the unattended mode in hostile environments. Therefore, an energy-efficient link-layer security framework is necessitated. In this paper; most link-layer security protocols are evaluated in terms of network lifetime and the end to end delay.

Keywords: Security protocol; encryption; data link layer; WSNs.

1. Introduction

A wireless sensor network (WSN) is an essential wireless network nowadays. WSN consists of a specific amount of sensors called sensor nodes. These nodes are classified into source nodes, sink nodes, and intermediate nodes or relay nodes. All sensor nodes have a function of sensing and monitoring a physical environment conditions such as humidity, pressure, temperature, and perform simple processing. Each sensor node located in the WSN communicates with each other and exchanges its data via wireless multi-hop i.e. each sensor node routes its data to the neighbor node until it reaches the sink node. Sensor nodes in WSNs have the following limitations: bandwidth, battery lifetime, and storage resources [1]. Most of the energy of the sensor nodes are consumed in communication between nodes. Hence, most of the recent researches trying to overcome these limitations of energy consumption. WSNs have several civilian applications smart homes, monitoring the instruments in the factories, monitoring the temperature, humidity, and water level in farms or greenhouses WSN has also secured applications such as military, battlefield environments so, data security is an essential aspect for secure WSNs. Data security and communication energy are the main aspects of modern link layer architecture designers; because both data protection and energy consumption during communication perform the main source of power consumption in WSNs. Achieving a higher security level and increasing data communication need extra processing which produces overhead in power consumption. Because of the communication between nodes taking place "through the air" the transmitted data will be more subjected to message injecting and altering by an attacker. Hence, the messages must be encrypted with a strong algorithm [1].

In this paper, data link protocols for WSNs are compared and evaluated. The comparison is based on the security services that each protocol provides the advantages and disadvantages, packet structure, the protocol modes if it exists, and ciphering primitives that are used. The reference for

most of the previous protocols depends on the Tiny OS operating system. In section 2 Tiny OS is described. Several link layer protocols are briefly described in section 3. The protocols are evaluated in terms of network lifetime and an end to end delay in section 4. Finally, the conclusion is given in section 5.

2. The Tiny OS

The operating system of WSNs is completely different from other wired traditional networks due to the serious challenges of WSNs. such as limitations of energy (battery), limited resources of memory, and cost issue These challenges lead us to fulfill the minimum requirements to perform an operating system for WSNs. Like synchronization, scheduling, re-programmability, and the link between all components in the system [3]. The application programming interfaces (APIs) [3] must be presented in the operating system for sensing and management of memory, power, and process APIs.one of the advantages of Tiny OS [4] that it's an open-source operating system used for WSNs in several applications such as commercial sensor nodes called "motes".in addition it's compatible with several types of hardware so it has wide use in industrial and academic researches applications. A based programming model code is used in Tiny OS programmed by nesC language [5]. One of the advantages of this language that it operates in limited resources and low memory so it's suitable for WSNs. The basic core of the Tiny OS code size is 400 byte only. And at least 16KB of memory is suitable for most of the applications due to efficient and low power of operation benefits.

The queuing schedule of the Tiny OS is the First Input First Output "FIFO" technique. This technique has advantages in simplifying the race condition and handling of the deadlock but its drawback appears in applications that require a long time for running its tasks which leads to reduce system response in addition to delaying the tasks that have a high priority[6]. The synchronization in Tiny OS is used at link-level and needs an acknowledgment after each transmission i.e. the next message doesn't send until the source node receiving an acknowledge to send the next message which is called Stop-And-Wait or alternate-bit Automatic repeat request (ARQ). Most WSNs apply this technique for synchronization in a communication protocol that's because most WSNs applications need an efficient power consumption, minimum communication bandwidth, and an accepted immunity against latency that the packet suffers from[7],[8]. WSNs have many sources of power consumption but power consumption in communication is more efficient than sensing and computation. One of the major parameters that reduce energy consumption is to reduce packet size which is the main target of most of the researches that are interested in efficient power consumption in link layer protocols that will be described in the next section[9].

The frame structure of the Tiny OS is shown in figure1. This frame structure is used as a reference for related ciphering protocols. Tiny OS maximum frame size is 43 byte divided into the maximum payload of 30 bytes and 13 bytes for both header and trailer fields. The header field consists of the preamble which has a 6-byte size, two bytes for a destination address (Dest), one byte for the Active Message (AM) field which has the same function as the port number in TCP/IP networks. The length of the message has one byte (len). Finally, to differentiate between networks there is a group field with one-byte size as a unique network identifier. The trailer of the frame format of TinyOS is a 2-byte size for Cyclic Redundancy Check (CRC) [4].

preamble	Des	AM	Len	Grp	payload	CRC
6 bytes	2bytes	byte	1byte	1 byte	0-30 bytes	2 bytes

Figure 1. Tiny OS frame structure.

