

RESEARCH ARTICLE

TeenySec: a new data link layer security protocol for WSNs

Murat Dener^{1*} and Omer Faruk Bay²¹ Graduate School of Natural and Applied Sciences, Gazi University, Besevler, Ankara, Turkey² Department of Electronics and Computer Education, Gazi University, Besevler, Ankara, Turkey

ABSTRACT

A Wireless Sensor Network (WSN) link layer security protocol called TeenySec is proposed in this paper. WSNs are caused by a lot of vulnerability because of factors such as hardware constraints of the sensor nodes, wireless communication medium, real-time computing, heterogeneous structure, large number of nodes, scalability, mobility, weight and cost requirements of application environment. In sensitive WSN applications like surveillance of enemy lines or border areas, security protocols must be used which provide confidential data transfer from sensors to base station. In this study, a new data link layer protocol is developed which is called TeenySec. The TeenySec combines the advantageous aspects of the Corrected Block Tea Algorithm (XXTEA) with the Offset Codebook Mode (OCB) approaches. TeenySec provides data confidentiality, data integrity, data freshness and data authentication and is also energy efficient. Moreover, proposed protocol has been implemented on the TelosB nodes. The security has been increased in the developed protocol. Also, better results have been obtained than TinySec, MiniSec and SNEP in energy consumption and memory usage which are very important for WSNs. Copyright © 2017 John Wiley & Sons, Ltd.

KEYWORDS

Wireless Sensor Network; security; data link layer; XXTEA; OCB

*Correspondence

Murat Dener, Graduate School of Natural and Applied Sciences, Gazi University, Besevler, Ankara, Turkey.

E-mail: muratdener@gazi.edu.tr

1. INTRODUCTION

In this paper, design and its implementation stages of a new highly secure and robust Wireless Sensor Network (WSN) link layer security protocol are presented. It is hereafter named as the TeenySec. The TeenySec mainly integrates the Corrected Block Tea Algorithm (XXTEA) [1] and the Offset Codebook Mode (OCB) [2] approaches in a new fashion. TeenySec provides data confidentiality, data integrity, data freshness and data authentication.

In current literature, despite the fact that most of the research on medium access aims at developing approaches to minimize energy consumption, WSN, especially military applications, should also consider data confidentiality, data integrity, data freshness and authentication aspects [3]. Security implementations that are used in a wide range of applications today and have been mainly designed for traditional networks cannot be used for WSNs as the sensor nodes have restricted energy supplies, insufficient storage capacities and restricted processing abilities [4]. Nowadays, various security protocols that consider these

aspects of WSNs and their nodes are being developed. While trying to fulfill these requirements, it should not be forgotten that WSN has restricted energy sources and the primary goal of a WSN is energy efficiency. Otherwise, a protocol that fulfills all the security requirements but consumes a bit of too much energy will be just impractical for WSN [3]. The security protocols to be developed should implement all the security issues (data confidentiality, data integrity, data freshness, data authentication) [5] but also provide high security with low energy consumption. Moreover, the fact that most of the suggested solutions are just based on the simulation platform and solutions on sensory platforms are not considered is a big deficiency in past research. Thus, in order to be able to use the suggested protocols in applications that require solid security, the protocols should also be tested on sensor nodes besides the simulation platform [3]. Within the solutions in literature TinySec [6], MiniSec [7], SPINS [8], Lsec [9], LLSP [10], LISA [11], LISP [12] and WSNSec [13], only TinySec, MiniSec and SNEP have software implementations on sensor nodes. Whereas, despite it has

been developed for Wireless Private Networks, IEEE 802.15.4 [14] has been used in WSN due to its low energy consumption, low cost and flexibility. Other security protocols have not been implemented on sensor nodes. In this study, a new data link security protocol has been developed that fulfill all the security requirements of WSN—namely data confidentiality, data integrity, data freshness and data authentication—and also is energy efficient. Besides, the proposed protocol has been implemented on TelosB node.

The aim of the study is developing a new data link layer protocol with high security and low energy consumption over sensor nodes. Rest of the paper is organized as follows. Security requirements in WSNs and available security solutions are mentioned in Section 2. Section 3 details the proposed TeenySec protocol design stages. Implementation of the TeenySec protocol is explained in the Section 4. Section 5 includes comparative performance analysis of the TeenySec and TinySec, MiniSec and SNEP. Finally, conclusions are given.

2. SECURITY ISSUES IN WSNs

In this section, the security requirements of WSN and security solutions present in current literature will be discussed.

2.1. Security requirements in WSNs

The so far listed security requirements of WSN are data confidentiality, data integrity, data freshness and data authentication [3,15–18]. These requirements are briefly explained herein after.

2.1.1. Data confidentiality

Data confidentiality in WSN prevents access of unauthorized people to collected data which is one of the most important requirements in sensitive WSN applications. A sensor node should not relay the data obtained from the environment to its neighbors. Especially in military applications, the data collected on the nodes can be very sensitive. Moreover, in many applications, nodes have to transmit very sensitive data (key distribution for example) to other sensor nodes through wireless transmission environment. In addition, routing data should also be kept secret against malignant nodes as these nodes can make use of these data and decrease the performance of the network. Because of these issues, it is very crucial to construct a safe communication channel for data transmission in WSNs. The standard approach for maintaining data confidentiality is the encryption of the data with a secret key. Due to their low energy consumption, encryption algorithms based on secret key substructure are used in WSNs.

