Energy-efficient Optimizations of the Authentication and Anti-replay Security Protocol for Wireless Sensor Networks

Laura Gheorghe, Răzvan Rughiniş, Nicolae Ţăpuş
University POLITEHNICA of Bucharest
Bucharest, Romania
{laura.gheorghe, razvan.rughinis, ntapus}@cs.pub.ro

Abstract **- Wireless Sensor Networks are an emerging technology used for environmental monitoring. Security is a major concern when deploying a network for critical applications, such as military or medical surveillance. We have previously developed a security protocol that provides authentication, anti-replay, integrity and reliability. This paper presents further optimizations in order to minimize energy consumption. We have implemented the Energy-efficient Authentication and Anti-replay Security Protocol in TinyOS and we have tested its functionality and performance using the TOSSIM simulator. We have developed a mathematical model to evaluate energy consumption, determining the control overhead of the security protocol and the energy saved through the optimizations and thus proving the efficiency of the optimizations. The optimized EAASP represents a security protocol that provides strong authentication and anti-replay, integrity and reliability, and energy-efficiency.**

Keywords - wireless sensor networks, security, authentication, anti-replay, integrity, reliability, energy-efficiency.

I. INTRODUCTION

Wireless Sensor Networks (WSNs) consist of a large number of small embedded devices with limited capabilities and low power consumption that have the abilities to self-organize into a network and to perform sensing, communicating and processing tasks [1].

WSNs are used to monitor their environment. Standard applications that use WSNs are environmental, medical, and military surveillance and emergency detection [2]. Such applications require high levels of security.

Securing sensor networks is a challenging task because of their specific constrains, such as the limited capabilities of sensor node hardware, and the deployment context [3].

We have previously developed the Authentication and Anti-replay Security Protocol (AASP), which provides authentication, integrity, anti-replay and reliability. Our contributions in this paper consist in several optimizations to the AASP in order to reduce the energy consumption, such as minimizing the control overhead, reducing the number of handshake packets, using negative acknowledgements instead of positive ones, and aggregating sensed data.

We have also developed a mathematical model that evaluates the energy consumption introduced by the security protocol, and we used it in order to perform a formal evaluation of the control overhead and energy savings.

The rest of the paper is structured as follows: Section II presents related work, Section III discusses the protocol design, Section IV describes the implementation of the protocol, Section V presents the mathematical model and the evaluation results, and Section VI discusses the values of the protocol and concludes the paper.

II. RELATED WORK

Several significant security solutions for WSNs include the ZigBee Security Architecture, SPINS, and TinySec.

ZigBee Security Architecture consists in a coordinator that acts as "Trust Center", which allows other devices to join the network and provides them keys [4]. ZigBee works with three roles: the trust manager that authenticates devices that want to join the network, the configuration manager that manages and distributes keys, and the configuration manager that provides end-to-end security. The infrastructure operates in two modes: the Residential Mode that is used for low security residential applications and the Commercial Mode that is used by high-security commercial applications.

SPINS is a suite of security protocols optimized for WSNs, consisting of two building blocks: the Secure Network Encryption Protocol (SNEP) and the "micro" version of the Timed, Efficient, Streaming, Loss-tolerant Authentication Protocol (μTESLA) [5]. SNEP provides confidentiality using encryption, authentication and integrity by Message Authentication Codes (MAC). μTESLA provides authenticated broadcast by emulating asymmetry through a delayed disclosure of symmetric keys. SPINS has been implemented on top of TinyOS [6].

TinySec has been designed as the replacement of SNEP and provides confidentiality, authentication, integrity and anti-replay protection [7]. It implements the Cipher Block Chaining (CBC) mode with cipher text stealing for encryption and the Cipher Block Chaining Message Authentication Code (CBC-MAC) for authentication. TinySec uses the TinySec-Auth format for authenticated packets and the TinySec-AE for authenticated and encrypted packets.

We aim to develop a security protocol that provides strong authentication through the establishment of an authentication connection, strong anti-replay through binding the packet to its context, integrity and reliability, while also being energy-efficient.

III. AN ENERGY-EFFICIENT SECURITY PROTOCOL

The purpose of this work is to develop a lightweight, energy-efficient security protocol that provides authentication, freshness and integrity for Wireless Sensor Networks. EAASP has been designed by minimizing the

 ISBN: 978-1-61208-133-5

energy consumption of the Authentication and Anti-replay Security Protocol (AASP) [8] [9].