Tiny OS didn't support any level of security Hence, we have a serious tradeoff between energy consumption and security services supposed to be performed because when we add any fields for security it will be at the expense of packet size, therefore, it increases the power consumption hence,

its suppose to have a balanced protocol that compromises between energy consumption and level of security that it provides [4].

3. Encryption protocols overview

In WSNs the security of the link layer is more important than the traditional network that's because the messages transmitted from any node until it reaches the sink node have a great amount of similarity so the sensor node collects this data traffic to minimize the communication overhead and energy consumption. The end-to-end security technique is completely different in WSN than the traditional network not only the data required to be examined and collected through its journey to the sink nodes but also link-layer supposed to reduce the security attack effect and it can discover the effect of security attacks on the next hop when it occurs. If the end-to-end security scheme is used, it will be detected at the sink node. In this section, many works of literature encryption protocols for WSN are reviewed. It's supposed to evaluate the encryption protocols concerning the level of security that each provides. Whenever any protocol reduces the level of security that it provides to reduce the communication overhead it will be more subjected to attack. Hence, most of the researches in encryption protocol design aims to provide all basic security services with minimum header size in different ways [1].

3.1. Zigbee protocol

Zigbee Protocol is [10] not only used for Personal Area Network (PAN) but also it is suitable for many other applications that require relatively low data rates less than 250Kbps, high security of the network, and long lifetime of the battery. In addition to low price and high efficiency in implementation to perform the encryption scheme, Hardware modules are needed in most Zigbee applications which are more efficient in computation. Zigbee has a relatively high level of security such as confidentiality and authentication that's implemented by encrypting the payload with Advanced Encryption Standard counter with cipher block chaining message authentication code modes (AES-CCM). The frame format for the Zigbee is shown in figure 2. It adds new fields compared with TinyOS frame formats such as source address (src) field which has a two-byte size, four bytes for Message Authentication Code (MAC), and another four bytes for the counter field to investigate frame freshness and replay protection. Zigbee frame format removes the group field (Grp) that was in TinyOS because its function is implemented in the keying scheme that occurs in MAC. This scheme guarantees differentiation between networks and access control services [10].

preamble	Des	Src	AM	Len	Ctr	Payload	CRC	MAC
6 bytes	2byte	2 bytes	1byte	1 byte	4 bytes	0-30 bytes	2 bytes	4 bytes

Figure 2. Zigbee frame structure.

3.2. Tinysec Protocol

TinySec protocol [11] was implemented as a part of TinyOS is a link-layer security structure. In this protocol, the designers compromise between performance and the level of security and produce into two modes of operation. The first mode is TinySec-Auth and the second is TinySec-AE. TinySec-Auth mode decreases the frame size at the expense of the low level of security by providing only integrity of the message and authentication as shown in figure 3.a. TinySec-Auth mode produces a modification in TinyOS by adding four bytes as MAC field that computed using CBC-MAC under Skipjack algorithm [12] on the frame header and payload. The second mode TinySec-AE increases the frame size to support more security services such as message confidentiality and replay protection in addition to the services that occur by adding two bytes as freshness counter as shown in figure3.b. TinySec-AE concatenates the destination address (Des), Active message (AM), message length (Len), source address (src), and freshness counter (ctr) as the initialization vector for CBC mode. TinySec-

AE chooses the skipjack algorithm due to light software implementation in the code size point of view compared with other encryption algorithms.

The encryption key of TinySec-AE is 80 bit and its block size is 64 bit. The key size is relatively small which is more subjected to the brute force attack. In existing of the MAC field not only group field (Grp) has been removed because its function was implemented in the keying mechanism in the MAC field but also CRC has been removed. Once the MAC field is used it becomes responsible for detecting the transmission errors instead of CRC. Both TinySec modes frame format is shown in Figures 3.a and 3.b respectively. The frame size of TinySec-Auth is 44 bytes which are larger than TinyOS by one byte only. The TinySec-AE is used as a reference for other protocols because it supports all basic security services. TinySec-AE mode adds three fields on a TinyOS consists of four bytes for MAC field, two bytes for freshness counter (ctr), and another two bytes for source address in addition to removing group field and CRC fields to be 48 bytes which are larger than TinyOS by four bytes [11].