2.1.2. Data integrity

Data confidentiality can prevent capturing of data by malicious nodes, yet it cannot stop data from being changed by unauthorized people. Data integrity assures

that the message will not be changed during communication. A malicious node can make the network work improperly by disrupting the message. What is more, without actual presence of a malicious node, the messages might get disrupted during transmission. Therefore, it is a must to use message authentication codes or cyclic codes for data integrity.

2.1.3. Data authentication

As WSNs use public wireless environment, they need authentication mechanisms to detect messages and deception packets coming from malicious nodes. Authentication mechanisms help a node with verifying the identity of a node it is in contact with. Without authentication, a malicious node can pretend to be a different node and might obtain some sensitive data and also prevent proper functioning of other nodes. If only two nodes are in contact, authentication can be accomplished by symmetric key cryptography. Transmitter and receiver can calculate the verification code of all the messages sent by a common hidden key.

2.1.4. Data freshness

In WSN structures, sensors send measurement data about the environment they are presently in with specific time intervals and the delivery of the measurement time's then matter. It is possible that copy of old measurement values are retransmitted by an attacker. Therefore, it is important to check that the data is new. Data freshness can be maintained by adding a counter to the message packet or by using a random number during encryption.

2.2. Available security solutions

Security-related solutions present in the literature are listed in Table I.

According to Table I, there are several studies about security data link layer protocols, but a few protocols—TinySec, MiniSec, IEEE 802.15.4 and SPINS (SNEP)—have software implementations on sensor nodes. However, there are some deficiencies in these protocols.

TinySec developed by the University of Berkeley is a link layer security architecture that has been included in the TinyOS version. In encryption of messages, Skipjack block encryption, 8-bit initialization vector (IV), and code block chaining (CBC) are used. MiniSec is implemented on Telos platform. SkipJack is used as the encryption algorithm and OCB as the encryption mode. IEEE 802.15.4 defines medium access and physical layers for wireless private area networks. Although this protocol was not developed for WSN, it is used in WSNs because of its low power consumption, low cost and flexibility. Currently, this protocol works on Micaz, TelosB nodes produced by the company CrossBow. ZigBee strong encryption AES-128 is used [3]. SPINS developed by Berkeley University consists of μ TESLA protocol used in identity authentication broadcasting, SNEP protocol providing confidentiality, identity authentication between two nodes and data

Table I. Available security protocols.

Security requirements for WSNs/Protocols	Data confidentiality	Data integrity	Data authentication	Data freshness	Application
TinySec	+	+	+	—	TinyOS
MiniSec	+	—	+	+	TinyOS
IEEE 802.15.4	+	+	+	+	TinyOS
SPINS	+	+	+	+	TinyOS
LSec	+	—	+	—	—
LLSP	+	+	+	+	—
LISA	+	+	+	+	—
LISP	+	+	+	—	—
WSNSec	+	+	+	+	—

freshness, and a routing protocol based on these. RC5 is used in encryption. The other protocols have no implemented on sensor nodes.

TinySec and MiniSec use Skipjack encryption algorithm that has a key of size 80 bit to provide data confidentiality, while IEEE 802.15.4 uses AES encryption algorithm and SNEP uses RC5 encryption algorithm with a key size of 128 bit. When DES—key size is 56 bit—is introduced in 1977, it was broken in 1982. According to this state, researchers have a study about the correlation between key size and year. They found that 80-bit key size cipher is safe till 2013 and 128-bit key size cipher is safe till 2076 [19].

Although this is provided in IEEE 802.15.4 and SNEP, AES needs high levels of memory utilization [20]. In addition, SNEP needs high levels of memory utilization. While TinySec cannot prevent message retransmission attacks, MiniSec cannot assure data integrity.

Therefore, TeenySec is developed which provides data confidentiality, data integrity, data authentication and data freshness with high security and low energy consumption, low memory usage and low latency.

3. THE PROPOSED TEENYSEC PROTOCOL

To provide Data confidentiality, Data Integrity, Data Authentication and Data Freshness in WSN, XXTEA + OCB is used. In this section, we explain why we use XXTEA algorithm and OCB mode. In addition, we present to TeenySec protocol design stages.

3.1. Why XXTEA

TEA has a key of size 128 bit, block of size 64 bit and 64 loops. Moreover, various cryptographic attacks have come to use against TEA. In order to overcome such limitations of TEA, XTEA has been proposed by Needham. Yet, it has been observed that XTEA has some deficiencies such as slow diffusion rate and low efficient rate [21]. Then, XXTEA has been suggested by Wheeler which filled these deficiencies and got stronger. XXTEA has a key size of 128 bit. Of course, XXTEA has been suggested not only because of its 128-bit key size but also because it is simpler

and minimum. Soren Rinne [22] has compared TEA, XTEA, SEA, AES, HIGHT and DES algorithms. Writers have stated that if memory is an important criterion, TEA or XTEA would be a good choice. Liu [23] has argued about the application of TEA algorithm to Berkeley node platforms. Writers have shown that TEA's performance is fast time-wise. Großshädl Johann [24] has evaluated block encryption algorithms energy-wise. Writers have focused on performance, energy consumption, duration of running and memory requirements in their article. They have compared RC6, AES, Serpent, Twofish and XTEA algorithms. XTEA has given the best results. In Lee's study [20], comparisons have been made on processor cycle during encryption, decryption and key ordering processes. If only encryption and decryption are taken into consideration, Skipjack algorithm gives better results than TEA family. Yet, Skipjack requires a high amount of processor cycling for key identification. In Rehman's study [25], Skipjack and TEA family have been compared based on energy consumption. Compared to Skipjack, XXTEA spends 6.27% and XXTEAQ spends 5.03% more energy. Energy is a critical resource in WSN nodes. But, XXTEA's key size is 128 bit, while Skipjack's is 80 bit. According to analysis based on all the accumulated experimental data, we think that XXTEA is the most suitable encryption method for WSN. Although Skipjack encryption method is faster and more energy efficient for encryption and decryption, it has consumed a lot of energy during key distribution. Therefore, overall energy requirements are higher in Skipjack. Therefore, in the proposed protocol, XXTEA has been used as the encryption algorithm.