In order to improve the energy efficiency of a protocol one has to reduce the number of packets communicated by nodes, and the average packet size. We aim to optimize the security protocol by reducing the control overhead.

A. Authentication and Anti-replay Security Protocol

We have previously developed a security protocol that provides two-way authentication, anti-replay protection and integrity [8].

Authentication is ensured by the use of the Message Authentication Code (MAC), which is computed from the payload of the message and a secret key, by applying a collision resistant hash function. We have used the Hash Message Authentication Code (HMAC) implementation.

The anti-replay protection derives from binding the packet to its context, specifically to the previous packet between the same source and destination, and its sequence number. The mechanism consists in including the MAC of the previous packet in the current packet for all messages sent between the same source and destination.

Integrity is ensured by including the MAC of the current message in the packet.

For the first packet, authentication is performed by checking the MAC of the current message. Still, this measure does not protect against replay attacks. If the exact sequence of packets between the same source and destination is captured, it can be easily replayed later.

In order to further strengthen the authentication and anti-replay protection, an authenticated connection has to be established before sending any data packets. The authenticated connection is established in AASP after a four-step handshake.

In order to prevent the de-synchronization of the anti-replay mechanism through loss of packets, we have implemented an acknowledgement mechanism [9]. The next packet is not sent until the previous packet is acknowledged by the destination. The source node waits for the acknowledgement for a specified period of time and, if it times out, the packet is re-sent.

The authenticated connection has to be re-initiated if one of the communicating nodes loses its connection data or if the anti-replay mechanism is de-synchronized. The connection times out after a specified period of time in which no data or ACK message is received from the other node. In that moment, connection data is erased and an authenticated connection can be re-initiated.

The AASP is effective against malicious injection and replay attacks. In order to increase its energy efficiency we aim to reduce the number of control packets and the control overhead from the data packets.

B. Energy-efficient Authentication and Anti-replay Security Protocol

In this paper, we present a lightweight security protocol that uses the basic mechanisms of AASP and has higher energy efficiency. In order to reduce energy consumption, we reduce the number of packets and the control overhead.

1) Reducing the control overhead

The AASP protocol has a header with the following fields: Previous Hash (P_Hash) – 2 bytes, Current Hash (C_Hash) – 2 bytes, Authentication (Auth) – 1 byte, Acknowledge (ACK) – 1 byte and Sequence (Seq) – 1 byte.

EAASP is designed with a protocol header as presented in Table 1.

TABLE I. HEADER STRUCTURE

	EAASP Header Fields		
	Hash	***Type***	***Seq***
Number of bytes	2	1	1

The Hash field contains a MAC computed from the current payload, the previous payload, the secret key and the sequence number, as shown in Formula (1). The Type field encodes the type of packet, as presented in section C.

$$Hash_i = MAC(Payload_i, Payload_{i-1}, Seq, K) \quad (1)$$

The sequence number is taken into account when computing the hash in order to avoid packet altering by intermediate nodes.

2) Reducing the number of control packets

AASP has two types of control packets: handshake packets and acknowledgement packets.

We aim at reducing the number of handshake packets, while still providing powerful authentication. The authentication method can be strengthened by sending random challenges to each other.

We have reduced the number of handshake packets to three, as presented in Figure 1.

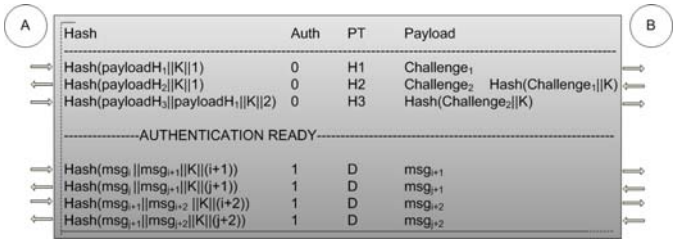

Figure 1. Energy-efficient Three Step Handshake

The first packet, designated H1, contains a challenge number randomly generated by the initiating node A. The second packet, designated H2, is the answer from node B, and it contains a hash based on the first challenge and the secret key, and the challenge randomly generated by node B. The third packet, designated H3, is the response of node A and contains a hash based on the second challenge and the secret key. After the three-step handshake is completed, data packets can be exchanged.