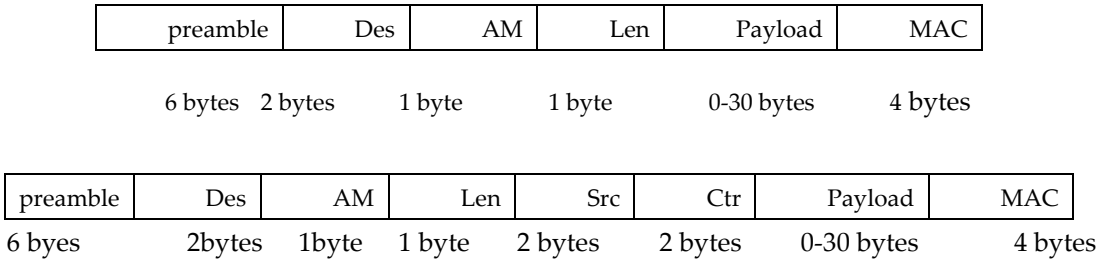

Figure 3. a) TinySec-Auth packet, b)TinySec-AE packet.

3.3. Secure Network Encryption Protocol (SNEP)

SNEP protocol supports data confidentiality, data freshness, and two parity authentication data in small overhead in packet size and it is designed in a partial of SPIN [13]. Rivest Cipher 5 (RC5) was chosen for SNEP in counter mode to perform data confidentiality of 64-bit block size and a key size of 128 bit in the encryption algorithm. The advantage of using (RC5) instead of AES because RC5 has a light code size in software implementation in addition to using CBC-MAC for the MAC field to support integrity and data authentication. In SNEP, there are two levels of freshness, weak freshness, and strong freshness. Weak freshness is automatically applied through RC5 encryption in counter mode. When the sender transmitted one message, it increases the counter one time and the receiver ensures that the message that received has a unique increasing counter. The scenario of the strong freshness is started when the sender generates 64 random bits and transmits it to the receiver node in a form of the request message. The receiver begins to use these bits in MAC computation and create a response message according to this computation. If the MAC computation of the response message is successfully verified, then the sender ensures that the receiver creates this response message and then strong freshness is implemented [13].

SNEP protocol saves the packet size not only by removing the group field (Grp) and CRC fields but also it removes two bytes of counter field (Ctr) because SNEP protocol updates the counter field in both transmitter and receiver. Then there is no need to transmit it in the packet. SNEP protocol use 8 bytes as a MAC field for more security that's because the attacker needs at least 264 trials to get a valid one, in other words, it needs 263 trials as an average to get the valid one which is more difficult. Figure 4 shows the frame format of SNEP protocol. Although SNEP saves 5 bytes of counter field (Ctr), group field (Grp), and CRC field it increases the source address field and twice the MAC field (8-bytes) compared with other protocols that make SNEP protocol exceed the plain TinyOS by 7 bytes [13].

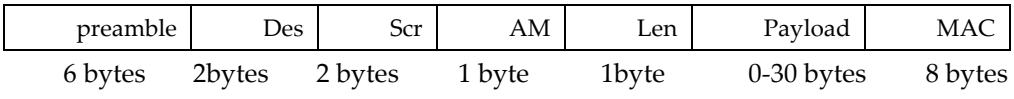

Figure 4. SNEP frame structure

3.4. TinySec protocol

TinySec protocol [11] was implemented as a part of TinyOS in the link-layer security structure. In this protocol, the designers compromise between performance and the level of security and produce into two modes of operation. The first mode is TinySec-Auth and the second is TinySec-AE. TinySec-Auth mode decreases the frame size at the expense of the low level of security by providing only integrity of the message and authentication as shown in figure 3.a. TinySec-Auth mode produces a modification in TinyOS by adding four bytes as MAC field that computed using CBC-MAC under Skipjack algorithm [12] on the frame header and payload. The second mode TinySec-AE increases the frame size to support more security services such as message confidentiality and replay protection in addition to the services that occur by adding two bytes as freshness counter as shown in figure3.b. TinySec-AE concatenates the destination address (Des), Active message (AM), message length (Len), source address (src), and freshness counter (ctr) as the initialization vector for CBC mode. TinySec-AE chooses the skipjack algorithm due to light software implementation in the code size point of view compared with other encryption algorithms.

The encryption key of TinySec-AE is 80 bit and its block size is 64 bit. The key size is relatively small which is more subjected to the brute force attack. In existing of the MAC field not only group field (Grp) has been removed because its function was implemented in the keying mechanism in the MAC field but also CRC has been removed. Once the MAC field is used it becomes responsible for detecting the transmission errors instead of CRC. Both TinySec modes frame format is shown in Figures 3.a and 3.b respectively. The frame size of TinySec-Auth is 44 bytes which are larger than TinyOS by one byte only. The TinySec-AE is used as a reference for other protocols because it supports all basic security services. TinySec-AE mode adds three fields on a TinyOS consists of four bytes for MAC field, two bytes for freshness counter (ctr), and another two bytes for source address in addition to removing group field and CRC fields to be 48 bytes which are larger than TinyOS by four bytes [11].