3.2. XXTEA (Corrected Block Tea)

The block has been designed by fixing the weak aspects of TEA (XTEA) algorithm. Roger Needham and David Wheeler from University of Cambridge have developed this algorithm. The paper that presents the algorithm has been published in 1998. XXTEA [1,23,26] algorithm has a key size of 128-bit, block length of 64 bit and loop number of 32. XXTEA has eight stages. The stages from the fourth to the eighth are repeated 32 times. XXTEA is an algorithm that uses left shifting, right shifting, addition and XOR operations. Operations conducted in one cycle of XXTEA have been shown in Figure 1.

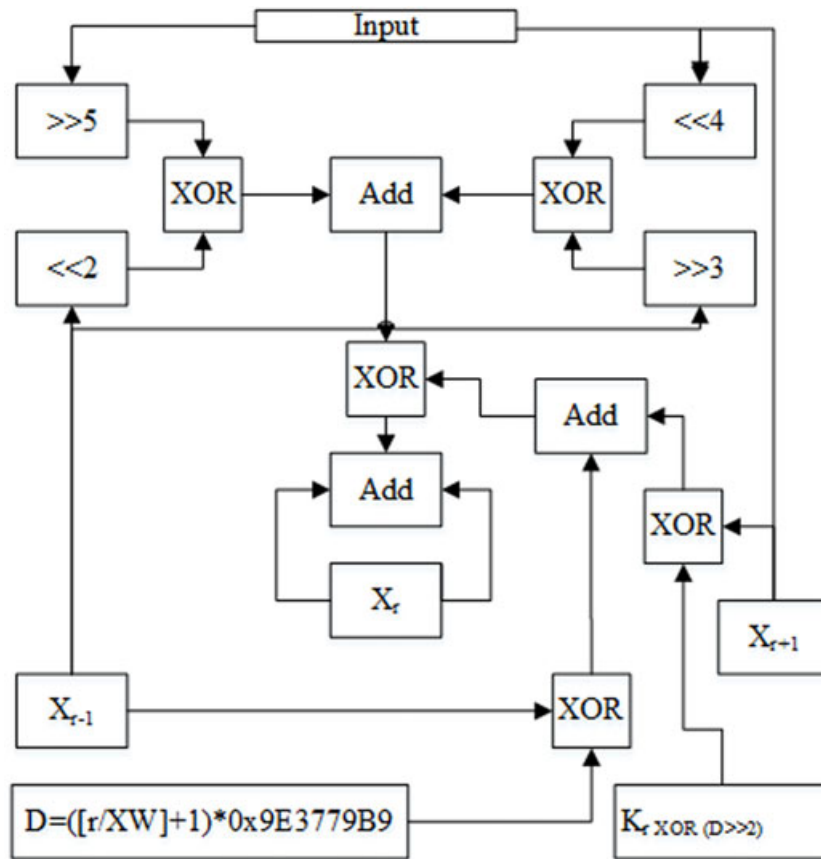


Figure 1. XXTEA.

First, the startup parameters are regulated in XXTEA. Then, the 128-bit key to be used in encryption is determined, and the message to be encrypted is separated into blocks of 64 bit. 0's is added to the last message until it reaches the size of 64 bit. Then, the following procedures are executed 32 times:

- Sum value is added to the Delta value (sum += DELTA) (DELTA is a specific value for XXTEA.)
- Shifting and XOR operations with sum value ($e = (\text{sum} > 2) \& 3$)
- For all the message blocks, the following operation is executed. (z is the first block of the message, y is the second block of the message, k is the encryption key)

$$(z > 5^y < 2) + (y > 3^z < 4)^{(\text{sum}^y)} + (k[p \& 3^e]^z)$$

This way, all the message blocks are encrypted.

3.2.1. Offset Codebook Mode

This is the encryption mode in which authentication and privacy principles are fulfilled together. Other modes (ECB, CBC, CTR...) use two different systems that consist of encryption mode for data confidentiality and MAC for authentication. OCB [2,27–29] has united these two

systems. It is costly compared to the CBC mode that only assures privacy. Still, OCB mode can easily be integrated into the software. Therefore, in the proposed protocol, OCB has been used as the operation mode. The schema of OCB depicted in Figure 2.

Δ is initial random value which is generated by Nonce (N) 96-bit random value, M is the message blocks and E_K is the encryption algorithm. Encryption then works as shown in the Figure 2. The value Checksum is the 128-bit string $\text{Checksum} = M_1 \oplus \dots \oplus M_m$. The number τ , the tag length of the scheme, is, like the blockcipher E, a parameter of the mode. It's a number $0 \leq \tau \leq 128$. The ciphertext is the $128m + \tau$ bit string $CT = C_1 C_2 \dots C_m T$.

