SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2014; **7**:2654–2668
Published online 18 January 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.406

SPECIAL ISSUE PAPER

Secure communication for the Internet of Things—a comparison of link-layer security and IPsec for 6LoWPAN

Shahid Raza[1]*, Simon Duquennoy[1], Joel Höglund[1], Utz Roedig[2] and Thiemo Voigt[1]

[1] Swedish Institute of Computer Science, 16440 Kista, Sweden
[2] Lancaster University, School of Computing and Communications, Lancaster, U.K.

ABSTRACT

The future Internet is an IPv6 network interconnecting traditional computers and a large number of smart objects. This Internet of Things (IoT) will be the foundation of many services and our daily life will depend on its availability and reliable operation. Therefore, among many other issues, the challenge of implementing secure communication in the IoT must be addressed. In the traditional Internet, IPsec is the established and tested way of securing networks. It is therefore reasonable to explore the option of using IPsec as a security mechanism for the IoT. Smart objects are generally added to the Internet using IPv6 over Low-power Wireless Personal Area Networks (6LoWPAN), which defines IP communication for resource-constrained networks. Thus, to provide security for the IoT based on the trusted and tested IPsec mechanism, it is necessary to define an IPsec extension of 6LoWPAN. In this paper, we present such a 6LoWPAN/IPsec extension and show the viability of this approach. We describe our 6LoWPAN/IPsec implementation, which we evaluate and compare with our implementation of IEEE 802.15.4 link-layer security. We also show that it is possible to reuse crypto hardware within existing IEEE 802.15.4 transceivers for 6LoWPAN/IPsec. The evaluation results show that IPsec is a feasible option for securing the IoT in terms of packet size, energy consumption, memory usage, and processing time. Furthermore, we demonstrate that in contrast to common belief, IPsec scales better than link-layer security as the data size and the number of hops grow, resulting in time and energy savings. Copyright © 2012 John Wiley & Sons, Ltd.

KEYWORDS
Security; Internet of Things; 6LoWPAN; IPsec; IEEE 802.15.4 Security

***Correspondence**
Shahid Raza, Swedish Institute of Computer Science, 16440 Kista, Sweden.
E-mail: shahid@sics.se

1. INTRODUCTION

The future Internet is an IPv6 network interconnecting traditional computers and a large number of smart objects [1]. Smart objects, often referred to as *things*, usually feature small embedded computers with communication, sensing and actuation capabilities. The resulting Internet of Things (IoT) will be the foundation for many services and will enable communication between traditional computers and smart objects on a global scale. It is therefore important to address the traditional security requirements, that is, authentication, integrity, nonrepudiation and confidentiality in the context of the IoT.

Smart objects are commonly interconnected using a wireless IEEE 802.15.4 [2] (referred to as 802.15.4 in the remaining paper) network. A border router can be used to connect an 802.15.4 network to the Internet to enable IPv6 communication between smart objects and Internet hosts. However, IPv6 packets traveling on 802.15.4 networks use compressed header formats as defined by IPv6 Over Low-power Wireless Personal Area Networks (6LoWPAN) [3] to conserve scarce bandwidth resources. The border router has to compress/decompress IP packet headers when forwarding packets to ensure compatibility with the existing Internet.

At present, 6LoWPAN relies on 802.15.4 security mechanisms. A single key is used in the network to secure data transfer on a hop-by-hop basis. This prevents, as long as the key is kept secure, unauthorized access to the 802.15.4 network. In situations where the 802.15.4 network is isolated and without connection to the Internet, such a mechanism may be considered adequate. However, in the context of the IoT such an approach fails to provide end-to-end (E2E) security in terms of authentication, integrity, nonrepudiation, and confidentiality. Clearly, additional or alternative mechanisms are required.

The established and tested method to implement generic E2E security in the Internet is IPsec [4]. IPsec defines security extensions to the IP protocol for the implementation of security services. Thus, it seems reasonable to explore the option of using IPsec in the context of 6LoWPAN networks. In this paper, we present a 6LoWPAN/IPsec extension and show the viability of this approach.

Our 6LoWPAN/IPsec extension to implement true E2E secure communication between smart objects and Internet hosts is illustrated in Figure 1. We define header compression for the IPsec-related IPv6 Extension Headers: Authentication Header (AH) and Encapsulating Security Payload (ESP). We present an implementation and evaluation of our 6LoWPAN/IPsec extension for smart objects running the well-known Contiki operating system [5]. This implementation exploits the cryptographic functionality provided by standard 802.15.4 transceivers. The hardware support is intended to be used for 802.15.4 link-layer security, but we show that it is possible to reuse this hardware for 6LoWPAN/IPsec.

A particular focus of this paper is the comparison of 6LoWPAN/IPsec security with traditional 802.15.4 link-layer security. For this purpose, we also implement 802.15.4 link-layer security for Contiki, which allows us to compare both security mechanisms in a testbed. Our experiments show that traditional 802.15.4 link-layer security does not provide significantly better network performance than our proposed 6LoWPAN/IPsec security. This is particularly true when the crypto hardware support of 802.15.4 transceivers is used for 6LoWPAN/IPsec security. Furthermore, as the size of the secured data increases and as the data is being carried over more hops, the E2E nature of IPsec leads to better performance than hop-by-hop link-layer security.

The core contributions of this paper are as follows:

- We present a definition of a 6LoWPAN extension for IPsec, supporting both AH and ESP.
- We present an implementation and a thorough testbed performance evaluation of the 6LoWPAN/IPsec extension. We show the performance gains obtained from the usage of cryptographic hardware support of 802.15.4 transceivers.
- We experimentally show that 6LoWPAN/IPsec outperforms 802.15.4 link-layer security as the payload size and/or the number of hops increase.

