Ad Hoc Networks 11 (2013) 2588–2604

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

A dual-mode energy efficient encryption protocol for wireless sensor networks

Abidalrahman Moh'd [a,*], Nauman Aslam [b], William Phillips [a], William Robertson [a]

[a] Department of Engineering Mathematics and Internetworking, Dalhousie University, Halifax, Nova Scotia B3J 2X4, Canada
[b] Faculty of Engineering and Environment, Northumbria University, Newcastle Upon Tyne NE18ST, UK

ARTICLE INFO

Article history:
Received 20 November 2012
Received in revised form 23 June 2013
Accepted 24 July 2013
Available online 11 August 2013

Keywords:
Energy efficiency
Security
Encryption protocols
Wireless sensor networks

ABSTRACT

This paper presents a novel link-layer encryption protocol for wireless sensor networks. The protocol design aims to reduce energy consumption by reducing security related communication overhead. This is done by merging security related data of consecutive packets. The merging (or combining packets) based on simple mathematical operations helps to reduce energy consumption by eliminating the requirement to send security related fields in headers and trailers. We name our protocol as the Compact Security Protocol referred to as C-Sec. In addition to energy savings, the C-Sec protocol also includes a unique security feature of hiding the packet header information. This feature makes it more difficult to trace the flow of wireless communication, and helps to minimize the cost of defending against replay attacks. We performed rigorous testing of the C-Sec protocol and compared it with well-known protocols including TinySec, MiniSec, SNEP and Zigbee. Our performance evaluation demonstrates that the C-Sec protocol outperforms other protocols in terms of energy savings. We also evaluated our protocol with respect to other performance metrics including queuing delay and error probability.

1. Introduction

Wireless Sensor Networks (WSNs) have garnered significant attention from the research community and are becoming very popular in many applications including civil, military, commercial and health care [1,2]. A typical WSN is composed of a large number of tiny resource-constrained sensors, equipped with non-rechargeable batteries. For such devices, wireless transmission consumes much more energy than computation. Therefore, to enhance the operational life of a WSN, it is vital that the amount of communication overhead should be kept as low as possible. In recent years, despite the developments in Micro-Electro-Mechanical systems (MEMs) technology, improvements to energy consumption of communication in the wireless medium are relatively small due to the modest improvements in transceiver technologies and the need to achieve acceptable levels of signal-to-noise ratio at specific distances. It is a well-known fact that the amount of energy consumed in transmitting one bit in the wireless medium is equivalent to the energy consumed to run thousands of clock cycles in most of the current sensor node controllers [3].

Link layer security is of unique importance in WSNs. Unlike traditional networks, realizing end-to-end security in WSNs is extremely difficult, because the data has to be inspected and aggregated on the way to the sink. These operations require the intermediate nodes to access and possibly modify the contents of the packets, which would not be possible if an end-to-end scheme was used. Therefore, more transparent mechanisms are required at the link layer level. In addition, link layer based security minimizes the effect of security attacks because those attacks can be discovered on the next hop once they happen, but can only

* Corresponding author. Tel.: +1 902 402 3210; fax: +1 902 423 1801.
E-mail addresses: Abidalrahman.Mohd@dal.ca (A. Moh'd), nauman.aslam@northumbria.ac.uk (N. Aslam), william.phillips@dal.ca (W. Phillips), william.robertson@dal.ca (W. Robertson).

1570-8705/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.adhoc.2013.07.006

be discovered at the sink in the case of end-to-end security schemes.

Many link-layer security protocols were recently proposed for WSNs [4–7]. The design of those protocols is challenging due to the constrained nature of WSNs that restricts reserving additional resources to provide security services. These resources can be in the form of energy consumption, processing overhead, additional hardware components, and communicated data.

Motivated by the above challenges, we present the C-Sec protocol [8], which is a dual mode encryption protocol for WSNs designed with an objective of reducing communication with a slight increase in the processing. Following Moore's law, dramatic reduction in energy consumption can be achieved from implementing the same hardware with a smaller process technology. In contrast, improvements to energy consumption of communication in the wireless medium are relatively small due to the modest improvements in transceiver technologies and the need to achieve acceptable levels of signal-to-noise ratio at specific distances. The trade-off between the energy costs of communication verses processing formulates the design philosophy of the C-Sec protocol. This protocol is based on reducing the energy-costly communication with a small increase in processing that has much lower-and relatively decreasing energy cost.

The saving in communication energy for the C-Sec protocol is a result of the innovative idea of excluding the requirement for explicitly transmitting all header fields related to security most of the time while keeping all related security services. Such fields include freshness counter, source address, and Message Authentication Code (MAC). The C-Sec protocol provides all the basic security services that are provided by other security protocols in the literature, such as data authentication, integrity, confidentiality, semantic security, and replay protection. However, it adds a new unique security feature of hiding the packet header, making it more difficult to eavesdrop on the flow of wireless communication between nodes to minimize the cost of defending against replay attacks. To our knowledge, this feature does not exist in any previous protocol for WSNs in the literature.

This paper evaluates the performance of the C-Sec protocol and provides a comparison between the C-Sec and related security protocols including MiniSec, SNEP, TinySec, and Zigbee. The evaluation criterion includes the security services provided in the link layer, communication energy consumption, queuing model, delay overhead, and error probability. The evaluation results demonstrate the advantage of the C-Sec protocol in terms of energy consumption over other protocols; it also proves analytically and by simulation that the relation between packets introduced by the C-Sec has limited effect on packet loss due to noise, and on end-to-end packet delay.

The rest of the paper is organized as follows. Section 2 reviews some of the proposed related protocols and discusses their design and implementation. Section 3 presents the design details of the novel C-Sec protocol. Section 4 analyzes the security properties and the selection of cryptographic algorithms for C-Sec. Section 5 presents a thorough performance evaluation of the proposed protocol. Finally, the summary of conclusions and the directions for our future work are presented in Section 6.

2. Related work

Many encryption protocols in the literature were designed to implement basic security services for WSNs. The most recognized are SNEP [4], TinySec [5], Minisec [6], and Zigbee security [7]. The designers of these protocols tried to minimize energy consumption in many ways, but all of them added fields to the header of each packet to implement security services. This section provides a comparison of the security services provided by each protocol and the amount of energy overhead.

WSNs tend to have small packet size because that is optimal for energy efficient communication [9]. In this paper, the TinyOS [10] packet, shown in Fig. 1, is used as a baseline reference for comparison between protocols. This packet has a maximum size of 43 bytes; 13 bytes of header and trailer fields, and a maximum of 30 bytes of payload. The basic header fields are: a 6-byte preamble, 2-byte destination address (*Dest*), a one byte Active Message (*AM*) field which plays a rule like the port number in TCP/IP networks, a one byte length field (*Len*), a one byte group field (*Grp*) used as a unique network identifier to distinguish from other networks, and finally a Cyclic Redundancy Check (*CRC*) trailer field.

Considering its small size, the size of the header is relatively large compared to the size of the packet. Increasing the header size by adding more fields to implement security services will increase the energy needed to transmit the packet.

Although the Zigbee security protocol suite is designed for wireless Personal Area Networks (PANs), which are less constrained than WSNs, it is widely used for WSNs because of its cheap price, and efficient implementation. Moreover, Zigbee includes hardware modules containing cryptographic algorithms which are computationally efficient.

Fig. 2 shows the TinyOS packet after adding header and trailer fields that implement authentication, integrity and replay protection as in the Zigbee AES-CCM-32 security-mode fields as the freshness counter, source address, and message authentication code were introduced to handle security services. Notice that the group field was removed to save energy. The functionality provided by the group field is implicitly implemented with the keying mechanism associated with the MACs. This mechanism provides access control and is sufficient to distinguish between networks. Although the Zigbee security design provides all of the basic security services, the additional security related packet size overhead is relatively high compared to the payload size of WSNs.

The TinySec protocol provides optional two-level security modes. The TinySec-Auth mode does not encrypt the packet payload; it only provides message integrity and

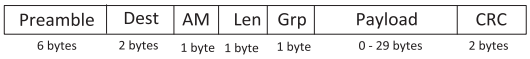


Fig. 1. TinyOS packet.

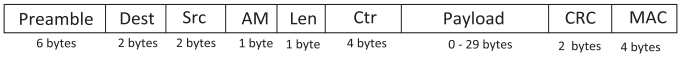


Fig. 2. TinyOS packet with AES-CCM-32 Zigbee security mode.

authenticity. The TinySec-AE mode adds confidentiality and replay protection. TinySec-AE is used in comparison because it provides similar security services compared to other related protocols. Fig. 3 shows the packet format of the TinySecprotocol.

MACs are used to detect malicious changes in communicated packets; they can also be used to detect transmission errors as a replacement for the CRC. The designers of the TinySec, SNEP, and MiniSec protocols make use of this fact to reduce the packet size by removing the CRC field. For further reduction in the header size, the SNEP protocol updates the freshness counters on both sides of communication and does not transmit them. However, it still adds 16% extra communication overhead for each packet due to the large size of the MAC field. Fig. 4 shows the packet format of the SNEP protocol.

The MiniSec protocol uses the Most Significant Bits (MSBs) of some header fields to overload the freshness counter and reduce packet size. The MiniSec has two-levels of security. As shown in Fig. 5, the MiniSec-U mode provides weak replay protection through a 3-bit counter stored in the MSBs of the length field. The MiniSec-B mode has a higher level of security; it uses an 8-bit freshness counter to provide strong freshness protection by adding the other 5-bits of the counter in the place of the five MSBs of the destination address. However, this will reduce the address space to around 2 K addresses instead of 64 K addresses. The MiniSec-B will be used for comparison because it provides equivalent replay protection with other protocols. Both of the MiniSec modes have more than 7% of additional packet size overhead.