3.2.2. TeenySec protocol

The TeenySec has been developed based on a new integrated approach. It mainly utilizes the idea that the XXTEA encryption algorithm is restructured using OCB. The schema of TeenySec Protocol is depicted in Figure 3. The TeenySec Protocol consists of eight steps. These are listed below:

- First Step Entered message is divided into blocks of 64 bit ($M[1], M[2], \dots, M[m]$)
- Second Step If the last message block is not 64 bit, then the remaining bits are filled with 0's.

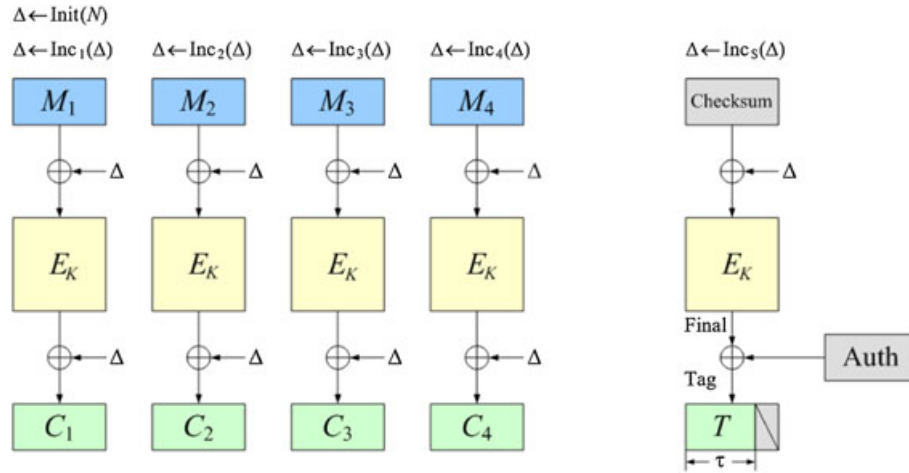


Figure 2. OCB.

Third Step One of the important properties of TeenySec Protocol is the L value. This value is different for different message blocks. L value increases the complexity in encryption method.

- In the beginning, the value that contains 8 bytes ($8 \times 8 = 64$ bit) is taken and encrypted with the help of the encryption algorithm. The encrypted value equals L [0].
- L value is generated till it matches the number of message blocks.
- Remaining values (L [1], L [2] L [n]) are produced with the following logic:

If MSB bit of L[i]=0 then $L[i+1]=L[i] < < 1$
 If MSB bit of L[i]=1 then $L[i+1]=L[i] > > 1$

Fourth Step NONCE value is a random value. The generated NONCE value is encrypted by an encryption algorithm, to produce the R value. This R value is then used in the production of the offset value.

Fifth Step This way, offset is produced by L values. R is XOR'ed with each L value. So, the offset of the message block is L[i] xor R. This offset value is XOR'ed with the message both before and after encryption.

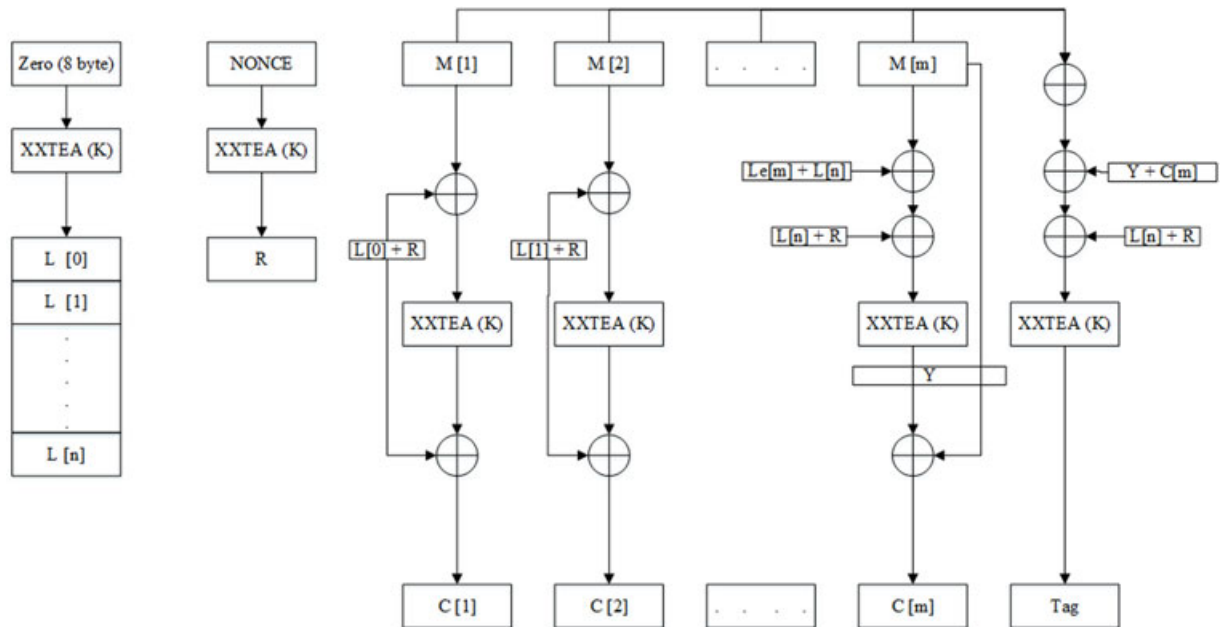


Figure 3. The TeenySec architecture.