The next section of this paper discusses the state-of-the-art in security for the IoT. Section 3 gives an overview of 6LoWPAN, 802.15.4 link-layer security, and IPsec. Section 4 describes our 6LoWPAN/IPsec extension. Section 5 details our 6LoWPAN/IPsec and 802.15.4 link-layer security implementation for the Contiki OS. Section 6 presents our performance evaluation of the 6LoWPAN/IPsec extension and its comparison with the 802.15.4 security mechanisms. Section 7 concludes the paper.

2. RELATED WORK

We present research work related to security in the IoT. After reviewing works aiming at designing cryptographic algorithms for constrained devices, we discuss the different layers at which security can take place in IP-based IoT.

2.1. Embedding cryptographic algorithms

Much research work has focused on reducing complexity of cryptographic algorithms or on improving efficiency of key distribution protocols. For example, TinyECC [6] and NanoECC [7] provide elliptic curve cryptography in order to make cryptography feasible on resource-constrained devices. Wood and Stankovic [8] and Hu *et al.* [9] demonstrated efficient cryptography for smart objects using dedicated crypto hardware support. For example, Liu and Ning [10] and Chung and Roedig [11] described key distribution mechanisms that save scarce bandwidth in resource-constrained networks.

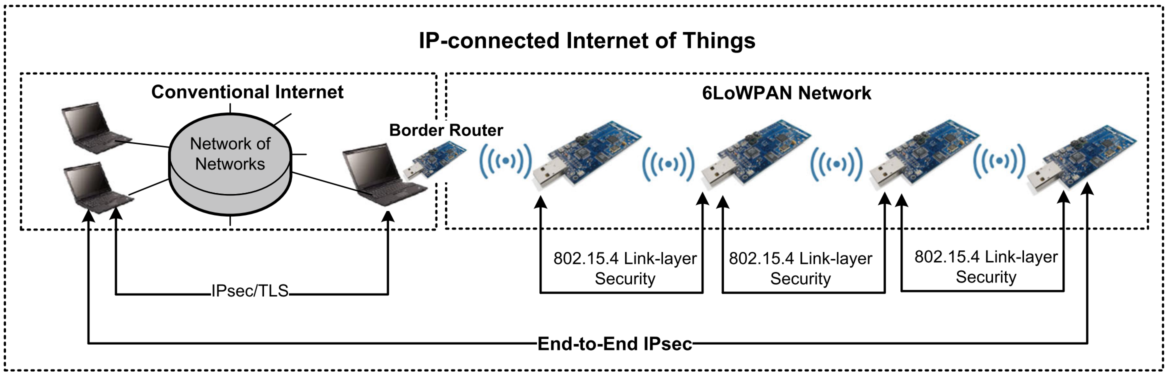

IP-connected Internet of Things
Conventional Internet
Network of Networks
Border Router
6LoWPAN Network
802.15.4 Link-layer Security
802.15.4 Link-layer Security
802.15.4 Link-layer Security
IPsec/TLS
End-to-End IPsec

Figure 1. IEEE 802.15.4 security can secure communication within IPv6 over low-power wireless personal area networks (6LoWPAN) devices with a shared network key. IPsec/transport-layer security (TLS) can be used to secure the communication between the border router and an Internet host. We rather propose to use IPsec in an end-to-end manner, that is, between 6LoWPAN devices and IPv6 Internet hosts.

2.2. Securing the IoT at the link layer

IP communication among smart objects uses 6LoWPAN [3], which in turn builds on the IEEE 802.15.4 [2] link layer. IEEE 802.15.4 link-layer security is the current state-of-the-art security solution for IP-connected IoT. IEEE 802.15.4 defines data encryption and integrity verification. Benefits are network protocol independence and hardware support for the cryptographic functions by currently used 802.15.4 radio chips. Link-layer security provides *hop-by-hop* security where every node in the communication path (including the 6LoWPAN border router; see Figure 1) has to be trusted and where neither host authentication nor key management is supported. A single pre-shared key is used to protect all communication. Furthermore, messages leaving the 802.15.4 network and continuing to travel on an IP network are not protected by link-layer security mechanisms. Therefore, in many solutions, a separate security mechanism is added to protect data traveling between Internet hosts and border routers. One such example is the ArchRock PhyNET [12] that applies IPsec in tunnel mode between the border router and Internet hosts. HIP DEX [13] is another solution that can be used directly as a keying mechanism for a medium access control layer security protocol. Recently, Roman *et al.* [14] proposed key management systems for sensor network that are applicable to link-layer security.

Because every node in the path has to be trusted, E2E security in the IoT cannot be achieved using the outlined approach.

2.3. Securing the IoT at the transport layer

End-to-end security can be provided by using transport-layer security (TLS) [15] or its predecessor Secure Sockets Layer (SSL). TLS and SSL are widely used in the Internet to secure communication between hosts. TLS and SSL include key exchange mechanisms and provide authentication between Internet hosts in addition to confidentiality and integrity. There are some drawbacks that make it difficult to use these protocols for securing the IoT. TLS can only be used over TCP, which is not the preferred method of communication for smart objects as TCP connection setup consumes scarce resources. Furthermore, the TLS/SSL session setup and key exchange require a number of message exchanges.

Nevertheless, SSL has been proposed as security mechanism for the IoT by Hong *et al*. [16]. Their evaluation shows that this security mechanism is indeed quite costly as a full SSL handshake and a data packet transfer requires 2 s to complete. Foulagar *et al.* [17] propose a TLS implementation for smart objects. However, this solution involves the border router to reduce computational effort on smart objects and cannot be considered a full E2E solution. The User Datagram Protocol (UDP) version of TLS named DTLS could be used in 6LoWPAN networks. However, the 6LoWPAN specifications neither provide compression for DTLS nor hooks in the specifications that can be used to compress DTLS.