We have used negative acknowledgements to reduce the number of acknowledgement packets while still maintaining a certain level of reliability.

The destination is able to detect packet loss when receiving out-of-sequence packets. The destination stores the sequence number of the last received and valid packet, and it

 ISBN: 978-1-61208-133-5

expects the packet with the next sequence number. When it receives a packet with a sequence number greater than the expected one, it generates and then sends a Negative Acknowledgement (NACK) packet back to the source node.

The out-of-order packet is not dropped at the destination node, and it is stored until all previous packets are received. The NACK packet contains the sequence number of the first lost packet – L_Seq, and the sequence number of the received out-of-order packet – R_Seq, in order to prevent the source node to deliver the out-of-order packet again. The source node resends all packets with a sequence number greater or equal to L_Seq and less than R_Seq, when receiving a NACK packet. The destination performs an integrity and an anti-replay check on the re-sent and out-of-order packets, and it delivers them to the application.

C. Protocol message structure

The EAASP header contains the following fields: Hash (2 bytes), Type (1 byte) and Seq (1 byte), as presented in Table 1. The Hash field contains the MAC computed from the payloads of the previous and current packets, the secret key, and the sequence number of the current packet. The sequence number is used by the destination node to detect lost packets. The Type field encodes the type of a certain packet and contains the following fields: Auth (1 bit), NACK (1 bit), PT (3 bits) and QoS (3 bits), as represented in Table 2.

The Auth flag is set to 0 during the three-step authentication handshake, and it is set to 1 after the authenticated connection has been established and data packets have been exchanged. The NACK flag is set to 1 in a NACK packet and to 0 otherwise.

TABLE II. TYPE FIELD STRUCTURE

	Type Field			
	Auth	***NACK***	***PT***	***QoS***
Number of bits	1	1	3	3

PT represents the Packet Type and is 001 for H1 packets, 010 for H2 packets, 011 for H3 packets, 100 for Data packets.

The QoS field is used for assigning a priority value to packets. Because it is implemented on 3 bits, it provides for 8 priority levels. Certain packets can be assigned a higher priority than others, such as the re-send or control packets.

IV. EAASP IMPLEMENTATION

The EAASP was implemented in TinyOS, an event-driven, component-based operating system for Wireless Sensor Networks [6].

1) Implementing the security protocol

EAASP consists of two layers, which were introduced in the communication stack of the operating system: the MAC layer and the Authentication layer.

The MACLayerSender component is placed between the AMSenderC component and the ActiveMessageC component, and it has access to all packets sent by the application layer. In the AASP version this layer placed the MAC of the previous and current payload in the analyzed packet, and it stored the MAC of the current payload in the component, for further use. In the current, optimized version, this process has been moved into the Authentication layer, because it stores a predefined number of sent packets for further retrieval. If lost, they can be requested by a NACK packet. This measure avoids duplicating the packet list. Therefore, the MACLayerSender contains only the routing protocol implementation.

The MACLayerReceiver is placed between the ActiveMessageC component and the AMReceiverC component, and it has access to all packets received by the node before reaching the application layer.

If the component receives an out-of-order packet, it stores it, and it delivers it to the upper layers only after all packets with a lower sequence number have also been delivered.

The MAC is computed from the payload of the current packet, the payload of the previous packet, the secret key and the sequence number of the current packet. If the MAC does not match the one found in the Hash field of the packet, the Altered flag is set. If the MAC is correct, the packet is stored for further use when verifying the MAC of the next packet. In either case, the packet is delivered to the upper layers.

The Application layer is placed on top of the operating system, and it uses the components AMSender and AMReceiver to send to and respectively to receive packets from the medium. We cannot send control packets, such as handshake and ACK/NACK packets, from the MACLayer components; therefore, we must divide the Application layer into two sublayers: the Authentication layer and the actual application layer.

Whenever a packet is received by the Authentication layer from the actual application layer, without having an authenticated connection with the destination node, the Authentication layer initiates the three-step handshake by sending an Authentication Request (H1) packet. The Authentication layer performs the handshake in order to establish the authenticated connection.