- Sixth Step Two operations are fulfilled ($\text{Length}[m] + L[n]$, $L[n] + R$) before encryption and Y value is obtained after encryption in the last message. Then, Y Value xor M [m] is generated to C [m] value.
- Seventh Step With the help of Hash function, tag values are generated. Hash function belongs to the original message. Hash function is the XOR'ed version of each message block. They are XOR'ed Y + C[m] and L [n] + R values. Then, they encrypted with the encryption algorithm. The generated tag value is then used for data authentication.

The last step consists of encryption of all the values and production of the encrypted script with the tag value.

Encryption algorithms are asked to produce two encrypted scripts when they encrypt the same script twice. Changing a vector used in encryption (IV) every time the algorithm is run fulfills this requirement. If IV gets longer, packet size and load increases. If IV gets shorter, then IV's repetition frequency increases. As TinySec and MiniSec use IV that is produced by a counter, they bring extra load to the packet. To prevent this, IV produced with NONCE has been chosen as the startup vector in TeenySec protocol. This way, data freshness is also secured. There is no need for a different algorithm in TeenySec (for example CTR).

While data confidentiality is provided with XXTEA, data integrity and data authentication will be secured with OCB. As a random number (NONCE) is used every time a message is sent, data freshness is also secured. In other protocols, counter is used to provide data freshness, and this brings extra load to the message packet. This problem does not exist anymore in TeenySec.

4. IMPLEMENTATION

We have developed TeenySec for the TelosB motes. It features the 8-MHz TIMSP430 micro-controllers, a 16-bit RISC processor that is well known for its low energy consumption. Although we implemented TeenySec on the TelosB motes, our design principles are general enough such that porting TeenySec to different sensor platforms should yield similar performance results. At first, restructured XXTEA + OCB have been coded by C programming language. Then, after the working principle of

these is comprehended, the written code is transferred into NesC. After this procedure, before loading the written NesC code onto the node, in order to test its accuracy, simulation results have been obtained from 30 nodes by the help of TOSSIM simulation program. The codes for XXTEA and OCB were written in different files as modules. Here, the incoming message packet for AMStandard has been sent to decryption by the help of OCBMode interface. So, it is sufficient to exchange these files with researchers's own algorithms or modes. The code written this way will be both modular and easily usable. In order to apply TeenySec, changes have been made to GenericComm and AMStandard files from TinyOS library. XxteaM and OcbM components' connection with OcbMode interface has been added to the GenericComm configuration file. On the other hand, in AMStandard file, a connection has been made with OcbMode interface for the encryption of sent messages and decryption of received messages. To use TeenySec, a developer simply needs to replace GenericComm, AMstandard with GenericCommTeenySec, AMstandardTeenySec in the application's configuration file. We implemented TeenySecProtocol in 2600 lines of NesC code the programming language used for TinyOS.

5. EVALUATION

This section is devoted to the comparison of TeenySec protocol to other protocols with applications (TinySec, MiniSec, SNEP). Results from TinySec, MiniSec and SNEP articles have been used for comparison. The comparisons have been made based on Security, Packet Format, Energy, Latency, Memory Usage principles.

5.1. Security

Security comparisons regarding the protocols which have been implemented on sensor nodes are given in Table II.

It is apparent that TeenySec protocol is safer than TinySec and Minisec due to the fact that it fulfills data integrity and data freshness criteria, and it has a key length of 128 bit. When compared with SNEP, Teenysec's security is approximately same. However, with the conception of CBC + CTR in SNEP gets additional load to system. As a result, this leads to additional energy, memory usage, latency and complexity.

Table II. Security comparisons regarding the protocols which have implemented on sensor nodes.

	TinySec	MiniSec	SNEP	TeenySec
Key length	80 bit	80 bit	128 bit	128 bit
Message retransmission attacks	—	+	+	+
Data integrity	+	—	+	+
	Skipjack + CBC	Skipjack + OCB	RC5 + CBC + CTR	XXTEA + OCB

1	2	1	2	2	1	1	0..28	2
Len	FCF	DSN	DstPan	DstAddr	AM	Grp	Enc Data	CRC

TinyOS (a)

1	2	1	2	2	1	2	2	0..28	4
Len	FCF	DSN	DstPan	DstAddr	AM	SrcAddr	Ctr	Enc Data	Tag / MIC

TinySec (b)

1	2	1	2	2	1	2	0..28	4
Len	FCF	DSN	DstPan	DstAddr	AM	SrcAddr	Enc Data	Tag / MIC

MiniSec (c)

1	2	1	2	2	1	2	1	0..28	8
Len	FCF	DSN	DstPan	DstAddr	AM	SrcAddr	Ctr	Enc Data	Tag / MIC

SNEP (d)

1	2	1	2	2	1	2	0..28	4
Len	FCF	DSN	DstPan	DstAddr	AM	SrcAddr	Enc Data	Tag / MIC

TeenySec (e)

Figure 4. Packet format.