2.4. Securing the IoT at the network layer

The IPsec [4] protocol suite, mandated by IPv6, provides E2E security for any IP communication. Like TLS and unlike link-layer solutions, it includes a key exchange mechanism and provides authentication in addition to confidentiality and integrity. By operating at the network layer, it can be used with any transport protocol, including potential future ones. Furthermore, it ensures the confidentiality and integrity of transport-layer headers and integrity of IP headers, which cannot be carried out with higher-level solutions as TLS. For these reasons, the research community [18–20] and 6LoWPAN standardizations groups [3] consider IPsec a potential security solution for the IoT. On the other hand, some have regarded it as a too heavy-weight option [21].

Granjal *et al.* [22] discussed the use of IPsec for 6LoWPAN. However, exact specifications of the required 6LoWPAN headers are not given. Furthermore, no implementation is provided and no detailed evaluation of possible communication performance is given. In their study, they analyze the execution times and memory requirements of cryptographic algorithms that they proposed for a 6LoWPAN/IPsec integration.

In our previous work, we have presented a 6LoWPAN/IPsec solution and have made a preliminary performance analysis of the overall system [23]. In this paper, we extend our previous work in several aspects. First, in this paper, we describe ESP for 6LoWPAN/IPsec, whereas in our previous work, we only discussed in detail the AH. Second, we compare the 6LoWPAN/IPsec solution with the commonly employed 802.15.4 link-layer security. Third, we present a thorough testbed performance evaluation of the 6LoWPAN/IPsec solution and 802.15.4 security.

3. BACKGROUND

In this section, we give an overview of technologies relevant for the work presented in this paper. We give background information on IPv6 [24] and 6LoWPAN [3], IPsec [4], and on 802.15.4 [2] security. More detailed information can be found in the corresponding requests for comments (RFCs) and standard documents.

3.1. Overview of 6LoWPAN

IPv6 over Low-power Wireless Personal Area Network [3] is used to tightly interconnect existing Internet and smart objects by specifying how IPv6 datagrams are to be transmitted over an IEEE 802.15.4 network. 6LoWPAN acts as a layer between the IP layer and the link layer that compresses IP headers and performs fragmentation when necessary. The maximum transmission unit (MTU) of 802.15.4 is 127 bytes. If 802.15.4 security is enabled, the maximum payload is reduced to 81 bytes, of which 40 would be consumed with uncompressed IPv6 headers. By compressing IPv6 header, 6LoWPAN increases the

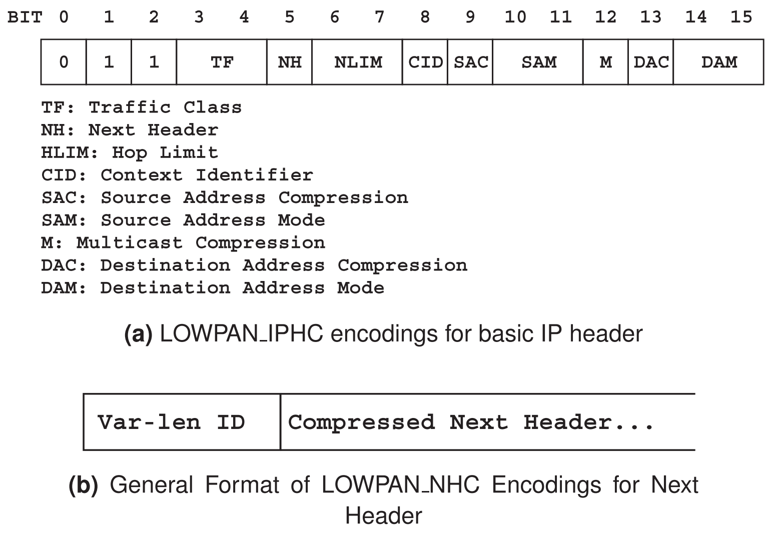

BIT 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 1 TF NH NLIM CID SAC SAM M DAC DAM
TF: Traffic Class
NH: Next Header
HLIM: Hop Limit
CID: Context Identifier
SAC: Source Address Compression
SAM: Source Address Mode
M: Multicast Compression
DAC: Destination Address Compression
DAM: Destination Address Mode
(a) LOWPAN_IPHC encodings for basic IP header
Var-len ID
Compressed Next Header...
(b) General Format of LOWPAN_NHC Encodings for Next Header

Figure 2. IPv6 over low-power wireless personal area networks (6LoWPAN) context-aware compression mechanisms.

payload carried in 802.15.4 frames. When data cannot fit in a single frame, 6LoWPAN performs fragmentation. All nodes in a 6LoWPAN network perform compression/decompression and fragmentation/reassembly. One specific node referred to as *border router* acts as a gateway between the 6LoWPAN and the conventional Internet.

The 6LoWPAN compression mechanisms [25] define header compression using LOWPAN_IPHC for IP header compression and LOWPAN_NHC for the next header compression. The IPHC* reduces the IP header length to 2 bytes for a single-hop network and 7 bytes in a multihop case. The IPHC header is shown in Figure 2(a). The next header field when set to 1 indicates that the next header following the compressed IPv6 header is encoded with NHC. The general format of NHC is shown in Figure 2(b). NHC has a length of 1 or more octets, where the first variable length bits identify the next header type and the remaining bits are used to encode header information. Currently, 6LoWPAN defines NHC for the IP extension header and the UDP header.

3.2. Overview of IEEE 802.15.4 security

Currently, 6LoWPAN relies on 802.15.4 [2] security to protect the communication between neighboring nodes. The standard supports access control, message integrity, confidentiality, and replay protection. Message integrity is achieved by including a message authentication code (MAC) in packets. If the receiver cannot verify the MAC, the packet will be discarded. Confidentiality is provided by applying symmetric cryptography to outgoing packets. Through the inclusion of a monotonically increasing counter in messages, nodes can discard packets being resent by malicious nodes, achieving replay protection.