The C-Sec protocol has two modes; both of them provide the basic security services of confidentiality, integrity, authentication, semantic security and replay protection. Fig. 6 shows the C-Sec packet formats. C-Sec maintains the freshness counters on both sides of communication without transmitting them, as in SNEP. It uses the most significant bit of the destination address to transmit the mode bit M^o. As opposed to MiniSec-B, this does not dramatically reduce the address space because only one bit is used. MiniSec-B uses 5-bits from the destination address to provide a strong level of freshness protection, which dramatically reduces the address space.

Using the most significant bit of the destination address space to specify the mode saves time and energy by helping the receiver to detect if the packet is conventional or compact as early as possible, and then to decide if the

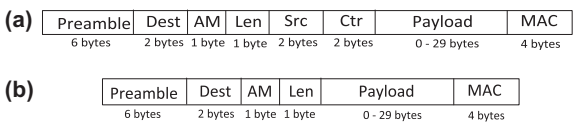


Fig. 3. (a) TinySec-AE packet. (b) TinySec-Auth packet.

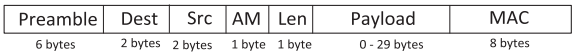


Fig. 4. SNEP packet.

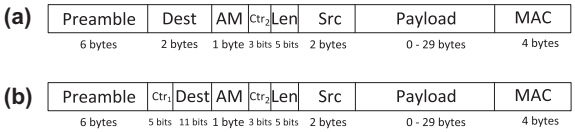


Fig. 5. (a) MiniSec-U packet. (b) MiniSec-B packet.

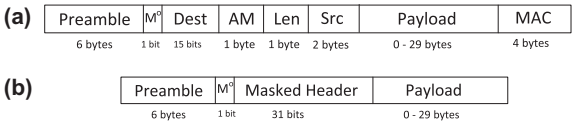


Fig. 6. (a) C-Sec conventional mode packet. (b) C-Sec compact mode packet.

packet should be accepted or rejected after receiving the destination address or the masked header based on the mode. Using one bit from another field will require the receiver node to wait longer, and consume more energy because it receives more bits before it can identify the mode and then decides whether to accept or early-reject the packet after that.

The compact mode of the C-Sec has smaller packet size because it does not explicitly transmit the MAC and the destination address fields but merges them with the header of next packet. This will result in hiding the header fields and it does not affect the basic security services. The packet size of the compact mode of the C-Sec protocol is 7% less than the packet size of the plain TinyOS packet that does not implement any security service. Using this mode will result in reducing the communication energy to less than the energy consumed in the "no security" case. Table 1 summarizes the additional packet size overhead for these protocols with reference to the TinyOS packet with no security.

3. Protocol design

To start the C-Sec protocol, node authentication-and encryption keys have to be exchanged in advance. We assume that communicating parties have already exchanged those keys using any key management protocol for WSNs, such as μTesla [4] or LEAP [11].

The C-Sec protocol operates in one of two modes, conventional or compact. The conventional mode adds security related data such as the source address and the MAC to the packet header and trailer fields. The header H_i consists of the basic fields only, such as the Mode bit M^o, the

Table 1
Additional security related packet size overhead (bytes).

Protocol	MAC	Source address	Freshness counter	CRC	Group ID	Total	%
Zigbee	4	2	4	2	0	12	21
TinySec	4	2	2	0	0	8	12
SNEP	8	2	0	0	0	10	16
MiniSec	4	2	0	0	0	6	7
C-Sec Conventional	4	2	0	0	0	6	7
C-Sec compact	0	0	0	0	0	0	−7
TinyOS (no security)	0	0	0	2	1	3	0

Destination address *Dest*, the Active Message *AM* fields and the Length *Len*. This mode can be described with Eqs. (1)–(3), where H_i is the header of and T_i is the trailer of the message M_i:

$$T_i = MAC(K_{Auth}, AM_i, Len_i, E_{K_{Encr}}(M_i), C_i) \quad (1)$$

$$H_i = M_i^o : Dest_i : AM_i : Len_i \quad (2)$$

$$A \rightarrow BH_i : E_{K_{Encr}}(M_i) : T_i \quad (3)$$

The C-Sec protocol is initiated in the conventional mode. On both sides of a communication, the sender and receiver should store the MACs of the first packet communicated in the conventional mode. The MAC is computed by running a one-way hashing algorithm with the encrypted message $E_{K_{Encr}}(M_i)$, authentication keyK_{Auth}, and freshness counter C_i as inputs. On the sender side, instead of adding the MAC to the message as a standalone field (the practice of all other protocols); the MAC of the previous encrypted message $E_{K_{Encr}}(M_{i-1})$, is XORed with the header of the current message H_i to produce a new Masked Header MH_i for the current message M_i. Eqs. (4) and(5) and Fig. 7 illustrate the protocol behavior

$$MH_i = H_i \oplus MAC(K_{Auth}, AM_i, LEN_i, E_{K_{Encr}}(M_{i-1}), C_{i-1}) \quad (4)$$

$$A \rightarrow BMH_i : E_{K_{Encr}}(M_i). \quad (5)$$

The compact mode of C-Sec is automatically initiated starting from the second packet on the sender side. All following packets are sent in the compact mode as long as the time between them and their previous packets remains less than a specific time limit called the Authentication Timer. If the authentication timer expires for any packet, it will be sent in the conventional mode. The conventional mode can also be enforced on demand in the case of high bit error rates or QoS time constraints.

The flow diagram in Fig. 8 shows the behavior the C-Sec protocol on the receiver side. Once a packet is received at the destination, the mode of the packet is specified based on the mode bit M^o. If the packet is conventional, the sender address is looked up in the sender-cache. If a match for sender j is found in the sender-cache, the MAC is verified and stored and the packet is accepted. If the sender address is not found in the sender-cache the authentication algorithm has to be run to decide whether the packet is authentic. If so, the packet is accepted, its MAC is added to the sender-cache, and the authentication timer is set for sender j. Otherwise, the packet is dropped.

If the mode is compact, the part of the masked header MH_i that maps to the destination address is XORed with the corresponding part of the pre-computed MACs of the last received packets from all senders in the sender cache. If the result of any of the XOR operation produces the receiver's address D, the packet is held and attributed to the sender j associated with that MAC. The receiver will continue receiving the packet, set the authentication timer, decrypt the payload using the session keyK_{Encr}, and update the value of the freshness counter C_i. The body of the packet will be pending for authentication when the next packet header arrives. If the authentication timer expires before the next packet from that sender arrives, the packet i is dropped and the authentication data associated with it is deleted from the sender-cache. When the next packet from sender j arrives, the algorithm is repeated.

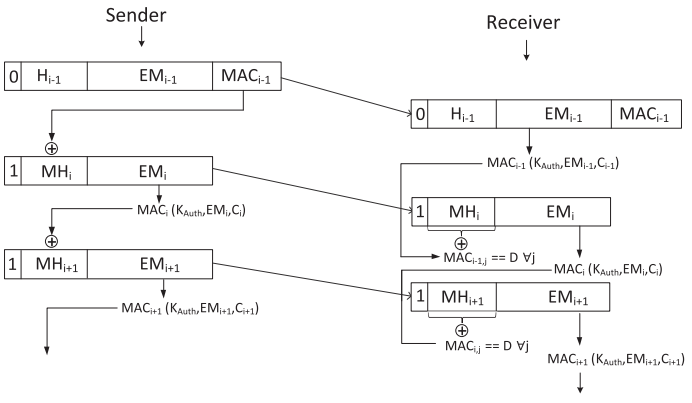


Fig. 7. The C-Sec protocol.

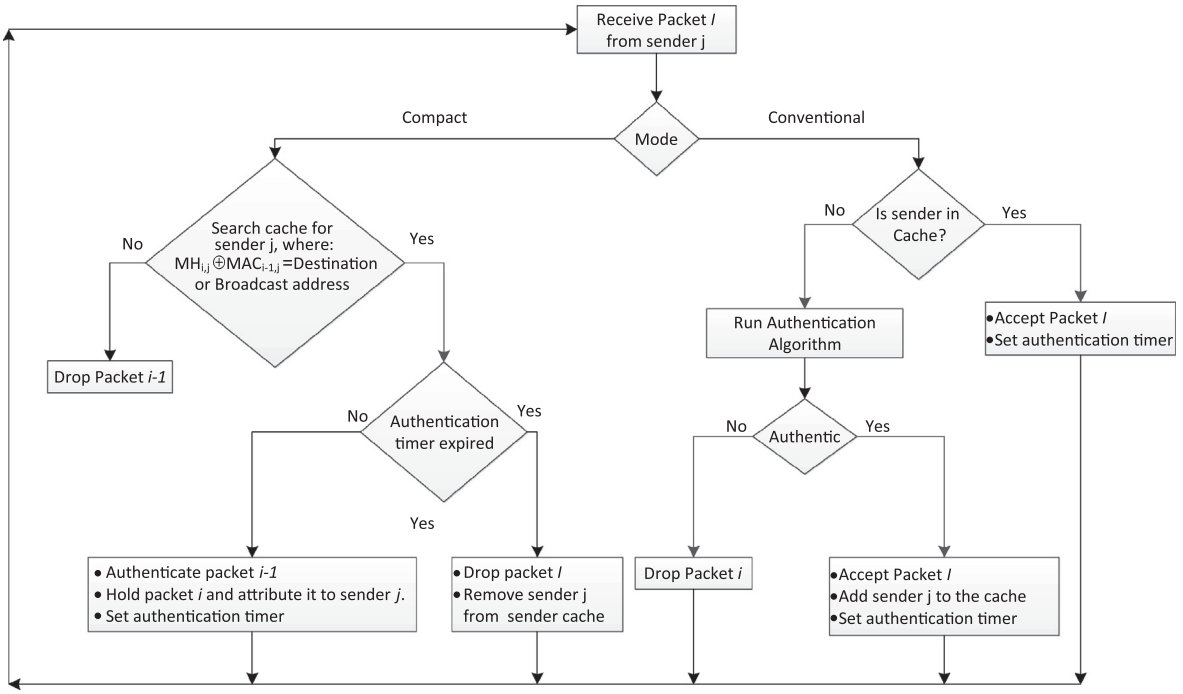


Fig. 8. The behavior of the C-Sec on the receiver side.