The Authentication layer keeps track of the sequence number and stores a list of sent packets that can be used when packets are lost and a NACK is received from the destination. The layer computes and writes the protocol header for each sent packet: sequence number, type and hash. If it receives an out-of-order or altered packet, it generates and sends a NACK to the source node. When sending a NACK, a timer is configured to be fired after a predefined period of time. If the lost data packets are not retrieved in that interval, the NACK is considered to be lost and the NACK is resent.

If the Authentication layer receives a NACK, it re-sends all lost packets. If multiple packets are lost, the first packet is re-sent from the Receive.receive() event, and the subsequent packets, except for the out-of-order packet, are sent from the AMSend.sendDone() event. If it receives a correct data packet, it delivers it to the actual application layer and stores the sequence number of this packet.

2) Implementing the routing protocol

TinyOS provides single-hop communication via the Active Messages stack. In order to ensure multi-hop

 ISBN: 978-1-61208-133-5

communication, we introduce layer 3 information in packets and we implement a routing protocol. We use the AM addresses as layer 2 addresses and we introduce a layer 3 source ID and destination ID in the packets.

The MAC layer contains the implementation of a simple routing protocol. This layer stores a routing table that contains the next hop associated with a certain destination. When the MACLayerSender component receives a packet from the application layer, it checks the routing table in order to determine the next hop towards the destination. After that, it sends the packet to the next hop node by setting it as the destination in the AM packet.

When the MACLayerReceiver receives a packet with the layer 3 destination different from the node ID, it checks the routing table to find the next hop and sends the packet to that node. A discussion of routing procedures in the EAASP lies beyond the scope of this article.

V. EVALUATION RESULTS

We evaluate EAASP by determining its energy efficiency and its scalability.

A. Simulation results

A first evaluation relies on several test scenarios implemented with TOSSIM, a simulation tool for TinyOS applications [10].

1) Single-hop scenario

The first test scenario has the purpose of demonstrating basic single-hop functionality. We determine the proportion of lost packets by computing an average value across 20 instances of scenario execution.

Figure 2 presents the TOSSIM output for a single-hop authentication initialization, connection establishment, and data packets exchange.

Each line has the following format: The ID of the node, the component that generates the output, the type of packet sent or received, the fields, the source and destination of the packet.

```
(3): AuthenticationLayer: H1 packet sent [payload=234 hash=56843 type=8 seq=1 (3->1)]
(1): AuthenticationLayer: H2 packet sent [payload=57195 hash=42756 type=16 seq=1 (1->3)]
(3): AuthenticationLayer: H3 packet sent [payload=56185 hash=47406 type=24 seq=2 (3->1)]
(3): AuthenticationLayer: Managed to authenticate myself to node 1
(1): AuthenticationLayer: Managed to authenticate myself to node 3
(3): ApplicationC: Data packet sent [payload=1235 hash=41396 type=160 seq=3 (3->1)]
(1): ApplicationC: Data packet received [payload=1235 hash=41396 type=160 seq=3 (3->1)]
```

Figure 2. Handshake and data packets

We can observe from Figure 2 that the Authentication layer is responsible for performing the handshake and for establishing the connection, and the Application layer has the role of sending and receiving data packets.

In Figure 3, we can observe that a packet with sequence 11 is lost and the subsequent packet is received by the destination. It is detected as an out-of-sequence number and a NACK packet is sent to the source node, which resends the packet. The out-of-order packet is stored in the MACLayerReceiver and it is delivered to the application layer after the lost packet is received. After that, the next packet is sent and received correctly at the destination.

```
(3): ApplicationC: Data packet sent [payload=1243 hash=43279 type=160 seq=11 (3->1)]
(3): ApplicationC: Data packet sent [payload=1244 hash=43260 type=160 seq=12 (3->1)]
(1): AuthenticationLayer: Out-of-order packet received [payload=1244 hash=43260 type=161 seq=12 (3->1)]
(1): AuthenticationLayer: NACK packet sent [payload=11 hash=42686 type=64 seq=2 (1->3)]
(3): AuthenticationLayer: NACK packet received [payload=11 hash=42686 type=64 seq=2 0 (1->3)]
(3): AuthenticationLayer: Data packet re-sent [payload=1243 hash=43279 type=162 seq=11 (3->1)]
(1): ApplicationC: Data packet received [payload=1243 hash=43279 type=162 seq=11 (3->1)]
(1): ApplicationC: Data packet received [payload=1244 hash=43260 type=160 seq=12 (3->1)]
(3): ApplicationC: Data packet sent [payload=1245 hash=43243 type=160 seq=13 (3->1)]
(1): ApplicationC: Data packet received [payload=1245 hash=43243 type=160 seq=13 (3->1)]
```