Figure 3 shows the structure of an 802.15.4 packet with optional security headers. The packet overhead with link-layer security varies between 4 and 21 bytes depending on the scheme used.

The security modes supported by the 802.15.4 standard include advance encryption standard in counter mode (AES-CTR) for encryption only, AES in cipher block chaining mode (AES-CBC) for message authentication only, and AES in counter with CBC-MAC mode (AES-CCM), which combines encryption and message authentication. For the MAC modes, the included authentication code is 4, 8, or 16 bytes. Besides the null mode (security features turned off), AES-CCM is the only mode mandated by the standard, which must be available on all standard-compliant devices. It has been pointed out that the security suite with encryption, AES-CTR, should not be used on its own. Networks with only encryption and no authentication are open to insertion of false packets and have been shown vulnerable [26].

The IEEE 802.15.4 standard currently uses pre-shared keys for encryption and integrity verification.

3.3. Overview of IPsec

IPv6 with potentially unlimited address space of 2^{128} addresses makes it possible to assign a unique address to each physical device on earth. Besides increased address space, IPv6, as compared to IPv4, also provides a simplified header format, better support for extensions and mandates IP security.

IPv6 uses IPsec [4] to secure IP communication between two end points. IPsec is a collection of protocols, which includes AH [27] that provides authentication services, ESP [28] that provides both authentication and privacy services, and a set of encryption and authentication

*In the rest of this article LOWPAN_IPHC is referred to as IPHC and LOWPAN_NHC is referred to as NHC.

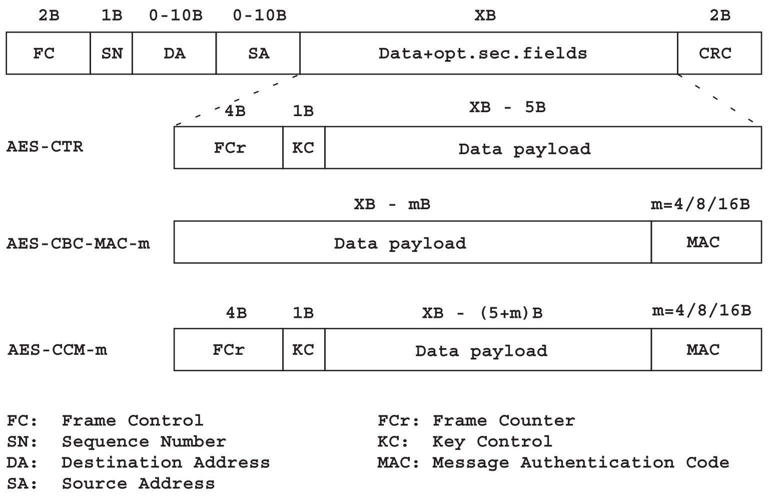


Figure 3. IEEE 802.15.4 frame with security headers.

algorithms [29]. A node keeps track of the so-called security associations (SA) that specify how a particular IP flow should be treated in terms of security.

Authentication Header provides connectionless integrity, data origin authentication for IP datagrams, and protection against replay attacks. AH uses a keyed message integrity code (MIC) to protect the complete IP packet including IP header, AH and IP payload. The IP header fields that are mutable while the packet is in transit are set to zero while calculating the MIC. AH includes a reference to the next header (for example, ESP, TCP, or UDP), a length field, the security parameters index (SPI) that identifies the SA used, a sequence number to prevent replay attacks, and the integrity check value (ICV) that is a MIC. The ICV must be an integral multiple of 32 bits for IPv6. For MIC calculation, all IPsec-enabled IPv6 hosts support at least AES-XCBC-MAC-96 and HMAC-SHA1-96 [29] that have sizes of 12 bytes each. A basic AH has a total size of 24 bytes.

Encapsulating Security Payload provides origin authenticity, data integrity, and confidentiality protection of IP datagram. In contrast to AH, ESP operates on the IP payload but not on the header. ESP has common fields with AH and also contains the encrypted payload and padding required by block ciphers. ESP only encrypts payload data, padding, pad length, and the next header; the ICV calculation, if selected, includes all header fields in the ESP. If we consider AES-CTR as encryption algorithm, ESP, with perfect block alignment, will have an overhead of 18 bytes (10 bytes for ESP and 8 bytes for initialization vector). If additional authentication using AES-XCBC-MAC-96 is used, the ESP size grows to 30 bytes, as the minimum length of AES-XCBC-MAC-96 is 12 bytes.

Both AH and ESP support two different modes: transport mode and tunnel mode. In transport mode, the IP header and payload are directly secured as previously described. In tunnel mode, a new IP header is placed in front of the original IP packet, and security functions are applied to the encapsulated (tunneled) IP packet. In the context of 6LoWPAN, tunnel mode would be very inefficient, as the additional headers further increase the packet size.

An IPsec SA can be established using protocols such as Kerberized Internet Negotiation of Keys [30], Internet Key Exchange (IKE) [31], or a DNS-based solution [32]. IPsec requires that nodes support authentication based on either certificates or pre-shared keys [4]. Thus, the usage of pre-shared keys in the context of IPsec is possible.

4. 6LoWPAN/IPsec EXTENSION

The 6LoWPAN [25] standard specifies compression schemes for IP and UDP, but not for IPsec AH and ESP. In this section, we address this shortcoming and provide an appropriate 6LoWPAN/IPsec extension. In our previous work [23], we have specified how IPsec AH can be compressed and connected to the 6LoWPAN compression system. We extend that solution and provide a specification for ESP and improved methods to compress IPsec headers.