5.1.1. Packet format

We based TeenySec's packet format on the current packet format in TinyOS. We show the differences between TeenySec packets and TinyOS packets. To detect transmission errors, TinyOS senders compute a 2-byte cycle redundancy check over the packet. To guarantee message integrity and authenticity, TeenySec replaces the cycle redundancy check with a 4-byte MAC. The TinyOS packet format contains a group field to prevent different sensor networks from interfering with each other. It can think of as a kind of weak access control mechanism for non-malicious environments. Because TeenySec enforces access control with a MAC, the group byte is unnecessary in TeenySec. While TinySec and SNEP use CTR, Minisec and TeenySec don't use it. Also, SNEP has 8-byte Tag/MIC for authentication. Finally, similar to TinySec, MiniSec and SNEP, TeenySec requires a 2-byte source address, which is absent in a standard TinyOS packet. The net overhead of a TeenySec packet is 3-byte increase over a standard TinyOS packet. Figure 4 shows the packet format for plain TinyOS, TinySec, MiniSec, SNEP and TeenySec.

Comparison of the packet sizes of TinySec, MiniSec and TeenySec protocols is given Table III.

In the energy, latency and memory usage section, the scenario is as follows: There are two TelosB motes in the media. One is end device, and the other is base station. End device node generates packet and sends it to base station with 10-ms frequencies.

5.1.2. Energy

TinySec, MiniSec and SNEP articles have been used for comparison. We take the results of values that increase over TinyOS. Digital multimeter has been used for energy measurements. The conclusion has been made after measurements on nodes with TinyOS code and TeenySec protocol. Excel Alkaline battery is loaded on the nodes. In order to make comparisons with TinySec, the type of measurement TinySec uses has been taken as basis. The procedure is as follows: current measurement procedure is executed on the current node. First value (firstvalue) is

Table III. Comparing packet size.

	Total packet size (byte)	Security overhead (byte)
TinyOS	40	—
TinySec	45	5
MiniSec	43	3
SNEP	48	8
TeenySec	43	3

recorded. The node continues working this way for about an hour. After 1 h, the value on the screen (lastvalue) is recorded. The result from lastvalue – firstvalue operation gives the total amount of energy consumption in 1 h. This scheme is fulfilled 30 times, and average results are obtained. As already stated in the literature [6], making measurements based on mAH value increases the accuracy of the measurement when two batteries have equal amount of voltage. mAH denotes the total amount of energy consumption in one hour. For the security and accuracy of the measurement, TinySec's way of energy measurement has been used as the reference. According to the results of the comparison as is seen in Table IV, while amount of energy consumption in TinyOS is 0.000160 mAH, it is 0.000172 in TeenySec protocol. TeenySec's energy costs come from two sources: increased packet lengths and extra computation from the cryptography. Energy consumption is less compared to TinySec because additions to the packet due to security reasons are 5 bytes for TinySec whereas it is 3 bytes for TeenySec.

Table IV. Total energy consumed to send a 24-byte packet.

	Energy (mAH)	Increase
TinyOS	0.000160	—
TinySEC	0.000176	10.0%
MiniSec	0.000173	8.3%
SNEP	0.000195	22.2%
TeenySec	0.000172	8.0%

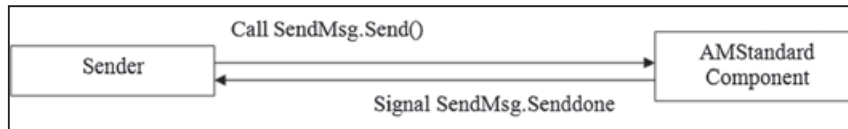


Figure 5. SendMsg.Send function and SendMsg.Senddone event.

5.1.3. Latency

MiniSec and SNEP have no information about latency analysis in their articles. So, latency analysis is conducted only TinySec. However, when thinking of packet overhead, while latency in SNEP is possibly higher than TeenySEC, latency in MiniSec is possibly equal with TeenySEC.

TinySec states in its article that it added a 7.9% of delay to TinyOS. To access the delay results in TeenySec protocol, the following operations have been conducted. To make it more clear, the connection between the node that sends the message and AMStandard component is depicted in Figure 5. The node with TinyOS operating system loaded on calls the SendMsg.Send function after it prepares the message packet. After the transmission is completed, the SendMsg.Senddone signal is called.

Counters have been added before the call to SendMsg.Send function and SendMsg.Senddone event. Before SendMsg.Send function is called sendcounter has been added, while senddonecounter is added after the call to SendMsg.Senddone event. In order to find out node's loss in time during transmission moment, the value obtained by adding up the senddonecounter – sendcounter values over 300 packet transmissions is taken as basis. The reason behind calculating the result after 300 packets are transmitted is to ensure an average, near accurate outcome. Then, the obtained value is divided to 300, and value of 7.53 ms is obtained. The same procedure has also been repeated during TeenySec protocol. The outcome is 8.38 ms. So, the increase in latency compared to TinyOS is 11.28%. In Table V, the increasing amount of latencies—in comparison to TinyOS—in TeenySec and TinySec protocols is depicted.

When the results are normalized, TinySec becomes 1 while TeenySec is 1.42. In other words, the latency does not differ significantly from TinySec. The utilization of 128 bit key rather than 80 bit affected the latency results.

5.1.4. Memory usage

TinySec, MiniSec and SNEP articles have been used for comparison. We take the results of values that increase over their own system. There are two types of storages in sensor nodes. These are RAM and ROM. RAM is a temporary storage area where the data to be processed by the

processor is held while ROM is the storage area for keeping the permanent data required for an application or for an operating system to function. The comparative memory usage by TeenySec and TinySec, MiniSec and SNEP protocols is depicted in Table VI.

TinySEC has been implemented on Mica2 node. The RAM capacity of Mica2 node is $4K = 4 \times 1024 = 4096$ bytes, while ROM capacity of it is $128K = 128 \times 1024 = 131\,072$ bytes. TinySEC uses 256 bytes of RAM and 8152 bytes of ROM. So, TinySEC uses 6.25% of RAM and 6.21% of ROM.