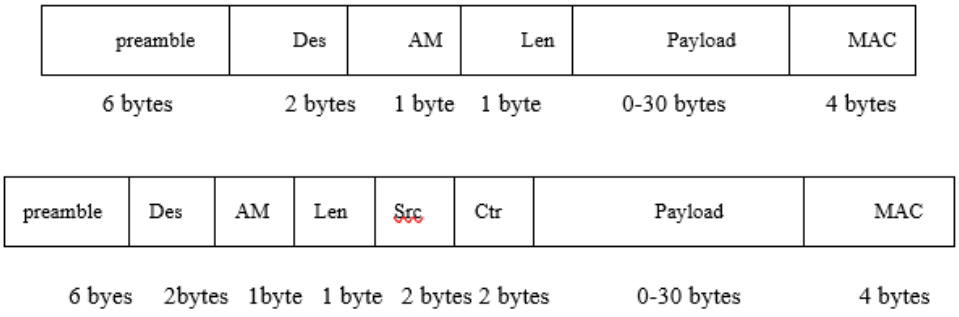

Figure 3. a) TinySec-Auth packet b)TinySec-AE packet.

3.5. Secure Network Encryption Protocol (SNEP)

SNEP protocol supports data confidentiality, data freshness, and two parity authentication data in small overhead in packet size and it is designed in a partial of SPIN [13]. Rivest Cipher 5 (RC5) was chosen for SNEP in counter mode to perform data confidentiality of 64-bit block size and a key size of 128 bit in the encryption algorithm. The advantage of using (RC5) instead of AES because RC5 has a light code size in software implementation in addition to using CBC-MAC for the MAC field to support integrity and data authentication. In SNEP, there are two levels of freshness, weak freshness, and strong freshness. Weak freshness is automatically applied through RC5 encryption in counter mode. When the sender transmitted one message, it increases the counter one time and the receiver ensures that the message that received has a unique increasing counter. The scenario of the strong freshness is started when the sender generates 64 random bits and transmits it to the receiver node in a form of the request message. The receiver begins to use these bits in MAC computation and create a response message according to this computation. If the MAC computation of the response message

is successfully verified, then the sender ensures that the receiver creates this response message and then strong freshness is implemented [13].

SNEP protocol saves the packet size not only by removing the group field (Grp) and CRC fields but also it removes two bytes of counter field (Ctr) because SNEP protocol updates the counter field in both transmitter and receiver. Then there is no need to transmit it in the packet. SNEP protocol use 8 bytes as a MAC field for more security that's because the attacker needs at least 264 trials to get a valid one in other words in needs 263 trials as an average to get the valid one which is more difficult. Figure 4 shows the frame format of SNEP protocol. Although SNEP saves 5 bytes of counter field (Ctr), group field (Grp), and CRC field it increases the source address field and twice the MAC field (8-bytes) compared with other protocols that make SNEP protocol exceed the plain TinyOS by 7 bytes [13].

preamble	Des	Scr	AM	Len	Payload	MAC
6 bytes	2bytes	2 bytes	1byte	1 byte	0-30 bytes	8 bytes

Figure 4. SNEP packet [13]

3.6. MiniSec protocol

Offset Code Book (OCB) [14] is one of the encrypted modes that perform both confidentiality and authentication simultaneously in one path which is used in MiniSec protocol [15] because of its run encryption algorithm rather than one time running it for authentication and runs it again for confidentiality. Skipjack algorithm with a block size of 64 bits and 80 bits encryption key is chosen for MiniSec protocol but MiniSec designers recommend using AES as a software implementation due to the weakness of Skipjack [16].MiniSec protocol has two modes of operation: MiniSec-U and MiniSec-B for unicast and broadcast communication respectively. In MiniSec-U, the designer tries to compromise between transmitting the counter field as described in TinySec and not transmit it like SNEP protocol. There is a problem in sending the counter freshness field in TinySec leads to increasing overhead in communication due to sending 2 bytes for the counter field. On the other hand, there is a serious problem in SNEP protocol leads to an absence of counter synchronization in case of the wrong update of the counter due to packet drop or packet retransmission hence, the MiniSec-U designers transmit the least 3 bits of the counter into each transmitted packet then it maintains the synchronization between the sender and receiver in addition to that the energy consumption is approximately not changed compared with no transmit of counter field. In broadcast communication, MiniSec-B is supposed to maintain resynchronization among the receivers which are expensive because the receivers simultaneously receive a large number of packets from different senders nodes the receiver's node is supposed to keep the freshness counter for each received packet which requires a large memory capacity. Hence, MiniSec-B produces a solution summarized in two approaches. In the first approach, the time is divided into small intervals (epochs) which are called a sliding window approach and the receiver drops the packet that has older epochs. In addition to restricting the replay attack in one epoch of the window. The second approach produces Bloom Filters (BFs) [17]. In broadcast communication ensure that each is a member of the set or not so, a space-efficient probabilistic data structure is used. The scenario started when the received packet is stored in an attached BFs. The two most recent epochs are stored by every node. Hence, if the received packet does not exist in any of the BFs, then the receiver accepts this packet but if the received packet exists in the BFs then the receiver drops this packet because it's sure a replayed packet.