Figure 3. Lost and recovered data packets

In order to determine the proportion of lost packets we have used a scenario in which we have generated 100 packets, we have counted the number of lost packets and we have computed the percent of lost packets. The scenario has been run for 20 times in order to compute an average value. The resulting average value for the single-hop case is 1.55% lost packets.

2) The multi-hop scenario

As a simple multi-hop scenario we choose a 3 node chain topology and we send packets from one end to another, as presented in Figure 4.

```
(0): RadioCountToLedsC: Data packet sent [payload=1257 hash=43030 type=160 seq=24 (0->2)]
(1): RoutingLayer: Routing packet through 2 [payload=1257 hash=43030 type=160 seq=24 (0->2)]
(2): RadioCountToLedsC: Data packet received [payload=1257 hash=43030 type=160 seq=24 (0->2)]
```

Figure 4. Multi-hop packet routing

To determine the proportion of lost packets for a multi-hop scenario, we have used 10 nodes placed in a chain topology. We have generated 100 packets, we have run the scenario for 20 times, and we have obtained an average of 9.55% lost packets for the 10 node chain topology. The average distance (in hops) from the source node where the packets are lost is 4.45.

B. Energy consumption

We have developed a mathematical model designated as the Sent/Received Bytes Evaluation Model that allows us to determine a measurement of energy consumption in order to

 ISBN: 978-1-61208-133-5

evaluate the control overhead and to compare EAASP with AASP.

Similar mathematical models such as [11] include 2nd layer information, which is not useful when analyzing a security protocol.

In order to evaluate energy consumption, our mathematical model takes in consideration only the number of bytes sent and received by the nodes. We did not include the relatively insignificant levels of energy consumed when executing code on sensor nodes, given that 1 bit transmitted in a sensor network consumes as much power as 800 -1000 instructions [12].

We consider 2 scenarios with the maximum number of hops between two nodes of 10, and respectively 100 hops. For each of these scenarios we transfer 10, 20, 50 and 100 packets between two nodes with the maximum number of hops between them. We use data payloads of 4 and 8 bytes.

Formula (2) evaluates power consumption when no packet is lost. Nb_{H1}, Nb_{H2} and Nb_{H3} represent the size (in bytes) of the handshake packets; Nb_D represents the size of a data packet; NP is the number of packets and NH is the number of hops between source and destination.

$$EC = (Nb_{H1}+Nb_{H2}+Nb_{H3}+ NP*Nb_D)*2*(NH+1) \quad (2)$$

Formula (3) evaluates power consumption when packets are lost and NACKs are used. PPL represents the percentage of lost packets; ADLP represents the average distance where packets are lost, Nb_{NACK} represents the size (in bytes) of the NACK packet. This formula takes into consideration that packets go through a number of nodes before being lost and that NACK packets are used to retrieve those packets.

$$EC = (Nb_{H1}+Nb_{H2}+Nb_{H3}+ NP*Nb_D)*2*(NH + 1) + NP*PPL*Nb_D*(1 + 2*ADLP*NH) + NP*PPL*Nb_{NACK}*2*(NH + 1) \quad (3)$$

1) 10 hops scenario

The results for sending 10, 20, 50 and 100 packets on a 10 hop chain are presented in Table 3 and Figure 5. All values are computed in bytes. Energy consumption for a byte depends on the hardware platform.

We assume that the average distance where the packets are lost is 50% from the total number of hops, a similar situation to the one determined experimentally.