MiniSec has been implemented on TelosB node. The RAM capacity of TelosB node is $10K = 10 \times 1024 = 10\,240$ bytes, while ROM capacity of it is $48K = 48 \times 1024 = 49\,152$ bytes. MiniSec uses 874 bytes of RAM and $16K = 16 \times 1024 = 16\,384$ bytes of code memory. So, MiniSec uses 8.53% of RAM and 33.33% of ROM.

SNEP has been implemented on their own sensor nodes whose RAM capacity is 512 bytes and ROM capacity is $8K = 8 \times 1024 = 8192$ bytes. SNEP uses 220 bytes of

Table VI. The comparative memory usage.

	Increase	
	ROM	RAM
TinySec	6.21%	6.25%
MiniSec	33.33%	8.53%
SNEP	25.00%	42.96%
TeenySec	4.43%	4.81%

Table VII. General evaluation.

	TinySec	MiniSEC	SNEP	TeenySec
Key size	80 bit	80 bit	128 bit	128 bit
Data confidentiality	+	+	+	+
Data integrity	+	—	+	+
Data authentication	+	+	+	+
Data freshness	—	+	+	+
Security overhead (byte)	5	3	8	3
Increase (energy)	10%	8.3%	22.2%	8.0%
Increase (latency)	7.9% (1)	*	*	11.28% (1.42)
Increase (memory usage—RAM)	6.25%	8.53%	42.96%	4.81%
Increase (memory usage—ROM)	6.21%	33.33%	25.00%	4.43%

*MiniSec and SNEP have no latency information in their own articles.

Table V. The increasing amount of latencies.

	Increase	Normalize
TinySEC	7.9%	1.00
TeenySec	11.28%	1.42

RAM and SNEP consumes about $2K = 2 \times 1024 = 2048$ bytes of program memory. So, SNEP uses 42.96% of RAM and 25% of ROM.

TeenySec has been implemented on TelosB node. The RAM capacity of TelosB node is $10K = 10 \times 1024 = 10\,240$ bytes while ROM capacity of it is $48K = 48 \times 1024 = 49\,152$ bytes. TeenySec uses 728 bytes of RAM and 7146 bytes of code memory. So, TeenySec uses 4.81% of RAM and 4.43% of ROM.

According to Table VI values, TeenySec protocol is better than TinySec, MiniSec and SNEP.

5.1.5. General evaluation

As it is seen in Table VII, in TeenySec protocol, security has been increased and criteria that matter to WSN such as energy and memory usage turned out to be better than TinySec, MiniSec and SNEP. The latency increase was negligible when compared to TinySec. Despite the fact that TeenySec uses a key size of 80 bit instead of 128 bit, the 2 bytes of packet increment in TinySec leads to such result. In addition, the fact that OCB mode is faster and less memory demanding than CBC mode used in TinySec can be also considered as one of the reasons.

6. CONCLUSIONS

During the production of security protocols, it is necessary to consider each one of WSN's security requirements (Data confidentiality, Data Integrity, Data Freshness, Data Authentication) without forgetting about high security and low energy consumption idea while fulfilling these requirements. Moreover, it is a big deficiency in past research that most of the suggested security solutions are just tested on the simulation environment and not evaluated on sensor platforms. Hence, in order to be able to use suggested solutions in solid security-requiring applications, it is necessary that the applications are not restricted to the simulation environment and also tested on sensor nodes. Within solutions present in literature, only TinySec, MiniSec and SNEP have been implemented on sensor nodes. Whereas, despite being developed for Private Area Wireless Networks, IEEE 802.15.4 is used in WSN due to its low energy consumption, low cost and high flexibility. Other security protocols have not been implemented on nodes. TinySec and MiniSec have used Skipjack algorithm of 80-bit key size, to secure data confidentiality. While TinySec fails in preventing message retransmission attacks, MiniSec cannot ensure data integrity. Therefore, a new data link layer security protocol has been produced that fulfill all the security requirements of WSNs, which are data confidentiality, data integrity, data freshness and data authentication principles and also is energy efficient. At the same time, the proposed protocol has been implemented on TelosB sensor nodes. Before the design of the protocol has been made, available encryption algorithms and encryption modes have been analyzed, and as WSNs have restricted hardware resources, it is concluded that

the most suitable algorithm is XXTEA and the most suitable encryption mode is OCB. Data confidentiality, data integrity, data freshness and data authentication requirements have been fulfilled with the combination of restructured XXTEA + OCB. In the proposed protocol, both the security are increased, and better results have been obtained compared to TinySec, MiniSec and SNEP regarding issues like energy and memory use which are important for WSN. In latency criteria, compared to TinySec, it has exhibited a negligible increase. The proposed protocol has a modular structure at the same time. While developing a security approach, it is necessary to consider wireless sensor node sources' (memory, processor, power source) capacity. In WSN applications, the increase of node's energy consumption and average latency period from one end to the other as a result of encryption mechanisms that increase security is an expected outcome. At this point, the accurate identification of the needs of applications is quite crucial. While security does not matter so much in a simple wide scale environmental or industrial WSN application, energy consumption is of capital importance. On the other side, security has high importance in health and military applications whereas node energy consumption is relatively negligible. Therefore, it is quite crucial to choose the appropriate encryption algorithm and encryption mode for security solutions developed for military and health applications. The security solutions to be developed should be modular. It means that if novel encryption algorithms and encryption modes in the literature provide better security, energy, memory use and latency, it should be possible to directly integrate these solutions to the proposed protocol.