To summarize this protocol MiniSec saves the packet size by using the most significant bit of some fields of the header to transmit a part of the counter. In MiniSec-U 3 bits are inserted to the header instead of 3 most significant of length message field for freshness counter. On the other hand, MiniSec-B inserts 5 bits and 3 bits instead of most significant bits (MSB) for destination address and length message fields respectively as shown in figure 5a, b. It means that the MiniSec-B is more secured compared with MiniSec-U because it uses one byte for freshness counter compared with 3 bits in MIniSec-U. it was Observed that the two modes have the same size but it is greater than the maximum frame size of TinyOS by three bytes [15].

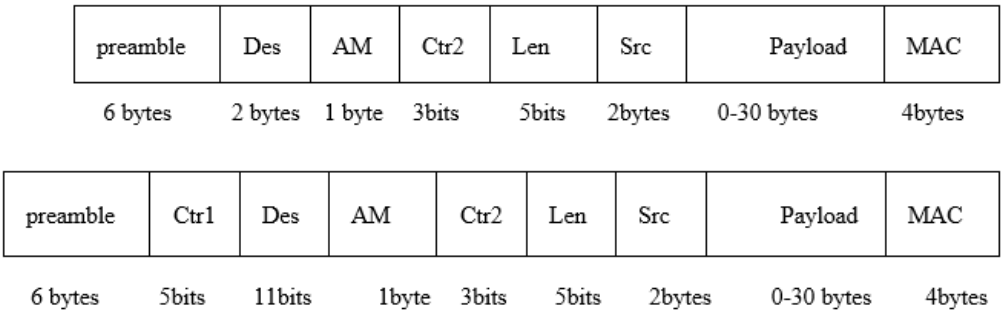

Figure 5. a) MiniSec-U. b)MiniSec-B packets [15]

3.7. Link-Layer Security protocol (LLSP)

AES encryption in CBC mode is chosen for the LLSP protocol [18] to guarantee message confidentiality. CBC-MAC supports message authentication. LLSP similar to SNEP protocol except the MAC field is four bytes instead of eight bytes for saving the packet size. The counter field has been updated in the transmitter and receiver node and does not transmit it like SNEP protocol. LLSP concatenates Des, AM, len, Scr, and Ctr fields to perform the initialization vector of the CBC-MAC. By reducing four bytes from the MAC field LLSP has a frame size of 46 bytes which is greater than TinyOS by 3 bytes as shown in figure 6.

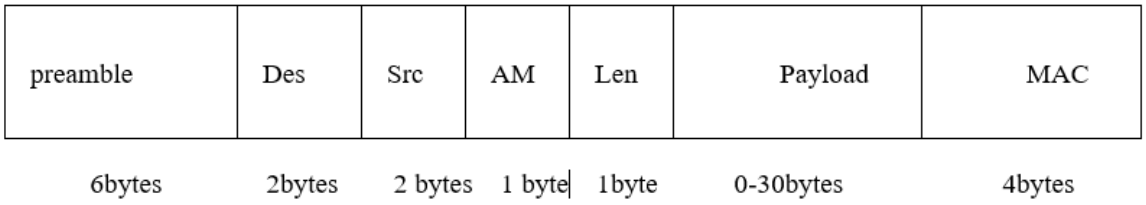

Figure 6. LLSP packet structure [18]

3.8. SenSec protocol

SenSec protocol [19] produces a new encryption algorithm called Skipjack-X as a modification for the Skipjack algorithm in addition to CBC-X mode which provides in one step both confidentiality and authentication. And Skipjack-X also introduces a padding technique for non-fixed messages size. The initialization vector of the CBC-X mode combines the Grp field that is kept in TinyOS, with Des, AM, Ran. Ran field is defined as a random number transmitted inside the packet. When encrypting the header of the first packet with zero fields Ran the first Ran is created. Then, the ciphertexts with at least three significant bytes are used to get it. The three-byte with Ran field of the next packet is determined by getting the MAC three significant bytes of the previous frame. The frame format of SenSec is shown in figure 7. Which exceeds TinyOS by 5 bytes.