4.1. LOWPAN_NHC extension header encoding

As discussed in the background section, the 6LoWPAN draft defines the general format of NHC that can be used to encode IP next header. We define NHC encodings for the two IP extension headers namely AH and ESP. 6LoWPAN already defines NHC encodings for IP extension headers (NHC_EH) that can be used to link AH and ESP extension headers. NHC encodings for the IPv6 Extension Headers consist of an NHC octet where three bits (bits 4, 5, and 6) are used to encode the IPv6 Extension Header ID (EID). The NHC_EH encoding for extension headers is shown in Figure 4.

Out of eight possible values for the EID, six are assigned by the HC15 [25] specification. The remaining

BIT	0	1	2	3	4	5	6	7
	1	1	1	0		EID		NH

```
EID: Extension Header
     ID (EID)
NH:  Next Header
```

Figure 4. LOWPAN_NHC_EH: next header compression encoding for IPv6 extension header.

two slots (101 and 110) are currently reserved. As AH and ESP are IP extension headers, it makes sense to use one of these reserved slots for AH and ESP compression. We propose to use one of the reserved slots, say 101, to identify that the next header is an AH or ESP header. The ID bits in the proposed NHC for AH and ESP identify that the current header is AH or ESP, see Sections 4.2 and 4.3. It is also necessary to set the last bit in NHC_EH to 1 to specify that the next header is NHC encoded.

4.2. LOWPAN_NHC_AH encoding

Figure 5 describes our NHC encoding for AH. Next, we describe the role of all fields:

- The first 4 bits in the NHC_AH represent the NHC ID that we define for AH. These are set to 1101.
- If $PL = 0$, the payload length (length of the IPsec header) field in AH is omitted. This length can be obtained from the SPI value because the length of the authenticating data depends on the algorithm used and are fixed for any input size.

  If $PL = 1$, the payload value is carried inline after the NHC_AH header.

- If $SPI = 0$, the default SPI for the 802.15.4 network is used and the SPI field is omitted. We set the default SPI value to 1. This does not mean that all nodes use the same SA but that every node has a single preferred SA, identified by SPI 1.

  If $SPI = 1$, all 32 bits indicating the SPI are carried inline.

- If $SN = 0$, first 16 bits of sequence number are elided. The remaining bits are carried inline.

  If $SN = 1$, all 32 bits of the sequence number are carried inline.

BIT	0	1	2	3	4	5	6	7
	1	1	0	1	PL	SPI	SN	NH

```
PL:  Payload Length
SPI: Security Parameter Index
SN:  Sequence Number
NH:  Next Header
```

Figure 5. LOWPAN_NHC_AH: next header compression encodings for IPv6 AH.

- If $NH = 0$, the next header field in AH will be used to specify the next header and it is carried inline.

  If $NH = 1$, the next header field in AH is elided. The next header is encoded using NHC.

Note that even when used in 6LoWPAN, AH calculates the MIC on the uncompressed IP header, thus allowing authenticated communication with Internet hosts. The minimum length of a standard AH, supporting the mandatory HMAC-SHA1-96 and AES-XCBC-MAC-96, consists of 12 bytes of header fields plus 12 bytes of ICV. After optimal compression we obtain a header size of 4 bytes plus 12 bytes of ICV. Figure 6 shows a compressed IPv6/UDP packet secured with AH using HMAC-SHA1-96.

4.3. LOWPAN_NHC_ESP encoding

Figure 7 shows the NHC encodings that we propose for ESP. Next, we describe the function of each header field:

- The first 4 bits in the NHC_ESP represent the NHC ID that we define for ESP. These are set to 1110.
- The next bit is unused. We leave this field empty to achieve coding similarity between AH and ESP (ESP does not have a payload length field). However, this field could be used to increase SPI coding to 2 bits if required.
- If $SPI = 0$, the default SPI for the 802.15.4 network is used and the SPI field is omitted. We set the default SPI value to 1. This does not mean that all nodes use the same SA but that every node has a single preferred SA, identified by SPI 1.

  If $SPI = 1$, all 32 bits indicating the SPI are carried inline.

- If $SN = 0$, first 16 bits of sequence number are used. The remaining 16 bits are assumed to be zero.

  If $SN = 1$, all 32 bits of the sequence number are carried inline.

- If $NH = 0$, the next header field in ESP will be used to specify the next header and it is carried inline.

If $NH = 1$, the next header will be encoded using NHC. In case of ESP, we cannot skip the next header unless the end hosts are able to execute 6LoWPAN compression/decompression and encryption/decryption jointly. The nodes in the 6LoWPAN network make their decision about the next header on the basis of the next header value, not the actual header that is carried inline.

Recall that the minimum ESP overhead without authentication, AES-CBC and perfect block alignment is 18 bytes. After optimal compression, this header overhead is reduced to 14 bytes. ESP with authentication contains additional 12 bytes of ICV. Figure 8 shows an UDP/IP packet secured with compressed ESP. The shaded portion represents cipher-text.

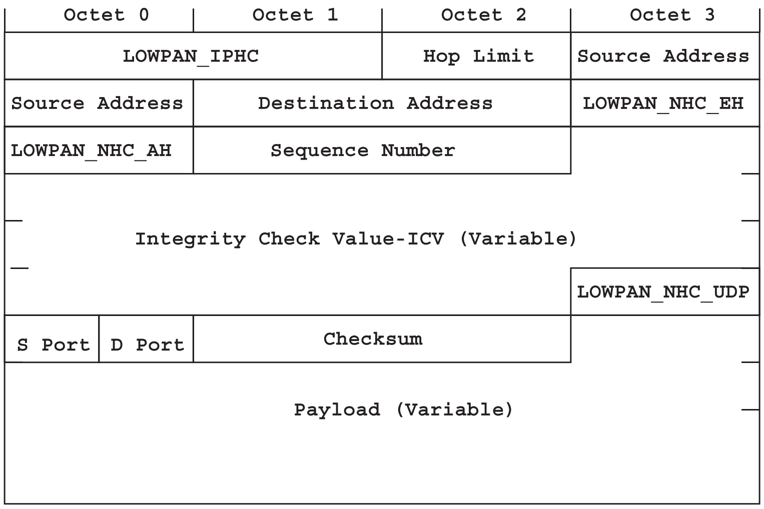


Figure 6. A compressed and authentication header secured IPv6/User Datagram Protocol packet.