If a sender node needs to start a broadcast, it will send the first broadcast packet in the conventional mode. The MAC of the first broadcast packet will be stored in the receiver cache of the receiving nodes. The next broadcast message is sent in the compact mode. The result of XORing the masked header of the second broadcast packet with the MAC of the first broadcast packet will result in the broadcast address (15-bits of all ones). Accordingly, the second compact packet is accepted as a broadcast packet.

If none of the XOR operations produce the destination address of the receiving node or a broadcast address, the packet will be "early dropped" for one of three reasons. Firstly, it could be addressed to another node. Secondly, it could have a transmission bit error. Thirdly, it could be a malicious packet injected by an attacker.

To prevent active attacks in the compact mode, the body of the received packet will be held back in order to be authenticated when the next packet header arrives. In cases of high priority messages or quality of service constraints, the sender can flexibly switch back and forth between the conventional and compact modes in order to prevent waiting for the next packet. If the next packet is not expected to be ready within an acceptable time frame, the authentication data can be sent with a standalone packet after a timeout period.

In early overhearing avoidance mechanisms [12], if the packet is not addressed to it, the receiver instantly stops receiving and moves the transceiver to a low power state before the whole packet is received. This can be decided when the receiver starts receiving the destination address and compares it to its own address. To gain the highest advantages of this mechanism, MAC computation and lookup should be completed within a limited time period (i.e. the time between the end of receiving packet i and the end of receiving the preamble and the masked header of the next packet $i+1$). Meeting this time constraint requires implementing the MAC algorithm in hardware. Hardware implementations are much faster and have much lower energy consumption than their software counterparts up to a factor of 10^{-3} [13]. For example running a software implementation of Cipher-based Message Authentication Code (CMAC) algorithm on TelosB platform for a 24 bits data packet takes 2.5 ms and consumes 13.24 μJ [14], whereas running the hardware based MAC scheme proposed in SN-SEC [13] takes 1.06 μs and consume 28.74 nJ.

Usually, nodes closer to the sink handle more traffic from the network than other nodes, hence consume more energy, and die first resulting in a disconnected network with potentially uncovered areas [15]. Our protocol helps to extend the lifetime of those nodes. The amount and frequency of the traffic they handle allows them to use the compact mode more often than other nodes, and hence save more energy and extend the lifetime of the WSN.

Masking the header of the message with the MAC makes it confidential; as a result, it will be more difficult to trace the flow of wireless communication. To get the header information, an adversary needs to get the authentication key K_{Auth} to re-compute the MAC_{i-1} and then XOR it with the masked header MH_i to resolve the original header H_i. This feature does not exist in any of the previously proposed protocols for WSNs in the literature.

The C-Sec compact mode is built based on – and has the same level of reliability as – stop and wait Automatic Repeat request (ARQ) strategy. Stop and wait is the most commonly used ARQ strategy for WSNs [3,16]. In this

strategy, the sender waits for the acknowledgment of the current packet before it sends the next one. Other ARQ strategies such as window-based ARQ could not be used in the C-Sec compact mode because in the case of out of order messages, the header of the packet (i.e. destination address) cannot be identified unless all pervious messages in the window have arrived. This requires storing all traffic in the wireless medium, including traffic destined to other nodes, and performing an exhaustive search for the correct message, which is not efficient.

4. Security services

This section analyzes the security services provided and the underlying cryptographic algorithms for C-Sec and compares it with other related protocols. This includes:

4.1. Data confidentiality and semantic security

Confidentiality is achieved by encrypting the data with a secret key that is only known by the intended receivers. Semantic security ensures that the adversary cannot learn anything about the plaintext from the ciphertext. This is usually achieved by using block cipher modes of operation that support semantic security.

As shown in Table 2, RC5 and Skipjack were selected for encryption protocols implemented in software because of their relatively smaller code size compared to software implementations of other encryption algorithms. However, AES algorithm was chosen for C-Sec. AES is an NIST standard that was selected over other algorithms because of its proven security and efficient implementation in hardware and software [17]. However, hardware implementation of AES was chosen because it is more efficient in energy and latency; and relatively more secure than software implementations [13].

The encrypted message is computed by running the AES algorithm using Cipher Feed Back (CFB) mode. In this mode of operation the encrypted message does not need to be padded to a multiple of the cipher block size and can have a variable size [16], which will save communication energy. The initial vector required to initialize the CFB mode is defined as:

$$IV = MAC(K_{Encr}, K_{Auth}, C_0) \tag{6}$$

4.2. Data authentication and integrity

Data integrity helps the receiver to ensure that the received data is not altered by an adversary during transmission. Data authentication allows verifying that the data is sent by the claimed sender. In C-Sec, these properties are achieved by computing a message authentication code MAC using the secret authentication key, K_{Auth} that is only known by the sender and the receiver. When the receiver verifies the correctness of the MAC of the received message it ensures that it was sent by the claimed sender who has the, and K_{Auth} that it was not altered during transmission.

As shown in Table 2, all protocols use an encryption algorithm with cipher block chaining CBC-MAC to produce the MAC. This method does not require implementing a separate MAC algorithm; however, it is not secure for variable-length messages [18]. Thus, we have chosen to use a standard hashing algorithm to produce the MAC. Another advantage is that the hashing algorithm can run in parallel with the encryption algorithm. This helps to meet the timing constraint of the compact mode. We have chosen to use a MAC based on SHA-256 for two reasons. First, both SHA-1 and MD5 have known weaknesses and were reportedly broken [19–21]. Second, it has higher implementation efficiency compared to other secure hashing algorithms [22]. To produce the MAC, SHA-256 is run two times as stated in the Eq. (7)

$$MAC = SHA((K_{Auth} \oplus opad) \| SHA((K_{Auth} \oplus ipad) \| M)) \tag{7}$$

where *ipad* and *opad* are paddings used to fit the key to the hashing algorithm key block and M is the message to be authenticated.

4.3. Data freshness and reply protection

Data freshness is used to prevent an adversary from playing old messages. This is achieved by ensuring that the sent data is recent through maintenance of a freshness counter on both sides of the communication. This counter is used in the computation of the MAC of the message as shown in Eq. (1).

In case of traditional protocols, if a replayed packet is sent to a node, it will not discover that until it completely receives it and runs the authentication algorithm with the freshness counter and then compares the resulted MAC with the MAC transmitted with the packet. However, if an old packet is being replayed in the compact mode of C-Sec, the computed destination address will not be correct because the counter is wrong and the packet can be dropped without completely receiving it and without the need to run the authentication algorithm to evaluate its MAC. The cost of replay attacks is reduced by the resulting energy savings.

The C-Sec can be used as attack detection technique. An adversary cannot get any information from a packet in the compact mode because the header is hidden and the

Table 2
Security algorithms comparison.

Protocol	MAC algorithm	Encryption algorithm	Implementation	Encryption key/block/rounds
SNEP	CBC-MAC	RC5-CTR	Software	64/128/18
TinySec	CBC-MAC	Skipjack-CBC	Software	64/80/32
Zigbee	CBC-MAC	AES-CCM	Hardware	128/128/10
MiniSec	CBC-MAC	Skipjack-CBC	Software	64/80/32
C-Sec	HMAC	AES-CFB	Hardware	128/128/10

payload is encrypted. It has to jam the network to create timeouts in the authentication timer to force the node to move to the conventional mode to at least get the packet headers. The percentage of conventional packets and the pattern of forcing the traffic to the conventional mode can give a good indication of attacks on the node.

4.4. Other security issues

C-Sec is a link layer protocol; its security is dependent on protocols and services in other layers, e.g. key management protocols. C-Sec does not explicitly deal with node compromise or physical tampering, and it does not address information leakage through covert channels. However, certain hardware specific measures can be used to prevent physical tampering [23].

5. Performance evaluation

This section presents a simulation as well as analytical models to evaluate the C-Sec protocol. It also compares the C-Sec protocol to other related protocols in the literature and with the TinyOS as a baseline to show the amount of improvement in energy efficiency C-Sec can achieve over other protocols, with different payload sizes and traffic densities. The additional overhead introduced by C-Sec and its effect on the end-to-end delay are also investigated in this section.

The simulation model for the C-Sec and other related protocols is based on the Castalia simulator [24], which is based on OMNeT++ [25] an event driven simulation platform. The simulation is done for a multi-hop single sink data-gathering network of 30 semi-random deployed nodes and with an area of 70×70. The T-MAC is used as the underlying MAC protocol. The value of the authentication timer is carefully chosen to cover more than two T-MAC cycle times, and allow reasonable time for contention on the wireless medium and to put a limit on the size of authentication data stored at the sender and receiver nodes. Table 3 shows the simulation parameters. All other parameters are set to their default values in the Castalia simulator.

Table 3
Simulation parameters.

Parameter	Value
Area size	70×70
Number of nodes	30
Link layer protocol	T-MAC
Transceiver	CC2420
Payload size	30 bytes
Physical layer overhead	6 bytes
Sensitivity	−95 dBm
Noise floor	−100 dBm
Event interarrival time	5 s
Packets generated by event	5
T-MAC frame time	610 ms
Authentication timer	1750 ms

5.1. Energy analysis

As explained in Section 3, nodes executingthe C-Sec protocol start communication in the conventional mode initially, and then switch to the compact mode. During transmission if the authentication timer expires, the communication goes back to the conventional mode. To evaluate the energy savings by the C-Sec protocol the total number of packets generated in the compact mode and the conventional mode is observed. The simulation is run for a period of 120 s. The results for this experiment are illustrated in Fig. 9 highlighting the variation of compact and conventional mode packets with respect to time.

To compare the communication energy consumption of the C-Sec and the other related protocols, the simulation was run for one thousand seconds and repeated with various packet payload sizes and the average energy consumed by a transceiver is computed with a confidence level of 95%. As Fig. 10 shows, the energy consumption increases with the increase in the payload size. It is also observed that the energy consumption varies among the protocols based on the amount of increase of the header size each protocol adds to the packet. The C-Sec consumes less energy than the baseline (TinyOS); this means that the C-Sec manages to provide all the required security services with saving on energy consumption.