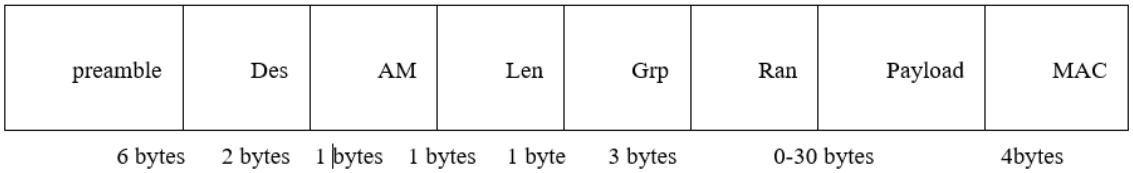

Figure 7. Packet structure of SenSec [19]

4. Performance Evaluations

In this section, different experiments are conducted to evaluate the performance and validate the effectiveness of the evaluated protocols [4], [10], [11], [13], [14], [18], [19]. The section starts by describing the performance metrics followed by a simulation environment and finally simulation results.

4.1. Performance Metric

For comprehensive performance evaluation, several quantitative metrics are defined below.

- Network Lifetime [20]: It is defined as the time duration from the beginning of the network operation until the first node exhausts its battery.
- Throughput Ratio (TR): This metric is defined as [21].

$$TR = \frac{Number\ of\ \ packets\ \text{correctly}\ received\ by\ the\ \text{sin}\ k}{Number\ of\ \ packets\ sent\ by\ source\ nodes}$$

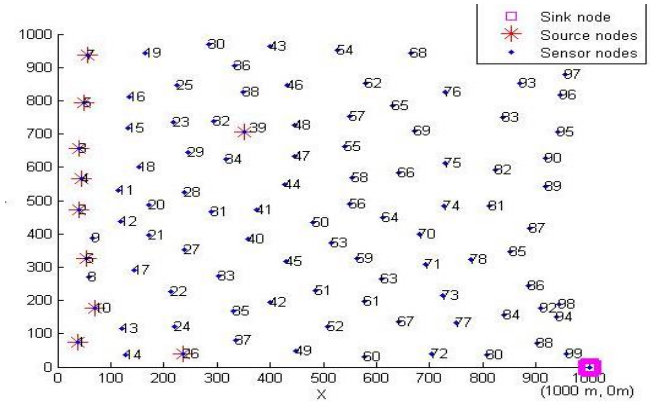

Figure 8. Network distribution

Average End-to-End Delay (Seconds) [22]: It is defined as the average time a packet takes to travel from the source node to the sink node. This includes propagation, transmission, queuing, and processing delay. The processing delay can be ignored as a result of the fast processing speed [23].

4.2. Simulation Environment

To simulate our proposal in this paper, we built our simulator in Matlab. 100 sensors are deployed in the monitored field. These sensors are randomly distributed in an area of 1000 m × 1000 m while the sink node is located at (1000 m, 0 m) as shown in figure 8. It is also assumed that there are no mobile nodes deployed in the monitored field and all the sensors including the sink node are stationary during all of our experiments. The later experiments are carried out for homogeneous node energy distributions. The network traffic is generated according to the Poisson process of intensity λ packets per second. Also, a practical wireless channel model is chosen which has shadowing, deep

fading, and noise effects. For better evaluation of the network performance, an efficient data routing technique in [24] is considered. The main reason for choosing this routing technique is that the design of this technique was not only to prolong the lifetime of the network but also to send the packets through reliable paths. Moreover, it is designed to prevent the nodes buffer overflow and balance the packets delivered to each node. The simulation parameters are listed in the following table.

Table 1. Simulation environment parameters

Parameters	Values
Network size	1000 m×1000 m
Number of nodes	100
Number of sink nodes	1
Node distribution	Random
Data rate	250 Kbps
Buffer size	128 byte
Frequency	2400 MHz
Transmission power	0 dBm
Maximum transmission range	224 m
Path loss exponent	4
Shadow fading variance	4
Noise power	-146 dBm
Reference distance	1 m
λ	5

5. Simulation results

In this section, a set of experiments are conducted to evaluate the performance of the Zigbee [10], SNEP [13],TinyOS [4], TinySec-Auth [11], TinySec-AE [11], LLSP [18], and SenSec [19]. Throughout this simulation is analyzed and compared to evaluate the performance of these security protocols in terms of network lifetime and average end-to-end delay for the homogeneous network. In all later experiments, each node is assumed to have initial energy of 125 mJ.

5.1. Evaluation of Network Lifetime

This experiment is conducted for security protocols [4, 10, 11, 13, 14, 18, and 19] evaluation in terms of network lifetime in homogenous network. The network lifetime of several security protocols are compared throughout this experiment including: 1) TinyOS [4], 2) Zigbee [10], 3) TinySec-Auth [11], 4) TinySec-AE presented in [11], 5) LLSP [18], 6) SenSec [19], 7) SNEP [13], 8) MiniSec-U [14], and 9) MiniSec-B [14].