TABLE III. 10 HOPS SCENARIO, 4 BYTE PAYLOADS

Packet loss rate	Number of packets			
	10	*20*	*50*	*100*
No packet loss	2200	3960	9240	18040
10% packet loss	2420	4400	10340	20240
20% packet loss	2640	4840	11440	22440
30% packet loss	2860	5280	12540	24640
40% packet loss	3080	5720	13640	26840

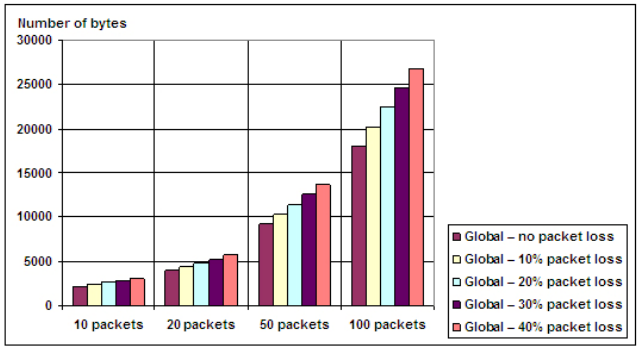

Figure 5. 10 hops scenario, 4 byte payloads

2) 100 hops scenario

The results for the 100 hops scenario are presented in Table 4 and in Figure 6.

TABLE IV. 100 HOPS SCENARIO, 4 BYTE PAYLOADS

Packet loss rate	Number of packets			
	10	*20*	*50*	*100*
No packet loss	20200	36360	84840	165640
10% packet loss	22220	40400	94940	185840
20% packet loss	24240	44440	105040	206040
30% packet loss	26260	48480	115140	226240
40% packet loss	28280	52520	125240	246440

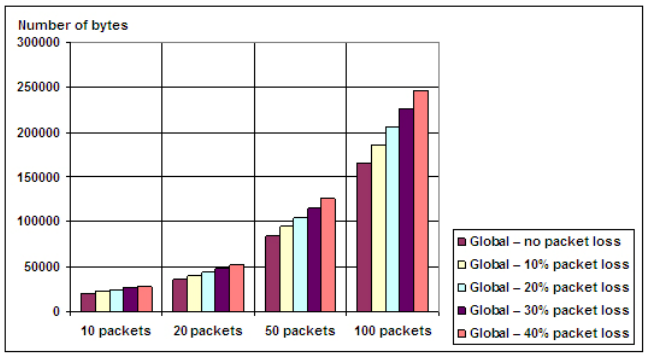

Figure 6. 100 hops scenario, 4 byte payloads

3) Control overhead

We aim to determine the control overhead (CO) of the security protocol for 4 and 8 byte payloads. We use Formula (4) to determine the control overhead. Nb_{HD} is the dimension in bytes of the EAASP header.

$$CO = (Nb_{H1}+Nb_{H2}+Nb_{H3}+ NP*Nb_D)*2*(NH + 1) + NP*PPL*Nb_{HD}*(1 + 2*ADLP*NH) + NP*PPL*Nb_{NACK}*2*(NH + 1) \quad (4)$$

Table 5 and Figure 7 present the control overhead for the 10 hops scenario, for 100 transferred packets.

TABLE V. CO FOR 4 AND 8 BYTE PAYLOAD PACKETS

Packet loss rate	CO bytes	CO for 4 bytes	CO for 8 bytes
No packet loss	9240	51%	34%
10% packet loss	11000	54%	37%

Packet loss rate	CO bytes	CO for 4 bytes	CO for 8 bytes
20% packet loss	12760	57%	40%
30% packet loss	14520	59%	42%
40% packet loss	16280	61%	44%

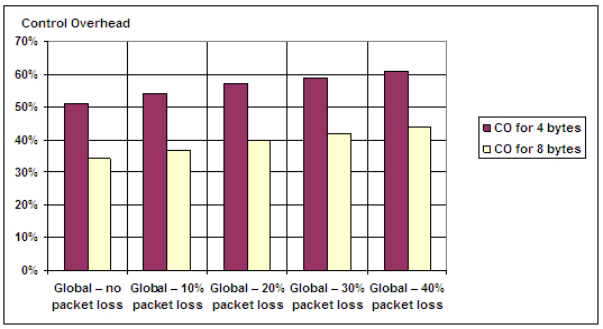

Figure 7. Control overhead for 4 and 8 byte payload packets

The percentage of control overhead decreases as the payload dimension is increased. For 4 byte payloads, CO goes from 51% for no packet loss, to 54% for 10% packet loss, to 61% for 40% packet loss. For 8 bytes, the CO goes from 34% for no packet loss, to 37% for 10% packet loss, to 44% for 40% packet loss.