It can also be inferred from the results in Fig. 10 that the C-Sec saves more energy than all other protocols with smaller payload sizes, which are more common in WSNs. For example the energy saving for the maximum payload size of 30 bytes is 11.7% compared to MiniSec, the encryption protocol with the closest packet size. However, for a payload size of 5 bytes the energy savings increases to 19.1%.

The energy consumption using different packet loads i.e. packets received per second, a fixed payload size of 30 bytes, and 95% confidence is evaluated. As shown in Fig. 11, the increase in the energy consumption depends on the amount of additional header size overhead each protocol adds to the packet. The C-Sec protocol uses less energy consumption than other protocols. The energy saving increases with higher packet loads because of the increased percent of compact packets, the energy savings reaches up to 11.7% at around 50 packets per second compared to MiniSec, but the gap decreases again to around 9.5% when the network is more saturated.

5.2. Computational overhead

The computational overhead of the compact mode of the C-Sec is defined as the number of CPU cycles needed to unmask the received packet or to decide to drop it. This is a new overhead added by the C-Sec and does not exist in other related works. Checking if the received packet is destined to the receiver node involves comparing its masked header with the MACs of the last received packet from each sender in the receiving node cache.

Fig. 12 shows the cache search algorithm used to determine if a received packet in the compact mode is accepted or rejected. Deciding if the packet is to be accepted or rejected at any iteration of the algorithm takes a maximum of four clock cycles, assuming that each condition of the

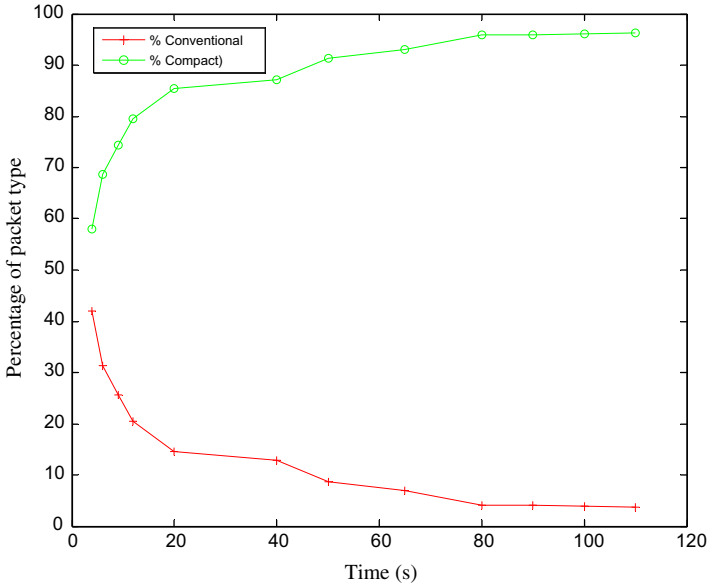


Fig. 9. C-Sec packet breakdown.

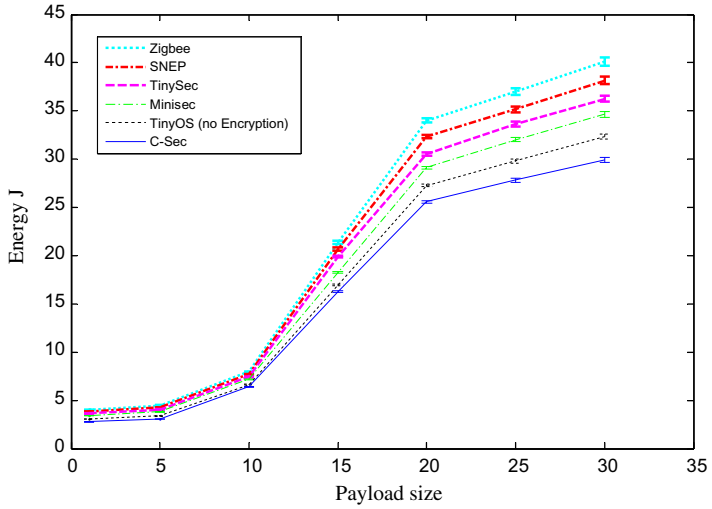


Fig. 10. Communication energy vs. payload size.

algorithm can be checked with one clock cycle i.e. a 32-bit processor.

The authentication timer is evaluated in the first clock cycle. If the authentication timer is expired, the packet is dropped and the MAC being tested is deleted from the receiver cache. If it did not expire, the result of XORing the masked header of the received packet with the MAC being tested is computed in the second clock cycle. The third clock cycle verifies if the result of the previous step is all ones, which means that the received packet is a broadcast packet. If not, the fourth clock cycle verifies if the result is the destination address of the receiving node, which means that the packet is a unicast packet destined to the receiving node. Otherwise, the packet will be dropped. These steps are repeated for each MAC in the receiving node cache.

To evaluate the computational complexity of the C-Sec protocol, simulations have been conducted with variable node densities (number of nodes ranging from 15 nodes to 1200 in an area of $70 \times 70\ m^2$). The packet size is set to one byte and the number of events per second is set to one thousand in order to guarantee generating the maximum possible packet load. The energy needed to run the cache search algorithm in Fig. 12 was computed by counting the clock cycles needed to run the algorithm, assuming that one instruction consumes 1 nJ.

As Fig. 13 shows, the energy consumption to evaluate one received packet increases as the node density increases. However, beyond the density of 0.2 nodes per square meter it becomes stable at around 0.7 μJ because the maximum number of packets with various destination

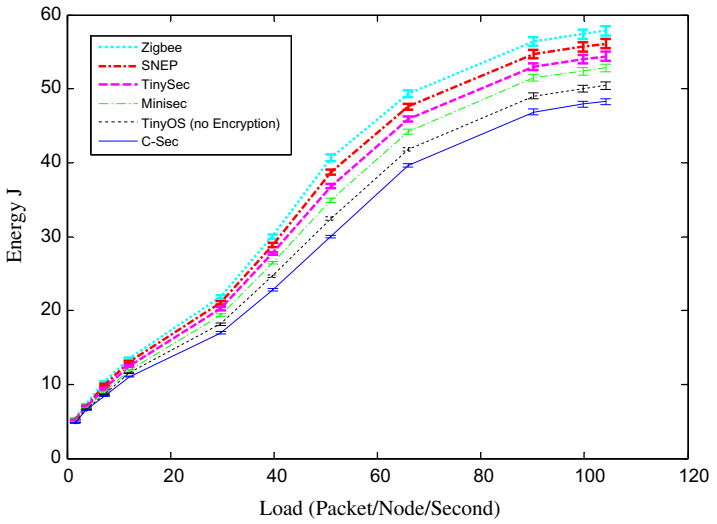


Fig. 11. Communication energy vs. load.

```
1.  For j = 1; j ≤ Cache size; j++
2.      If ( Auth. timer of MAC_j is expired )
3.          Early drop the packet i
4.          Delete MAC_j from receiver cache
5.          Break
6.      Else
7.          Temp ⇐ MH_i xor MAC_j
8.          If ( Temp = Destination Address )
9.              Accept the packet i as a unicast packet
10.             Break
11.         Else if ( Temp = Broadcast Address )
12.             Accept the packet i as a broadcast packet
13.             Break
14.         End if
15.     End if
16. End for
```

Fig. 12. Cache search algorithm in the receiving node cache.

addresses that can be sent within the authentication timer period is reached at that point. Packets delivered outside the authentication timer window are sent in the conventional mode. Considering that the energy needed to transmit one bit in the wireless medium is equivalent to more than one thousand clock cycles for most of the WSN nodes [3] the energy needed to compute that is small compared to the communication energy gain of the C-Sec.

5.3. Delay and queuing analysis

The compact mode of C-Sec protocol introduces relations between packets; a packet cannot leave the node before the next packet arrives if it is the only packet in the output buffer. This will introduce additional delay overhead that did not exist before for that packet; to evaluate this delay a simulation is run to evaluate the end-to-end delay for C-Sec and MiniSec at 20 packets per second, for one thousand seconds. The MiniSec protocol is used for comparison because it has the smallest packet size among other protocols and the closest to C-Sec.

Figs. 14 and 15 show the histograms for end-to-end packet delay results with 95% confidence. It can be observed that C-Sec introduces additional delay overhead of about 200 ms on average, and more packets wait more than 800 ms for the C-Sec compared to MiniSec. This is a direct result for the relation between packets introduced by the C-Sec.

To study this delay more deeply, the mathematical queuing model and delay analysis used in traditional protocols like the MiniSec is investigated. These protocols follow the M/M/1 queuing model, which represents a system having a single server, where arrivals are determined by a Poisson process and job service times have an exponential distribution [26]. This model is compared to a modified model that considers the new condition introduced by the C-Sec that a packet cannot depart until the next packet arrives even if the service is finished. This restriction happens only when there is one packet in the queue.

An M/M/1 queue is described by the following parameters:

λ: Arrival rate, interarrival times are exponentially distributed with a mean of $1/\lambda$.
μ: Departure rate, service times are exponentially distributed with a mean of $1/\mu$.

As Fig. 16 shows, the C-Sec queuing model can be described as a variation of a continuous time markov chain, which is described by transition rates between states. The states are the number of packets in the system.