REFERENCES

1. Wheeler D, Needham R M. XXTEA: corrections to XTEA. Technical report, Computer Laboratory, University of Cambridge, (1998).
2. Ucdavis Computer Science, "OCB mode", <http://www.cs.ucdavis.edu/~rogaway/ocb/> (2016).
3. Dener M. Security analysis in wireless sensor networks. *International Journal of Distributed Sensor Networks* 2014; **2014** Article ID 303501.
4. Chelli K. Security issues in wireless sensor networks: attacks and countermeasures. *Proceedings of the World Congress on Engineering* 2015; **1**.
5. Chen Z, He M, Liang W, Chen K. Trust-aware and low energy consumption security topology protocol of wireless sensor network. *Journal of Sensors* 2015; **2015** Article ID 716468.
6. Karlof C, Sastry, N Wagner D, TinySEC: a link layer security architecture for wireless sensor networks, 2nd ACM Conference on Embedded Networked Sensor Systems SENSYS 2004, Maryland, USA, 162-175, (2004).

7. Luk M, Mezzour, G Perrig, A Gligor V. "MiniSec: a secure sensor network communication architecture", ACM International Conference on Information Processing in Sensor Networks, (2007).
8. Perrig A, Szewczyk R, Tygar JD, Wen V, Culler DE. Spins: security protocols for sensor networks. *Wireless Networking* 2002; **8**(5):521–534.
9. Shaikh RA, Lee S, Khan MA, Song YC. LSec: light-weight security protocol for distributed wireless sensor network. *PWC* 2006:367–377.
10. Leonard E, Ren LJ, Li T. An Energy Efficient Link-Layer Security Protocol for Wireless Sensor Networks. *IEEE EIT* 2007; **7**:233–238.
11. Tripathy S. LISA: lightweight security algorithm for wireless sensor networks, Proceedings of the 4th international conference on Distributed computing and internet technology, (2007).
12. Park T, Shin KG. LiSP: a lightweight security protocol for wireless sensor networks. *ACM Transactions on Embedded Computing Systems* 2004; **3**(3):634–660.
13. Bandirmali N, Erturk I. WSNSec: a scalable data link layer security protocol for WSNs. *Ad Hoc Networks* 2012; **10**:37–45.
14. IEEE-TG15.4, "Part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)", IEEE standart for Information Technology, (2003).
15. Fatema N, Brad R. Security requirements, counterattacks and projects in healthcare applications using WSNs—a review. *International Journal of Computer Networking and Communication* 2014; **2**(2):1–9.
16. Mathur A, Newe T. Comparison and overview of wireless sensor network systems for medical applications. *Proceedings of the 8th International Conference on Sensing Technology* 2014:272–277.
17. Kaur L, Malhotra J. Review on security issues and attacks in wireless sensor networks. *International Journal of Future Generation Communication and Networking* 2015; **8**(4):81–88.
18. Pule M, Jamisola R, Ibikunle F. Energy requirements in cryptographic mechanisms for secure wireless sensor networks: an overview. *International Conference on Clean Energy for Sustainable Growth In Developing Countries* 2015:1–6.
19. Jinwala DC, Patel DR, Dasgupta KS. Investigating and analyzing the light-weight ciphers for wireless sensor networks. *Infocomp Journal of Computer Science* 2009; **8**(2).
20. Lee J, Kapitanova K, Son SH. The price of security in wireless sensor networks. *Computer Networks* 2010; **54**:2967–2978.
21. Wikipedia, "XXTEA", <https://en.wikipedia.org/wiki/XXTEA> (2016).
22. Rinne S, Eisenbarth T, Paar C. Performance analysis of contemporary light-weight block ciphers on 8-bit microcontrollers, ECRYPT Workshop SPEED—Software Performance Enhancement for Encryption and Decryption, Amsterdam, (2007).
23. Liu S, Gavrylyako OV, Bradford PG. Implementing the TEA algorithm on sensors, ACM-SE 42: Proceedings of the 42nd annual Southeast regional conference, pages 64–69, New York, NY, USA, (2004).
24. Großschadl J, Tillich S, Rechberger C, Hofmann M, Medwed, M. Energy evaluation of software implementations of block ciphers under memory constraints, DATE '07: Proceedings of the conference on Design automation and test in Europe, pages 1110–1115, San Jose, CA, USA, (2007).
25. Rehman S, Bilal M, Ahmad B, Yahya KM, Ullah A, Rehman OU. Comparison based analysis of different cryptographic and encryption techniques using message authentication code (MAC) in wireless sensor networks (WSN). *IJCSI International Journal of Computer Science Issues* 2012; **9**(1):2.
26. Schneier B. *Applied Cryptography* (Second edn). John Wiley & Sons, Inc.: New York, Ny, 1996.
27. Devesh CJ, Dhiren R, PatelKankar S. Dasgupta, Optimizing the block cipher resource overhead at the link layer security framework in the wireless sensor networks, Proceedings of the World Congress on Engineering 2008 Vol I WCE 2008, July 2–4, (2008).
28. Wikipedia, "OCB Mode", http://en.wikipedia.org/wiki/OCB_mode (2016).
29. Rogaway P, Bellare M Black J, Krovetz T. "OCB: a block-cipher mode of operation for efficient authenticated encryption".