4) EAASP vs. AASP

We compare the two versions of the protocol in order to determine the extent to which EAASP is more energy-efficient.

The difference in energy consumption between the two versions of the protocol is determined in Formula (5). Nb_H represents the dimension of the handshake packets in AASP; Nb_{ACK} is the dimension of the ACK packet; Nb_{De} is the dimension of EAASP data packet and Nb_{Da} is the dimension of the AASP data packet.

$$AASP - EAASP = [4*Nb_H + NP*(Nb_{Da}+Nb_{ACK} - Nb_{De} - PPL*Nb_{NACK}) - Nb_{H1} - Nb_{H2} - Nb_{H3}]*2*(NH + 1) + NP*PPL*(Nb_{Da} - Nb_{De})*(1 + 2*ADLP*NH) \quad (5)$$

We present results for 10 hops scenario and 4 byte payloads in Table 6 and Figure 8. All values are represented in number of bytes, because the energy depends directly on the number of sent and received bytes.

TABLE VI. AASP VS. EAASP – 10 HOPS, 4 BYTE PAYLOADS

Packet loss rate	AASP	EAASP	Saved energy
No packet loss	42592	18040	24552
10% packet loss	43802	20240	23562
20% packet loss	45012	22440	22572
30% packet loss	46222	24640	21582
40% packet loss	47432	26840	20592

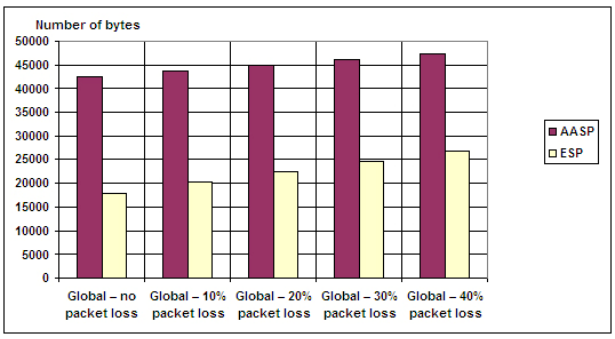

Figure 8. AASP vs. EAASP

Results indicate that EAASP is considerably more energy efficient than AASP: the saved energy amounts to 24KB when no packet is lost, to 23KB for 10% packet loss, and to 20KB when 40% packets are lost. The saved energy decreases slightly as the percentage of lost packets increases.

5) Data aggregation

We consider sending two 4 byte values into a single payload in order to reduce energy consumption.

In Table 7 and Figure 11 we compare energy consumption when sending 50 packets with an 8 bytes payload and when sending 100 packets with a 4 bytes payload, for the 10 hops scenario.

TABLE VII. 8 VS 4 BYTE PAYLOADS

Packet loss rate	50 packets 8 bytes data	100 packets 4 bytes data	Saved energy
No packet loss	13640	18040	4400
10% packet loss	14960	20240	5280
20% packet loss	16280	22440	6160
30% packet loss	17600	24640	7040
40% packet loss	18920	26840	7920

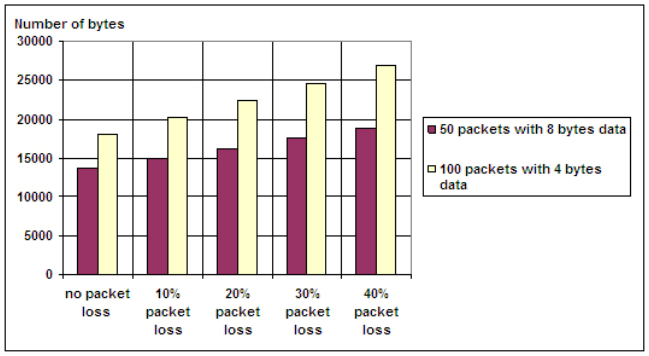

Figure 9. 8 vs. 4 byte payloads

We observe that, in each case, sending 50 packets with 8 byte payloads consumes less energy than sending 100 packets with 4 byte payloads. The saved energy is estimated to 4.4 MB when no packet is lost, 5.2MB for 10% packet loss, and 7.9MB when 40% of the packets are lost. As the percentage of lost packets increases, the saved energy also increases.