Note that all transitions from state n to state $n+1$ are λ except for the case, where $n = 1$. For this case we have to take into account whether the packet is being processed or waiting for the next packet to arrive. Solving this system will reveal the expected value of number of packets in the system in Eq. (8), where $\rho = \frac{\lambda}{\mu}$. The complete solution is shown in Appendix A

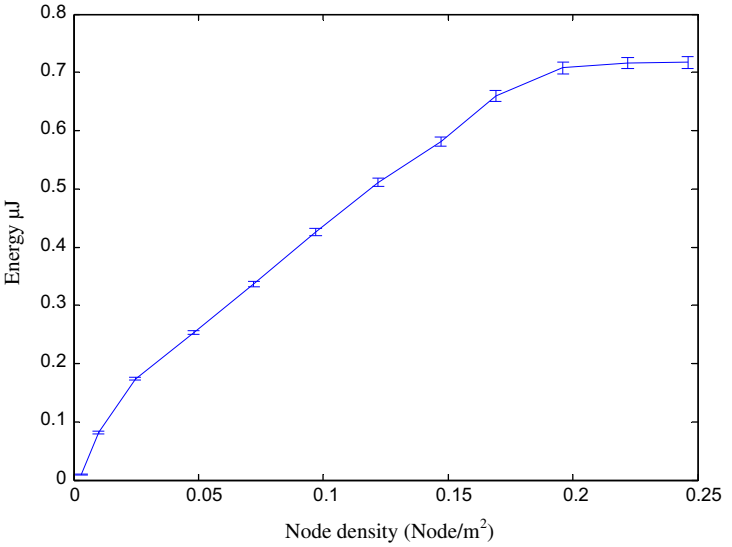


Fig. 13. Computational overhead of the compact mode of C-Sec.

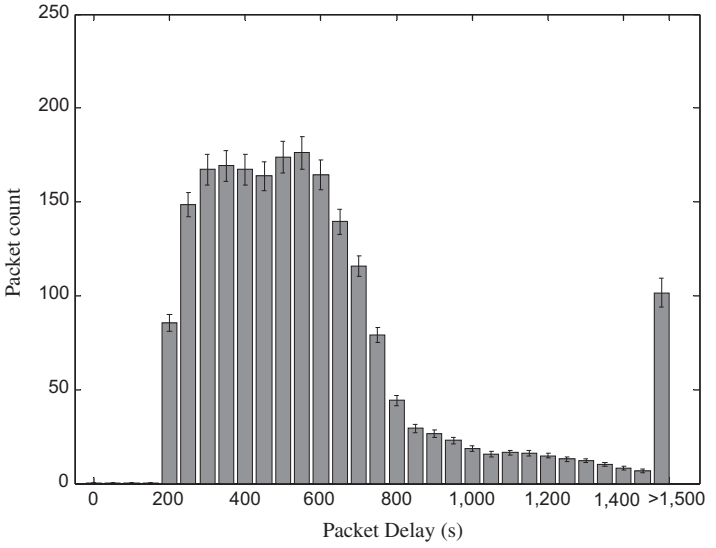


Fig. 14. Histogram of average end-to-end packet delay per node for C-Sec.

$$E_{C-Sec}(\textit{Number of packets in the system}) = 1-\rho+\frac{\rho}{1-\rho} \tag{8}$$

The expected value of the number of packets in the system for the M/M/1 queuing model is:

$$E_{M/M/1}(\textit{Number of packets in the system}) = \frac{\rho}{1-\rho} \tag{9}$$

Comparing Eq. (8) and (9), it can be noted that difference between the queuing models is $1-\rho$. This value is less than 1 because $\rho > 1$. We can conclude that expected value of the C-Sec queue size is one packet greater than the M/M/1 queue in the worst case and the difference approaches to zero when the network is more utilized.

Using Little's law, the expected time in the system:

$$\begin{aligned} E_{C-Sec}(\textit{time in the system}) &= E_{C-Sec}(\textit{packets the system})/\textit{arrival rate} \\ &= \left(1-\rho+\frac{\rho}{1-\rho}\right)\Big/\lambda \\ &= \frac{1}{\lambda}-\frac{1}{\mu}+\frac{1}{\mu-\lambda} \end{aligned} \tag{10}$$

The expected value of waiting time for the M/M/1 queuing model which is given by:

$$E_{M/M/1}(\textit{time in the system}) = \frac{1}{\mu-\lambda} \tag{11}$$

Fig. 17 shows the expected value of waiting time for both the C-Sec and the M/M/1 queuing models. Compared to the expected value of the waiting time for the M/M/1 queue, C-Sec has higher waiting time for low of arrival

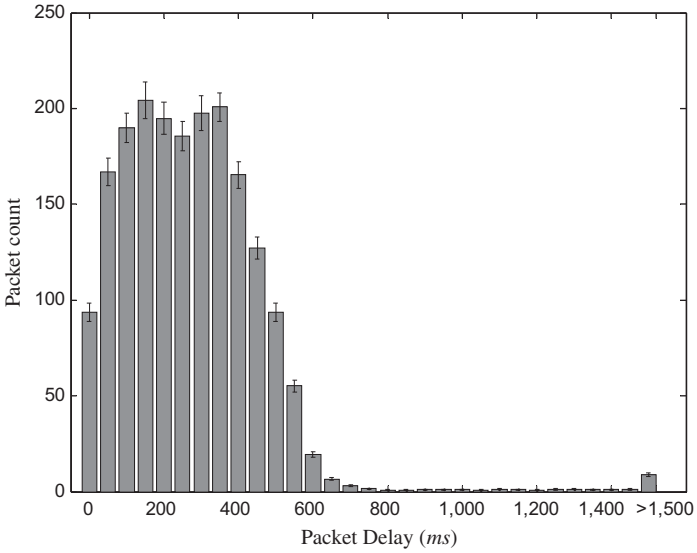


Fig. 15. Histogram of average packet delay per node MiniSec.

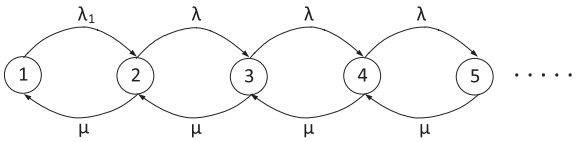


Fig. 16. Markov Chain for C-Sec.

rates. This is because single packets in the queue are held until the next packet arrives. However, C-Sec approaches the M/M/1 queue behavior for larger arrival rates because the likely hood of a single packet in the queue is small, which is the only difference between C-Sec and M/M/1 queuing models.

Eq. (12) shows the difference of expected time in the system between the C-Sec and M/M/1 models, which is the average waiting time introduced by the C-Sec over the classic M/M/1 queuing model

$$E_{C-Sec}(\textit{time in the system}) - E_{M/M/1}(\textit{time in the system}) = \frac{1}{\lambda} - \frac{1}{\mu} \quad (12)$$

Fig. 18 compares the value in Eq. (12) with simulation results of the additional waiting time introduced by C-Sec, i.e. the average period of time the current compact packet waits until the next packet arrives when it is the only packet in the system. It can be inferred that as the traffic load increases, the additional waiting time introduced by the C-Sec queuing model decreases. The two curves have similar trends; however, the additional waiting time introduced by the C-Sec protocol in the simulation results is slightly higher. This is because it is affected by the duty cycle of the T-MAC and packet retransmissions

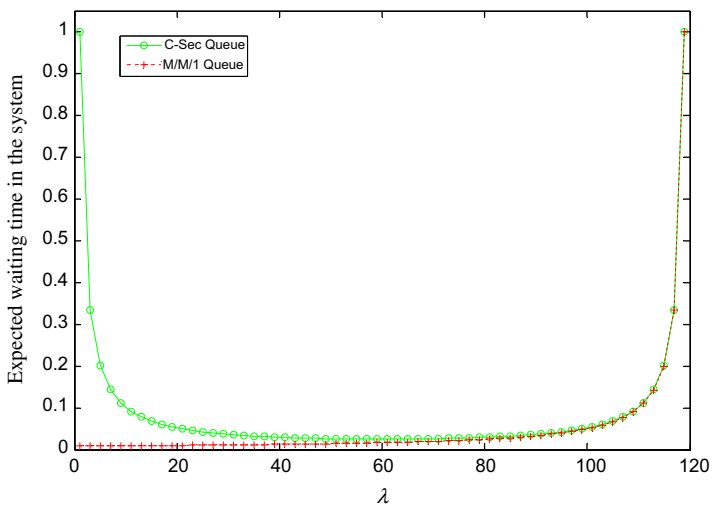


Fig. 17. Expected value of waiting time in the system ($\mu = 120$).

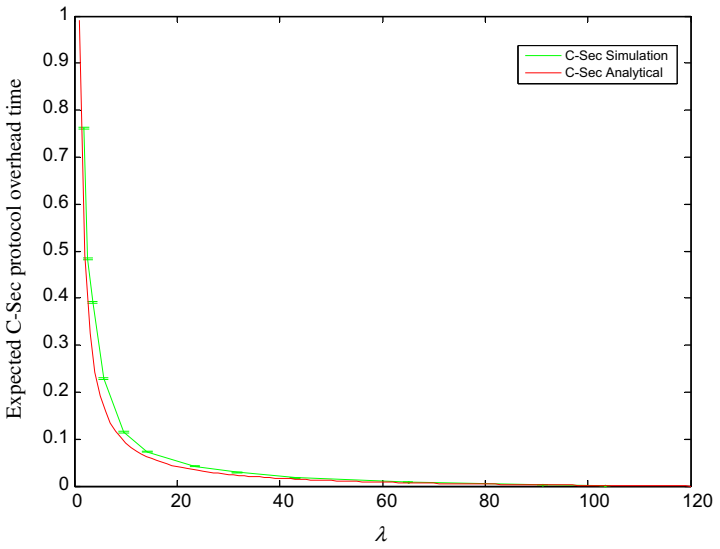


Fig. 18. Average C-Sec overhead time ($\mu = 120$).

due to bit errors. For lower values of packet arrival time, the simulation results shows less overhead time, that is because packets with waiting times larger than the authenticationtimer are transmitted in the conventional mode, this puts a cap on the maximum waiting time for compact packets.

Generally, most of WSNs applications do not require hard real-time constraints [27], and such additional packet delays can be tolerated. The importance of the energy savings the C-Sec can achieve outweighs the additional packet delay it introduces.

5.4. Error probability

This section analyzes the impact of the dependency between packets introduced by the C-Sec protocol on error probability. We extend stochastic models that measure the impact of block cipher encryption on post decryption bit errors [28–30] to measure the impact of packet dependency introduced by C-Sec using Markov characterization stochastic model.