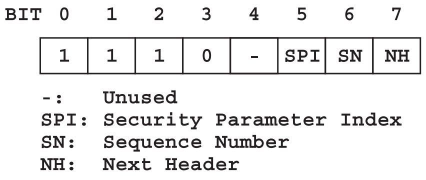


Figure 7. LOWPAN_NHC_ESP: next header compression encoding for IPv6 Encapsulating Security Payload.

When ESP is used, the UDP header is encrypted and therefore cannot be compressed. One solution to enable UDP header compression when ESP is used is to specify a new encryption algorithm for ESP, which is able to perform 6LoWPAN UDP header compression plus encryption at the source and destination. As such a solution would not be viable until massive adoption of 6LoWPAN, we do not specify its details. In the rest of the paper, we focus on the case were the UDP header remain uncompressed.

5. IMPLEMENTATION

We implement 802.15.4 link-layer security and the 6LoWPAN/IPsec extension described in the previous section for the Contiki operating system [5]. The implementation of both security mechanisms makes use of the hardware security functions provided by the CC2420 radio chip available on many platforms. Our implementations are for the Tmote Sky platform that features a CC2420 radio chip [33].

5.1. Link-layer security implementation

Our IEEE 802.15.4 link-layer security implementation supports header construction for all security modes described in the standard. The construction of the 802.15.4 frame is performed in software, whereas the cryptographic operations are performed by theCC2420 radio chip. Manual key distribution is used as key management is not standardized in 802.15.4. Hence, one pre-defined key is used on all nodes in the network.

IEEE 802.15.4 link-layer security is applied after a message is compressed and fragmented at the 6LoWPAN layer. It is important to note that when fragmentation is necessary and link-layer security is enabled, the security header and/or MIC are added to each fragment. Thus, as the payload increases, so does the overhead of link-layer security headers.

5.2. IPsec implementation

Contiki includes a well-tested and widely used IPv6 stack, named μIP. Contiki also has an implementation for 6LoWPAN compression and fragmentation mechanisms called SICSLoWPAN. We extend μIP to support IPsec and extend SICSLoWPAN to support our AH and ESP compression schemes.

We use the SHA1 and AES implementations from MIRACL (Shamus Software, Dublin, Ireland) [34], an open source library. For AH and ESP, we implement all cryptographic modes of operation needed for authentication and encryption. For AH, we implement the mandatory HMAC-SHA1-96 and AES-XCBC-MAC-96. For ESP, we implement the mandatory AES-CBC for encryption and HMAC-SHA1-96 for authentication. Additionally, in ESP, we implement the optional AES-CTR for encryption and AES-XCBC-MAC-96 for authentication.

We propose an alternate implementation of AES-based modes of operation leveraging the cryptographic capabilities of the CC2420 radio chip. Although the mode of operations are still implemented in software, they rely on the CC2420 to perform 128-bit AES block encryption/decryption. We use this hardware-aided operation for both authentication and encryption in IPsec.

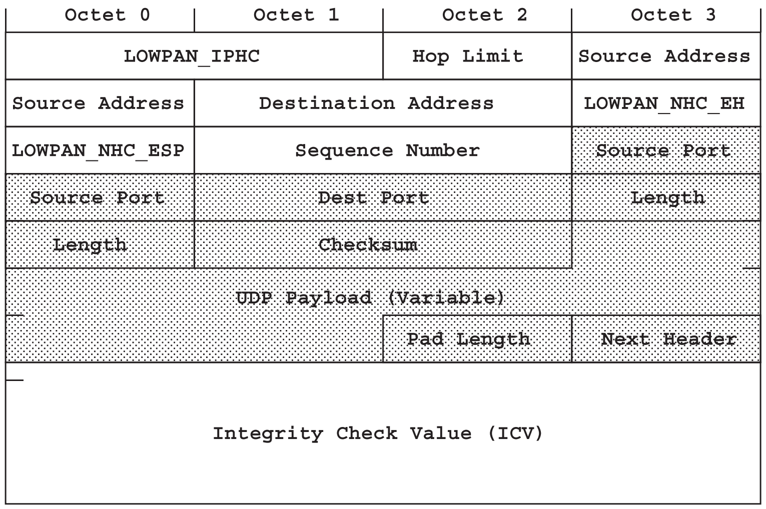


Figure 8. A compressed and Encapsulating Security Payload secured IPv6/User Datagram Protocol packet.

Our Contiki 6LoWPAN/IPsec implementation uses pre-shared keys to establish SAs, which is a standard-compliant way to set keys. In future work, we plan to add support for automatic key distribution, for instance with the IKE protocol [31] with certificates. This would provide dynamic key allocation to various IP hosts and would allow periodic key renewal. Also, current research in wireless sensor network adds database support for sensor networks [35], which can be used to create IPsec/IKE databases such as security policy database, security association database, and so on [4].