 ISBN: 978-1-61208-133-5

VI. CONCLUSIONS

We have previously developed a security protocol (AASP) that provides authentication, integrity, anti-replay and reliability. In this paper we present protocol optimizations that reduce energy consumption: the control overhead has been minimized, the number of handshake packets has been reduced, NACKs have been used for selective data retransmission, and benefits of data aggregation have been evaluated.

We implemented the improved security protocol (EAASP) in TinyOS as two layers in the communication stack, and we have used TOSSIM to run several test scenarios in order to demonstrate its functionality and to evaluate its performance.

We have developed a mathematical model in order to determine energy consumption. In several test scenarios we have estimated the energy consumption, we have evaluated the control overhead, and we have determined the energy saved by optimizations and also by aggregating data.

The formal evaluation proves that EAASP provides substantial energy savings in relation to AASP. Data aggregation can be further used, when possible, to increase energy efficiency.

The Energy-efficient Security Protocol is an appropriate choice when authentication, integrity and anti-replay are required for low-power devices. We have used simulation and mathematical results to prove the energy-efficiency of all introduced optimizations.

In further work we will use the QoS bits to prioritize certain packets, such as negative acknowledgements and re-sent packets, in order to speed up the retrieval of lost packets. The QoS bits may also be used in order to reduce energy consumption caused by traffic congestion.

In a future evaluation, we will compare our security protocol with other state of the art security protocols, such as TinySec and SPINS, regarding energy consumption, and the strength of authentication, freshness and reliability mechanisms.

ACKNOWLEDGEMENTS

The work has been partially funded from the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/88/1.5/S/60203, partially from the national project "Excellence in research through postdoctoral programs in priority areas of knowledge-based society(EXCEL)", Project POSDRU/89/1.5/S/62557 and partially from POSCCE project GEEA 226 - SMIS code 2471, which is co-founded through the European Found for Regional Development inside the Operational Sectoral Program "Economical competivity improvement" under contract 51/11.05.200.

REFERENCES

[1] J. Zheng and A. Jamalipour, *Wireless Sensor Networks: A Networking Perspective*, Wiley-IEEE Press, 2009.

[2] M. Winkler, K.D. Tuchs, K. Hughes, and G. Barclay, "Theoretical and practical aspects of military wireless sensor networks," *Journal of Telecommunications and Information Technology*, vol. 2, 2008, pp. 37-45.

[3] T. Kavitha and D. Sridharan, "Security Vulnerabilities In Wireless Sensor Networks: A Survey," *Journal of Information Assurance and Security*, vol. 5, 2010, pp. 31-44.

[4] D. Boyle and T. Newe, "Security protocols for use with wireless sensor networks: A survey of security architectures," *Third International Conference on Wireless and Mobile Communications*, 2007, pp. 54-59.

[5] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D.E. Culler, "SPINS: Security protocols for sensor networks," *Wireless networks*, vol. 8, 2002, pp. 521-534.

[6] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An operating system for sensor networks," *Ambient Intelligence*, 2004, pp. 115-148.

[7] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," *Proceedings of the 2nd international conference on Embedded networked sensor systems*, ACM, 2004, pp. 162-175.

[8] L. Gheorghe, R. Rughiniş, R. Deaconescu, and N. Ţăpuş, "Authentication and Anti-replay Security Protocol for Wireless Sensor Networks," *The Fifth International Conference on Systems and Networks Communications, ICSNC 2010*, 2010, pp. 7-13.

[9] L. Gheorghe, R. Rughiniş, R. Deaconescu, and N. Ţăpuş, "Reliable Authentication and Anti-replay Security Protocol for Wireless Sensor Networks," *The Second International Conferences on Advanced Service Computing, SERVICE COMPUTATION 2010*, 2010, pp. 208-214.

[10] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," *Proceedings of the 1st international conference on Embedded networked sensor systems*, ACM, 2003, pp. 126-137.

[11] M. Amiri, "Evaluation of Lifetime Bounds of Wireless Sensor Networks," *Arxiv preprint arXiv:1011.2103*, vol. abs/1011.2, 2010.

[12] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *ASPLOS-IX: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, vol. 35, Dec. 2000, pp. 93-104.

 ISBN: 978-1-61208-133-5