For typical encryption protocols, the packet consists of two parts, the header block and the encrypted message block. If a bit error occurs in the encrypted message block during transmission in the wireless channel, this error will expand to all bits of the decrypted message with a probability of 50%; this is a result of the Strict Avalanche Criterion (SAC) of the encryption algorithms [31]. Some researchers tried to avoid the effect of SAC by inventing new encryption algorithms [32,33]. Such algorithms trade off security with performance and are not scrutinized enough to be used for wireless communication.

The packet can be divided into two parts, an encrypted body that is affected by the SAC and the header. If a bit error occurs in the encrypted body of the message, it will expand as a result of the SAC effect. However, if it happens in the header of the message, only that bit will be changed in the receiver side. As show in Fig. 19, this will result in four different possibilities for the bit error.

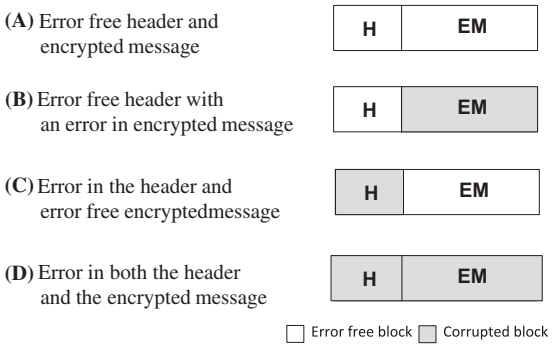


Fig. 19. error event states.

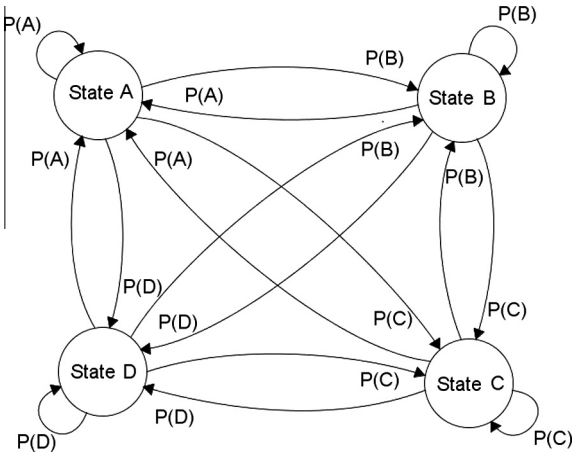


Fig. 20. State diagram for error events at decryption process.

Let b_i denote the ith bit of the header block. $\forall i = 1 \ldots H$. Let $b_1 b_2 b_3 \ldots b_H$ be the transmitted header bits, and $b'_1 b'_2 b'_3 \ldots b'_H$ be received header bits.

Let $P(b'_i|b_i)$ denote the probability of receiving b'_i when b_i is transmitted, and $P\left(b'_1 b'_2 \ldots b'_{H_l} \middle| b_1 b_2 \ldots b_H\right)$ denote the probability that that the received header is $b'_1 b'_2 \ldots b'_H$ when the transmitted header is: $b_1 b_2 \ldots b_H$. Assuming reception of each bit is independent of all the remaining bits, then:

$$P(b'_1 b'_2 \ldots b'_{N_c}|b_1 b_2 \ldots b_{H_c}) = P(b'_1|b_1).P(b'_2|b_2) \ldots P(b'_H|b_H) \tag{13}$$

If, in a certain received block, i bits are in error, then $H - i$ bits are correct. Then the probability of receiving this block will be $P_{bit}^{i}(1 - P_{bit})^{H-i}$. There are $\binom{H}{i}$ different ways in which i errors can occur in H bits. Hence, if the header of the message is sent and it contained i bits that had errors, then $H - i$ bits are correct. Then, the probability of receiving a header with i errorsis:

$$P(i - errors\ in\ the\ header) = \binom{H}{i} P_{bit}^{i}(1 - P_{bit})^{H-i} = \binom{H}{i} P_{bit}^{i} Q_{bit}^{H-i}.$$
$$Where\ Q_{bit} = (1 - P_{bit}) \tag{14}$$

The probability of a receiving correct header:

$$P(header\ is\ correct) = \binom{H}{0} P_{bit}^{0} Q_{bit}^{H-0} = Q_{bit}^{H} \tag{15}$$

Similarly, the probability of receiving a correct encrypted message payload:

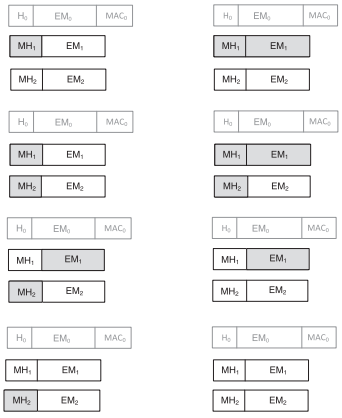
$$P(EM\ is\ correct) = Q_{bit}^{EM} \tag{16}$$

Depending on the state of the received header and message body at each decryption cycle, the decryptor node can be in one of the four states described in Fig. 18. The probability of receiving a message with a correct header and body is:

A

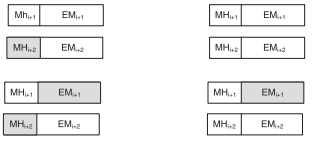
$$P_{error} = P_{bit}\left(1 - Q_{bit}^{H+MAC}\right) Q_{bit}^{EM} + 0.5 Q_{bit}^{H+MAC}\left(1 - Q_{bit}^{EM}\right) + 0.5\left(1 + P_{bit}\right)\left(1 - Q_{bit}^{H+MAC}\right)\left(1 - Q_{bit}^{EM}\right). \tag{25}$$

B



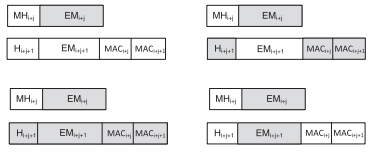
$$P_{error} = 2P_{bit}\left(1 - Q_{bit}^{MH}\right) Q_{bit}^{EM+MH} + Q_{bit}^{EM}\left(P_{bit}\left(1 - Q_{bit}^{MH}\right)\right)^2 + 0.5\left(\left(1 + P_{bit}\right)\left(1 - Q_{bit}^{MH}\right)\right)^2\left(1 - Q_{bit}^{EM}\right) 0.5 Q_{bit}^{2MH}\left(1 - Q_{bit}^{EM}\right) + Q_{bit}^{MH}\left(1 + P_{bit}\right)\left(1 - Q_{bit}^{MH}\right)\left(1 - Q_{bit}^{EM}\right). \tag{26}$$

C



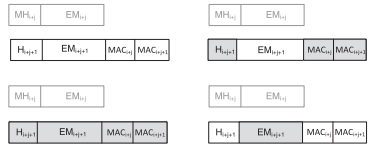
$$P_{error} = P_{bit}\left(1 - Q_{bit}^{MH}\right) Q_{bit}^{MH} + 0.5 Q_{bit}^{MH}\left(1 - Q_{bit}^{EM}\right) + 0.5\left(1 + P_{bit}\right)\left(1 - Q_{bit}^{MH}\right)\left(1 - Q_{bit}^{EM}\right). \tag{27}$$

D



$$P_{error} = 0.5\left(1 + P_{bit}\right)\left(1 - Q_{bit}^{2MAC+H}\right)\left(1 - Q_{bit}^{EM}\right) + 0.5 Q_{bit}^{2MAC+H}\left(1 - Q_{bit}^{EM}\right) + P_{bit}\left(1 - Q_{bit}^{2MAC+H}\right)\left(1 - Q_{bit}^{EM}\right). \tag{28}$$

E



$$P_{error} = P_{bit}\left(1 - Q_{bit}^{2MAC+H}\right) Q_{bit}^{EM} + 0.5\left(1 - Q_{bit}^{EM}\right) Q_{bit}^{H+2MAC} + 0.5\left(1 + P_{bit}\right)\left(1 - Q_{bit}^{H+2MAC}\right)\left(1 - Q_{bit}^{EM}\right). \tag{29}$$

Fig. 21. Various error event states and related error probability equations for C-Sec protocol.

$$P(A) = P(\textit{header is correct}) * P(\textit{EM is correct}) = Q_{bit}^{H} * Q_{bit}^{EM} \quad (17)$$

Similarly:

$$P(B) = (1 - Q_{bit}^{EM}) * Q_{bit}^{H} \quad (18)$$

$$P(C) = Q_{bit}^{EM} * (1 - Q_{bit}^{H}) \quad (19)$$

$$P(D) = (1 - Q_{bit}^{EM}) * (1 - Q_{bit}^{H}) \quad (20)$$

Using Eq. (17)–(20) the transition state diagram in Fig. 20 can be constructed. Its associated transition probability matrix that donates the probability of moving from state i–j and is given as follows:

$$P_{i,j} = \begin{pmatrix} P(A) & P(B) & P(C) & P(D) \\ P(A) & P(B) & P(C) & P(D) \\ P(A) & P(B) & P(C) & P(D) \\ P(A) & P(B) & P(C) & P(D) \end{pmatrix} \quad (21)$$

Then the mean probability of error P_{error} can be computed as:

$$P_{error} = P(A)e_1 + P(B)e_2 + P(C)e_3 + P(D)e_4 \quad (22)$$

where e_1, e_2, e_3, and e_4 are the error rates associated with each of the states in Fig. 18. The value of $e_1$is equal to 0 because the packet is error free at state 1; e_2 is equal to 0.5 because the decrypted body of the packet follows the SAC; and e_3 is equal to P_{bit}. Assuming that the bit errors events in the header and bit errors in the decrypted message are independent, the bit error rate in state 4 can be computed as:

$$e_4 = 1 - (1 - 0.5)(1 - P_{bit}) = 0.5(1 + P_{bit}) \quad (23)$$

Then the post decryption probability of error will be:

$$P_{error} = P_{bit}\left(1 - Q_{bit}^{H}\right)Q_{bit}^{EM} + 0.5Q_{bit}^{H}\left(1 - Q_{bit}^{EM}\right) + 0.5(1 + P_{bit})\left(1 - Q_{bit}^{H}\right)\left(1 - Q_{bit}^{EM}\right) \quad (24)$$

For C-Sec, the relations between packets introduce dependency in the error probability. To model this dependency, all possible packets format with different error events associated with them are considered. This includes five different pairings of dependency between packets with different formats. The error events associated with these cases are shown in Fig. 21. The shaded fields in the packets represent fields with errors. The mean probability of error P_{error} is computed for each case in a similar manner, taking into account the dependency in a stream of packets as follows:

A. The first packet is independent conventional, it follows the traditional post decryption error probability model.
B. The second packet has dependency on the header of the following packet.
C. The header of the packet $i + 1$ is already verified, it has dependency on the header of the packet $i + 2$.
D. The last compact packet depends on the header of the last conventional packet, which has different size.
E. The last conventional packet is similar to the first conventional packet, but with different header size.