From the results in Figure 9, it is clear that the network lifetime increases as the packet size decreases and vice versa. This is attributed to the fact that decreasing the packet size often decreases the energy required for transmitting the same amount of data. However, the figure shows clearly that the TinyOS has a longer lifetime compared to the other algorithms. The reason for such a result is that TinyOS didn't support any level of security where adding any level of security services leads to more packet size and hence more energy consumption. On the other hand, the Zigbee protocol has the lowest lifetime. That's because it provides a high level of security services. From the simulation results and analysis of this metric, it can be concluded that the Zigbee security protocol is not

recommended for WSNs Since the energy resource in WSN is considered the main extremely critical resource [25, 26].

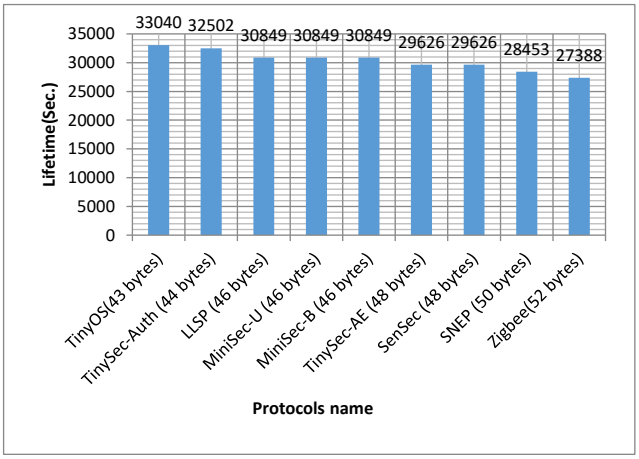

Figure 9. network lifetime for homogeneous network.

5.2. *Average End-to-End Delay Evaluation*

Throughout this section, another experiment is conducted regarding average End-to-End delay evaluation of various security protocols including TinyOS [4], Zigbee [10], TinySec-Auth [11], TinySec-AE [11], LLSP [18], SenSec [19], SNEP [13], MiniSec-U [14], and MiniSec-B [14]. It can be observed from Figure 10 that the end-to-end delay increases as the packet size increases. This can be justified as follow. As the packet size decreases, the queuing delay and transmission delay decrease leading to a lower end-to-end delay. Although the TinyOS has the lowest end-to-end delay compared to the others as illustrated in figure 9, it didn't provide any level of security. On the contrary, Zigbee has the highest end-to-end delay. Regardless of the high level of security provided by the Zigbee protocol, it is not suitable for real-time communication in WSNs.

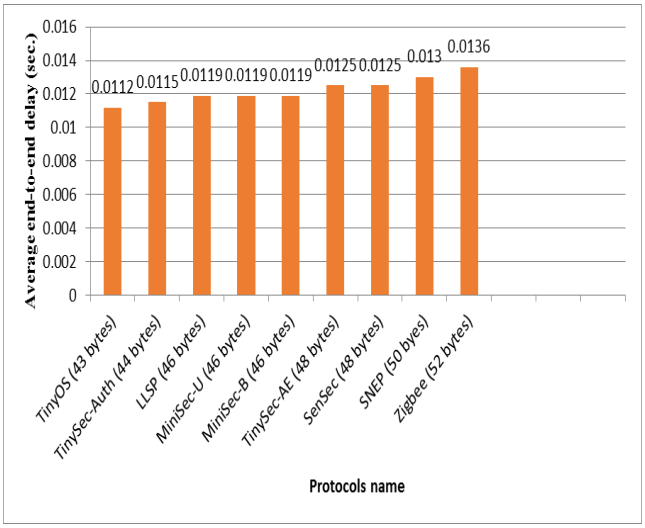

Figure 10. Average end-to-end delay for homogeneous network

6. Conclusion

In this paper, data link protocols for WSNs are compared and evaluated. The comparison is based on the security services that each protocol provides the advantages and disadvantages, packet structure, the protocol modes if it exists, and ciphering primitives that are used. The evaluation of these security protocols in terms of network lifetime and average end-to-end delay for a

homogeneous network is carried out. The simulation shows that the network lifetime increases as the packet size decreases and vice versa. And the end-to-end delay increases as the packet size increases.