5.3. Concurrent use

Security at network-layer security and that at link-layer security are not interchangeable. IPsec is used to implement true E2E security. Link-layer security is often used for controlling access to the wireless medium. Link-layer security can also be used when we need to encrypt the IP headers as IPsec only encrypts the IP payload. The two previously outlined security implementations can be executed concurrently. When IPsec is used in ESP mode, the data is already encrypted and authenticated at the network layer and it does not add more security to re-encrypt the data at the link layer. However, it is important to detect the data modification attack as early as possible in the wireless medium. With IPsec, the integrity of the data is verified at the end nodes as IPsec is E2E. For this reason, link-layer security with authentication service is important even though we already use IPsec at the network layer. In this case, full CCM-128 at the link-layer may be unnecessary and CBC-128 is sufficient.

6. EVALUATION AND RESULTS

In this section, we evaluate our 6LoWPAN/IPsec extensions and compare it with 802.15.4 link-layer security. After describing our experimental setup, we compare link-layer security to IPsec in terms of memory footprint, packet size, energy consumption, and communication performance in a variety of network settings.

6.1. Experimental setup

Our experimental setup shown in Figure 1 consists of four Tmote Sky [33] nodes, one Tmote Sky acting as a 6LoWPAN border router and an Internet host running Ubuntu OS. The Internet host runs the IPsec stack of the Linux kernel (version 2.6.38). The four smart objects (nodes) on the right side in Figure 1 form a multihop network. Using this setup, we test different security configurations. In all experiments, the nodes execute a single application that listens to a fixed UDP port. Messages are generated by a client program on the Linux host and are forwarded over the multihop network to the destination node. Depending on the experiment, different nodes on the 6LoWPAN network are selected as destination. Every node needs to process packets using the 6LoWPAN layer and μIP. If IPsec is used, the Linux host and the destination node must perform cryptographic processing. If link-layer security is used all wireless nodes in the transmission path are involved in cryptographic processing. The message payload is forwarded to the application on the node that generates a new datagram of the same size as response. This reply is sent back to the Linux host following the reverse transmission path. For some of the experiments, IPsec is used to provide secure E2E communication between the destination node and the Internet host, whereas in other experiments, hop-to-hop security is provided at the link layer.

As we aim to evaluate security-related performance aspects, we use Contiki's NullMAC, which does not apply any power-saving mechanisms. Therefore, we avoid measuring security-unrelated performance aspects of the experimental setup.

6.2. Memory footprint comparison

We measure the ROM and RAM footprints of our link-layer security and IPsec implementation (see Table I). The memory footprints are compared with a reference Contiki system including µIP and SICSLoWPAN.

The memory footprint overhead of the link-layer security implementation is very small. Most of the processing is performed by the CC2420 radio chip, and little functionality has to be added to the Contiki operating system.

The memory footprint overhead of the IPsec implementation is comparatively larger as most cryptographic algorithms have to be implemented in software. However, with hardware AES support the overhead is significantly decreased. A pure software implementation of ESP with AES-CBC + AES-XCBC-MAC consumes 35.3 Kb of ROM that can be reduced to 30.5 Kb with hardware support. Table I shows a full comparison of memory footprints with no security, with link-layer security, with software implemented IPsec, and with hybrid implementation of IPsec when AES is performed in hardware and modes of operations are performed in software. It is worth mentioning that unlike AES-CBC, the AES-CTR mode of operation only relies on AES encryption. Thus, the AES-CTR + AES-XCBC-MAC-96 configuration can be implemented without AES decryption, resulting in a particularly low-memory footprint. The ROM additional overhead is 2.8 Kb that is comparable with the 1.2 Kb overhead of the link-layer footprint. The RAM footprint is calculated as the sum of the global data segments and the runtime stack usage that we measure by running Contiki in the MSPSim emulator [36]. With an additional footprint of 1.1 kB, the AH HMAC-SHA1 configuration is the most RAM-consuming configuration. When using other modes of operation, the RAM usage lies between only 0.3 and 0.5 kB.

These results show that there is always significant space for the application-layer programs as the Flash ROM size of the Tmote Sky is 48 Kb. Hence, link-layer security and IPsec (with AH and ESP) can be embedded in constrained devices.

6.3. Header overhead comparison

Both link-layer security and IPsec require additional headers, which increases header overhead. In this section, we compare the header overhead of IPsec and 802.15.4 link-layer security.

When large packets have to be carried, 6LoWPAN link-layer fragmentation has to be used. 6LoWPAN defines a fragmentation scheme in which every fragment contains a reassembly tag and an offset. When security is enabled, either IPsec or link-layer security, the IEEE 802.15.4 frame size may exceed the MTU size of 127. In that case, addition fragment(s) are needed, which result in energy/network overhead.

As IPsec operates at the network layer, its header is added to every IP datagram. However, no additional header related to security has to be added to potential 6LoWPAN link-layer fragments. When using 802.15.4 link-layer security, every frame is secured independently. Thus, security-related header information has to be added to every single 6LoWPAN link-layer fragment, and the overall header overhead is dependent on the number of fragments transported. Table II shows the per message header overhead incurred using IPsec and link-layer security for different security services.

Integrity only. When using link-layer security, the packet overhead for the authentication scheme is exactly the length of the MAC, that is, 12 bytes per fragment when using AES-CBC-MAC-96. With IPsec and AES-XCBC-

Table I. Memory footprints show that AH and ESP consume just 1.5Kb and 4.4Kb for standard recommended IPsec algorithms using hardware-aided AES.