To evaluate the performance of post-decryption error probability for the C-Sec, a weighted average of cases given by Eqs. (25)–(29) is computed assuming a percentage of compact packets that matches the percent of compact packets in the simulation results in Fig. 9 with a payload size of 30 bytes. The post-decryption error probability of the MiniSec protocol is given by Eq. (24). The MiniSec is used as a comparison because it has the smallest packet size among other protocols and the closest to C-Sec. Fig. 22 shows the plots of post-decryption error probability for both cases. The difference in output error probability is negligible and it does not exceed 1% in the active region.

Fig. 23 shows the percentages of packet loss, with 95% confidence, for C-Sec and MiniSec protocols, obtained from simulations using Castalia with varying noise floor (the

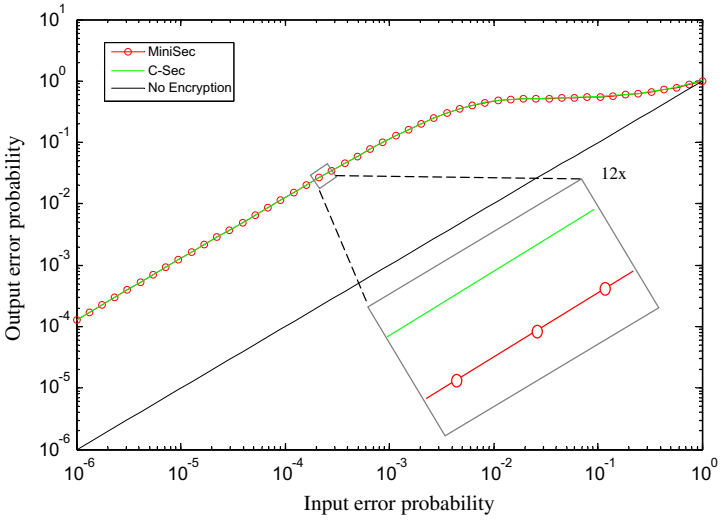


Fig. 22. Post decryption error probabilities for C-Sec and MiniSec.

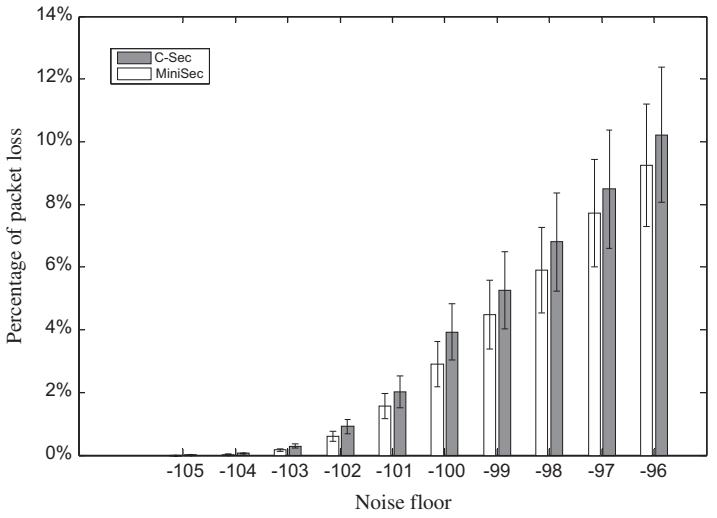


Fig. 23. Packet loss.

sum of all the noise sources and unwanted signals). The packet loss increases with the increase of noise floor. The C-Sec has about 1% more dropped packets than the Minisec in the active region. Compared to the high energy gain of the C-Sec, this amount of packet loss is can be probably be tolerated in many applications. As mentioned in the protocol description, the used of conventional mode can be enforced in case of high QoS constraints or the used of packet loss sensitive applications.

6. Conclusion and future work

The C-Sec protocol was presented in this paper; a link layer encryption protocol that provides all required security services, hides most of the packet headers making it more difficult to eavesdrop on the flow of wireless communication and minimizes the cost of defending against replay attacks. In addition to the above, it reduces energy consumption by minimizing security related communication overhead. A comparison between the C-Sec protocol with well-known protocols including the Zigbee security, SNEP, TinySec and MiniSec is also presented. The evaluation criteria included the security services provided, communication energy consumption, packet delay overhead, and the effect of the protocol on error probability. Simulation results with various payload sizes and packet loads using Castalia simulator shows that the C-Sec has significant advantage of in terms of communication energy consumption over other protocols. The C-Sec protocol renders significant energy saving that reaches up to 19.1% compared to the MiniSec protocol with a comparable packet size. We also prove analytically and by simulation that the relation between packets introduced by the C-Sec has limited effect on packet loss and end- to- end packet delay. This effect can be tolerated compared to the high-energy gain of the C-Sec.

In future work, we look forward to implement C-Sec protocol as part of securely designed wireless sensor platform that include hardware implemented encryption primitives like [13]. Based on that, we aim to conduct experimental testing to further prove the efficiency and performance of C-Sec as compared to other protocols.

Appendix A

To solve the Markov chain system in Fig. 12, let $X(t)$ be the state of the system at time t, and let Δt be a small time interval. Then:

$$P((X(t+\Delta t)=2)|(X(t)=1)) \cong \lambda_1 \Delta t \tag{30}$$

The event $X(t)=1$ is the union of the disjoint events A and B, where

A: The packet is being served.
B: The packet is waiting for the next arrival.

Then Eq. (30) will be:

$$P((X(t+\Delta t)=2)|(A\cup B)) = \frac{P((X(t+\Delta t)=2)\cap(A\cup B))}{P(A)+P(B)} = \frac{P((X(t+\Delta t)=2)\cap A)+P((X(t+\Delta t)=2)\cap B)}{P(A)+P(B)} \tag{31}$$

However, the first packet will not wait in the system if it finished processing and the next packet has arrived, therefore:

$$P((X(t+\Delta t)=2)\cap B)=0 \tag{32}$$

If the first packet is being processed, then it has to wait even if the next packet arrives, then:

$$P((X(t+\Delta t)=2)\cap A) = P((X(t+\Delta t)=2|A)\cdot P(A)) = \lambda P(A)\Delta t \tag{33}$$

Substituting (33) and (32), in (30):

$$P((X(t+\Delta t)=2)|(X(t)=1)) = \frac{\lambda P(A)\Delta t}{P(A)+P(B)} = \frac{\lambda P(A)/P(B)}{1+P(A)/P(B)} \tag{34}$$

However, there is a relation between $P(A)$ and $P(B)$ because:

$$P(Service\ at\ time\ t+\Delta t|Waiting\ at\ time\ t) \cong \lambda\Delta t$$
$$P(service\ at\ time\ t+\Delta t|Service\ at\ time\ t) \cong \mu\Delta t \quad (35)$$

Substituting ρ from the Markov-chain balance equation in Fig. 24 yields:

$$\lambda_1\Delta t = \lambda\Delta t\frac{\rho}{1+\rho}$$
$$\lambda_1 = \lambda\frac{\rho}{1+\rho} \quad (36)$$

Let p_1, p_2, p_3, ... be the probabilities of each of the states. The global balance equation: *Ratein = Rateout* for each node yields:

$$State1 \quad \lambda_1 P_1 = \mu P_2 \quad \rightarrow \quad P_2 = \frac{\rho^2}{1+\rho}P_1$$
$$State2: \quad \mu P_3 = \lambda p_2 \quad \rightarrow \quad P_3 = \rho P_2 \quad \rightarrow \quad P_3 = \frac{\rho^3}{1+\rho}P_1$$
$$\vdots$$
$$Statej-1: \quad \mu P_j = \lambda p_{j-1} \quad \rightarrow \quad P_j = \rho P_{j-1} \quad \rightarrow \quad P_j = \frac{\rho^j}{1+\rho}P_1$$

Now:

$$\sum_{j=1}^{\infty} p_j = 1 \quad (37)$$

Substituting the probabilities of the states into Eq. (36):

$$\rightarrow \quad p_1 + \frac{\rho^2}{1+\rho}p_1 + \frac{\rho^3}{1+\rho}p_1 + \frac{\rho^4}{1+\rho}p_1 + \cdots = 1 \quad (38)$$

From Eq. (38) p_1 can be computed as:

$$p_1 = (1-\rho)(1+\rho) \quad (39)$$

Combining (15) and (17):

$$p_j = \rho^j(1-\rho)j > 1 \quad (40)$$

From (39) and (40), the expected number of packet in the C-Sec system:

$$E_{C-Sec}(Number\ of\ packets\ in\ the\ system) = \sum_{n=1}^{\infty} np_n$$
$$= (1-\rho)(1+\rho) + \sum_{n=2}^{\infty} n\rho^n(1-\rho)$$
$$= (1+\rho) + \sum_{n=1}^{\infty} n\rho^n(1-\rho)$$

$$\lambda P(B) = \mu P(A)$$
$$\frac{P(A)}{P(B)} = \frac{\lambda}{\mu} = \rho$$

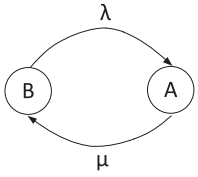


Fig. 24. Markov chain balance equation.

$$= (1-\rho) + \rho(1-\rho)\sum_{n=1}^{\infty} n\rho^{n-1}$$
$$= (1-\rho) + \rho(1-\rho)\frac{d}{d\rho}\sum_{n=0}^{\infty}\rho^n$$
$$= (1-\rho) + \rho(1-\rho)\left(\frac{1}{(1-\rho)^2}\right)$$
$$= 1-\rho+\frac{\rho}{1-\rho} \quad (41)$$

References

[1] Michael Allen, Lewis Girod, Elena Gaura, James Brusey, Geoffrey Challen, Wireless Sensor Networks: Deployments and Design Frameworks, Springer, 2010.