References

1. Akyildiz, I. F., & Vuran, M. C. (2010). Wireless sensor networks (Vol. 4). John Wiley & Sons.
2. Niewiadomska-Szynkiewicz, E., Kwaśniewski, P., & Windyga, I. (2009). Comparative study of wireless sensor networks energy-efficient topologies and power save protocols. Journal of Telecommunications and information technology, 68-75.
3. Hassan, M. M., Ramadan, R. A., & El Boghdadi, H. M. (2014). Finding the best sink location in WSNs with reliability route analysis. Procedia Computer Science, 32, 1160-1167.
4. Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., ... & Culler, D. (2005). TinyOS: An operating system for sensor networks. In Ambient intelligence (pp. 115-148). Springer, Berlin, Heidelberg.
5. Gay, D., Levis, P., Von Behren, R., Welsh, M., Brewer, E., & Culler, D. (2003). The nesC language: A holistic approach to networked embedded systems. Acm Sigplan Notices, 38(5), 1-11.
6. Bloch, R., & Wattenhofer, R. (2006). Enhanced Task Scheduling in TinyOS 2.0 & Channel Allocation in AMUHR. Semester Thesis, Distributed Computing Group.
7. Labrador, M. A., & Wightman, P. M. (2009). Topology Control in Wireless Sensor Networks: with a companion simulation tool for teaching and research. Springer Science & Business Media.
8. Rogaway, P., Bellare, M., & Black, J. (2003). OCB: A block-cipher mode of operation for efficient authenticated encryption. ACM Transactions on Information and System Security (TISSEC), 6(3), 365-403.
9. Ramadan, R. A., Alreshidi, E. J., & Sharif, M. H. (2020, October). Energy Efficient Framework for Fog Network Based on Multisink Wireless Sensor Networks. In 2020 2nd International Conference on Computer and Information Sciences (ICCIS) (pp. 1-5). IEEE.
10. Ramadan, R. A., Sharifa, M. H., & Salem, M. S. (2020, August). SIoT: Secure IoT Framework for Smart Environments. In International Conference for Emerging Technologies in Computing (pp. 51-61). Springer, Cham.
11. Karlof, C., Sastry, N., & Wagner, D. (2004, November). TinySec: a link layer security architecture for wireless sensor networks. In Proceedings of the 2nd international conference on Embedded networked sensor systems (pp. 162-175).
12. Dworkin, M. (2007). NIST Special Publication 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. US National Institute of Standards and Technology http://csrc. nist. gov/publications/nistpubs/800-38D/SP-800-38D. pdf.
13. Moh'd, A., Aslam, N., Robertson, W., & Phillips, W. (2012, January). C-Sec: Energy efficient link layer encryption protocol for Wireless Sensor Networks. In 2012 IEEE Consumer Communications and Networking Conference (CCNC) (pp. 219-223). IEEE.
14. Luk, M., Mezzour, G., Perrig, A., & Gligor, V. (2007, April). MiniSec: a secure sensor network communication architecture. In 2007 6th International Symposium on Information Processing in Sensor Networks (pp. 479-488). IEEE.
15. Rogaway, P., Bellare, M., & Black, J. (2003). OCB: A block-cipher mode of operation for efficient authenticated encryption. ACM Transactions on Information and System Security (TISSEC), 6(3), 365-403.
16. Bandirmali, N., & Erturk, I. (2012). WSNSec: A scalable data link layer security protocol for WSNs. Ad Hoc Networks, 10(1), 37-45.
17. Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, 13(7), 422-426.
18. Jinwala, D., Patel, D., & Dasgupta, K. (2012). FlexiSec: a configurable link layer security architecture for wireless sensor networks. arXiv preprint arXiv:1203.4697.
19. Li, S., Li, T., Wang, X., Zhou, J., & Chen, K. (2007). Efficient link layer security scheme for wireless sensor networks. Journal of Information And Computational Science, 4(2), 553-567.
20. Ren, F., Zhang, J., He, T., Lin, C., & Ren, S. K. D. (2011). EBRP: energy-balanced routing protocol for data gathering in wireless sensor networks. IEEE transactions on parallel and distributed systems, 22(12), 2108-2125.
21. Yessad, S., Bouallouche-Medjkoune, L., & Aïssani, D. (2015). A cross-layer routing protocol for balancing energy consumption in wireless sensor networks. Wireless Personal Communications, 81(3), 1303-1320.

22. Verma, V. K., Singh, S., & Pathak, N. P. (2014). Analysis of scalability for AODV routing protocol in wireless sensor networks. Optik, 125(2), 748-750.
23. Jian, D. (2012). Cloud model and ant colony optimization based QoS routing algorithm for wireless sensor networks. In Advanced Technology in Teaching-Proceedings of the 2009 3rd International Conference on Teaching and Computational Science (WTCS 2009) (pp. 179-187). Springer, Berlin, Heidelberg.
24. El-Fouly, F. H., Ramadan, R. A., Mahmoud, M. I., & Dessouky, M. I. (2016). Resource aware and reliable data reporting algorithm for object tracking in WSNs. Journal of Intelligent & Fuzzy Systems, 31(1), 99-113.
25. Ammari, Y. M. (2009). Challenges and opportunities of connected k-covered wireless sensor networks. Studies in Computational Intelligence. Springer, Berlin.
26. Shu, Y., Cheng, P., Gu, Y., Chen, J., & He, T. (2015). TOC: Localizing wireless rechargeable sensors with time of charge. ACM transactions on sensor networks (TOSN), 11(3), 1-22.