[2] Jennifer Yick, Biswanath Mukherjee, Dipak Ghosal, Wireless Sensor Network survey, Computer Networks 52 (12) (2008) 2292–2330.

[3] Ian Fuat Akyildiz, Mehmet Can Vuran, Wireless Sensor Networks, Wiley, Chichester, West Sussex, UK; Hoboken, NJ, 2010.

[4] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J.D. Tygar, SPINS: security protocols for sensor networks, in: Seventh Annual ACM International Conference on Mobile Computing and Networks (MobiCom 2001), Rome, Italy, July, 2001.

[5] C. Karlof, N. Sastry, D. Wagner, Tinysec: A link layer security architecture for Wireless Sensor Networks, in: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys 2004, Baltimore, MD, USA, November, 2004.

[6] M. Luk, G. Mezzour, A. Perrig, V. Gligor, MiniSec: a secure sensor network communication architecture, in: Proceedings of ACM and IEEE Conference on Information Processing in Sensor Networks (IPSN), Cambridge, Massachusetts, USA, April, 2007.

[7] ZigBee Alliance, ZigBee Specification, Technical report document 053474r17, 2008.

[8] Abidalrahman Moh'd, Nauman Aslam, William Robertson, William Phillips, C-Sec: energy efficient link layer encryption protocol for Wireless Sensor Networks, in: Consumer Communication and Networking Conference (IEEE-CCNC), Las Vegas, January, 2012.

[9] Y. Sankarasubramaniam, I.F. Akyildiz, S.W. McLaughlin, Energy efficiency based packet size optimization in Wireless Sensor Networks, in: The First IEEE International Workshop on Sensor Network Protocols and Applications, Anchorage, Alaska, USA, 2003.

[10] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, D. Culler, TinyOS: An Operating System for Sensor Networks, Ambient Intelligence, Springer, 2005.

[11] Sencun Zhu, Sanjeev Setia, Sushil Jajodia, LEAP+: efficient security mechanisms for large-scale distributed sensor networks, ACM Transactions on Sensor Networks 2 (4) (2006) 500–528.

[12] Hu Siquan, Mehul Motani, Early overhearing avoidance in Wireless Sensor Networks, NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet 4982 (2008) 26–35.

[13] Abidalrahman Moh'd, Nauman Aslam, William Phillips, William Robertson, Hosein Marzi, SN-SEC: a secure wireless sensor platform with hardware cryptographic primitives, Journal of Personal and Ubiquitous Computing (2012), http://dx.doi.org/10.1007/s00779-012-0563-9.

[14] Marcos A. Simplicio Jr., Bruno T. de Oliveira, Cintia B. Margi, Paulo S.L.M. Barreto, Tereza C.M.B. Carvalho, Mats Näslund, Survey and comparison of message authentication solutions on Wireless Sensor Networks, Ad Hoc Networks (2012).

[15] Paolo Santi, Janos Simon, Silence is golden with high probability:maintaining a connected backbone in Wireless Sensor Networks, Lecture Notes in Computer Science 2920 (2004) 106–121.

[16] Miguel A. Labrador, Pedro M. Wightman, Topology Control in Wireless Sensor Networks: With a Companion Simulation tool for Teaching and Research, Springer, New York, 2009.

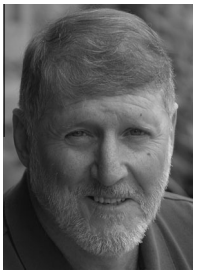
[17] James Nechvatal, Elaine Barker, Lawrence Bassham, William Burr, Morris Dworkin, James Foti, Edward Roback, Report on the Development of the Advanced Encryption Standard (AES), National Institute of Standards and Technology, October 2, 2000.

[18] M. Bellare, J. Kilian, P. Rogaway, The security of cipher block chaining, Advances in Cryptology, Crypto '94 Proceedings, Lecture Notes in Computer Science 839 (1994) 340–358.

[19] Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger, MD5 considered harmful today: creating a rogue CA certificate. December 30, 2008.
[20] X. Wang, Y.L. Yin, H. Yu, Finding collisions in the full SHA1, Crypto (2005).
[21] M. Sugita, M. Kawazoe, H. Imai, Grobner, Basis based cryptanalysis of SHA-1, Cryptology ePrint Archive, Report 2006/098, 2006.
[22] Abidalrahman Mohammad, H. Marzi, N. Aslam, L. Tawalbeh, Hardware implementation of secure hasing Functions on FPGAs for WSNs, in: Third International Conference on the Applications of Digital Information and Web Technologies (ICADIWT), Istanbul, Turkey, July 2010.
[23] S. Skorobogatov, Physical attacks on tamper resistance: progress and lessons, in: 2nd Army Research Office Workshop on Hardware Assurance, Washington, DC, USA, April 2012.
[24] Castalia Simulator. <http://castalia.npc.nicta.com.au/>.
[25] Omnetpp. <http://www.omnetpp.org/>.
[26] G. Bolch, S. Greiner, H. de Meer, K.S. Trivedi, Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications, Wiley-Interscience, 2006.
[27] Holger Karl, Andreas Willig, Protocols and Architectures for Wireless Sensor Networks, John Wiley & Sons, New York, 2007.
[28] J. Reason, End-to-End Confidentiality for Continuous-Media Applications in Wireless Systems, Doctoral Dissertation, UC Berkeley, 2000.
[29] F. Sattar, M. Mufti, On modeling post decryption error processes in UMTS air interface, in: Inscrypt 2007, LNCS 4990, 2007.
[30] Qiuhua Yang, Jian Wang, Shuangqing Wei, and Jian Yuan, Statistical analysis of error patterns in block ciphered crypto-system, in: International ICST Conference on Communications and Networking in China (CHINACOM), Harbin, China, 2011.
[31] R. Forre, The strict avalanche criterion: spectral properties of Booleans functions and an extended definition, Advances in cryptology, Crypto'88, Lecture Notes in Computer Science 403 (1990) 450–468.
[32] W.Y. Zibideh, M.M. Matalgah, Modified-DES encryption algorithm with improved BER performance in wireless communication, in: Proceedings of the 2011 IEEE Radio and Wireless Symposium (RWS 2011) Part of the Radio Wireless Week (RWW 2011), Phoenix, AZ, USA, January 2011.
[33] Mohamed A. Haleem, Chetan Nanjunda Mathur, Rajarathnam Chandramouli, K.P. Subbalakshmi, Opportunistic encryption: a trade-off between security and throughput in wireless networks, IEEE Transactions on Dependable and Secure Computing 4 (4) (2007) 313–324.

Abidalrahman Mohammad is currently a Ph.D candidate in Engineering Mathematics and Internetworking Department at Dalhousie University, Canada, under the supervision of Dr. William J. Phillips and Dr. Nauman Aslam. His main research concern is to develop energy efficient security protocols and cryptographic algorithms for energy efficient and low-power architectures such as wireless sensor networks. He received both his B.Sc. and M.Sc. Degrees in Computer Engineering from Jordan University of Science and Technology in February, 2006 and July, 2007 respectively.

Nauman received his Ph.D in Engineering Mathematics from Dalhousie University, Halifax, Nova Scotia, Canada in 2008. He was awarded Master of Engineering Degree in Internetworking from Dalhousie University in 2003; and B.Sc. in Mechanical Engineering from University of Engineering and Technology, Lahore, Pakistan in 1993. Prior to joining Northumbria University he worked as an Assistant Professor at Dalhousie University, Canada from 2008 - 2011. Currently, he also holds an adjunct assistant professor position at Dalhousie University. Dr. Nauman has extensive research experience in wireless ad hoc and sensor networks. Dr. Nauman has published over 30 refereed research articles in international journals and conference proceedings. He has also co-organized international workshops and conference. He also served on many technical program committees and reviewed papers for several journals. Dr. Nauman is a member of IEEE and IAENG.



William J. Phillips received the B.Sc. degree in Engineering Mathematics, the M.Sc. degree in Mathematics from Queen's University at Kingston and the Ph.D in Mathematics from the University of British Columbia. Dr. Phillips held visiting positions at the University of Guelph, Queen's University, Dalhousie University, and Saint Mary's University before joining the Department of Applied Mathematics at Technical University of Nova Scotia. He is currently Professor and Head of the Department of Engineering Mathematics at Dalhousie University (merged with the Technical University of Nova Scotia in 1997). Dr. Phillips research program is aimed at algorithms and implementations for communication networks.

William Robertson received his B.Sc. degree in Electrical Engineering, and his M.Sc. degree in Electronics Engineering from Aberdeen University, Scotland, and his Ph.D degree in Electrical Engineering from the Technical University of Nova Scotia in 1986. He worked in the Department of Telecommunications, Pretoria, and then at Cape Town University and Groote Schuur Hospital on computer software and signal processing before joining Stellenbosch University, Department of Electrical & Electronic Engineering. He has been working in Technical University of Nova Scotia (merged with the Dalhousie University in 1997) since 1980. Dr. Robertson held various academic positions and he is currently Director of the Master of Engineering in Internetworking Program. His research interests are in signal processing, network communications, quality of service, and wireless sensor networks.