	ROM (kB)		RAM (kB)	
System	overall	diff	overall	diff
Without Security	26.1	–	8.0	–
802.15.4 Link-Layer (all modes)	27.3	1.2	8.0	–
AH with HMAC-SHA1-96	30.7	4.6	9.1	1.1
AH with XCBC-MAC-96	32.3	6.2	8.5	0.5
AH with XCBC-MAC-96 (with hardware)	27.6	1.5	8.3	0.3
ESP with AES-CBC	34.8	8.7	8.3	0.3
ESP with AES-CBC (with hardware)	30.0	3.9	8.3	0.3
ESP with AES-CTR	33.2	7.1	9.1	0.3
ESP with AES-CTR (with hardware)	28.4	2.3	9.1	0.3
ESP with AES-XCBC-MAC-96	32.7	6.6	8.3	0.3
ESP with AES-XCBC-MAC-96 (with hardware)	28.0	1.9	8.3	0.3
ESP with AES-CBC + AES-XCBC-MAC-96	35.3	9.2	8.3	0.3
ESP with AES-CBC + AES-XCBC-MAC-96 (with hardware)	30.5	4.4	8.3	0.3
ESP with AES-CTR + AES-XCBC-MAC-96	33.7	7.6	8.3	0.3
ESP with AES-CTR + AES-XCBC-MAC-96 (with hardware)	28.9	2.8	8.3	0.3

Table II. The header overhead of compressed IPsec is slightly larger than that of 802.15.4, while IPsec provides end-to-end security. As soon as fragmentation is needed, IPsech as a lower header overhead than link-layer security.

	Compressed IPsec		802.15.4 link-layer security	
Service	Mode	Overhead	Mode	Overhead
Integrity	HMAC-SHA1-96	16 *B*	AES-CBC-MAC-96	12 *B* × *nfrags*
Confidentiality	AES-CBC	12 *B*	AES-CTR	5 *B* × *nfrags*
Integrity and Confidentiality	AES-CBC and HMAC-SHA1-96	26 *B*	AES-CCM-128	21 *B* × *nfrags*

MAC-96, the AH header involves an overhead of 24 bytes per IP packet. Thanks to the IPsec header compression we defined, this overhead is reduced to 16 bytes. As soon as the IP datagram requires more than a single fragment, IPsec provides a lower header overhead than link-layer security.

Confidentiality only. If only message encryption is required, the 802.15.4 link-layer security provides AES-CTR, which has a 5-byte overhead per fragment. In comparison, IPsec with ESP and AES-CBC leads to an overhead of 18 bytes per packet, reduced to 14 bytes thanks to header compression. Here, the data overhead of IPsec is amortized for datagrams of three or more fragments.

Integrity and confidentiality. With AES-CCM-128, the overhead for 802.15.4 is 21 bytes per fragment, whereas IPsec ESP has an overhead of 30 bytes per packet, reduced to 26 bytes when using our 6LoWPAN compression extension. Again, IPsec offers a lower header overhead than link-layer security if fragmentation is required.

6.4. Evaluation of cryptographic algorithms

In this section, we analyze cryptographic algorithms used in IPsec and 802.15.4 link-layer security. We investigate only the time necessary to perform encryption and do not take packet processing within the communication stack into account. We carry out the analysis in the next paragraph.

The purely hardware-based encryption used for link-layer security is extremely efficient. Compared with IPsec, processing times and energy consumption are very small when looking at a single node. The link-layer processing time for a message of 512 bytes is less than 0.3 ms, which corresponds to an energy consumption of 2 μJ. Figure 9 shows the IPsec overhead in time for different message sizes, with or without hardware encryption. The authentication algorithms are compared separately for AH and ESP: with AH, the MAC is calculated over the IP header and payload packet, whereas in ESP the IP header is neither encrypted nor authenticated.

Regarding authentication, the cost is higher for AH than for ESP because AH needs to process the 40-byte IP header. HMAC-SHA1-96 is not as efficient as the other solutions as it has a high fixed-cost for small data sizes and poor scalability.

The efficiency of IPsec calculation can be improved by employing cryptographic functions provided by node hardware. For example, the CC2420 radio chip present on many node platforms provides such functionality. This allows part of the encryption and decryption to be performed faster, using less energy. In the benchmark, it can be seen that by using hardware encryption, the overhead

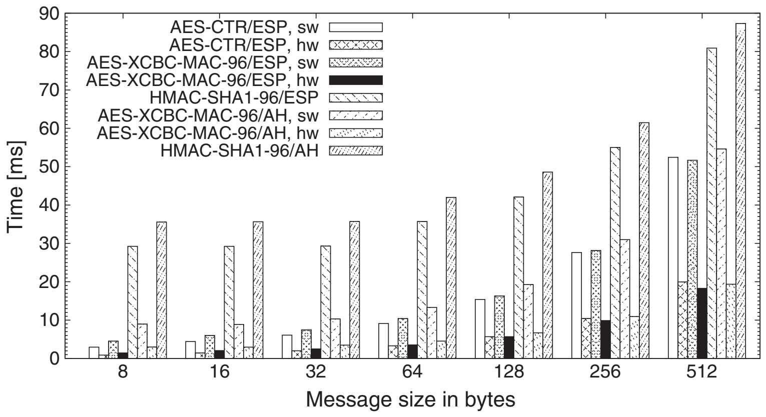


Figure 9. Micro-benchmarks for the supported security algorithms. Also, pure link-layer encryption was evaluated but is not shown, as the maximum overhead, the time for 512 bytes, was 0.27 ms, which would not be visible in the graph. The comparison shows that AES-CTR and AES-XCBC-MAC-96 are the most efficient in terms of processing time and energy consumption, especially if hardware encryption is available. Together, they provide the best level of security among the standardized algorithms.

is reduced with up to 60% in time, with a corresponding reduction in energy.

We have also evaluated AES-CBC, but because the performance is very similar to AES-CTR, we omit the results from the graph. The proposed standard for cryptographic suites for IPsec [37] specifies that future IPsec systems will need to provide at least AES-CBC-128 for encryption and AES-XCBC-MAC-96 mode for authentication [37]. Because AES-CTR gives equivalent or better results (and is also part of the suggested standard), we use AES-CTR together with AES-XCBC-MAC in the following energy experiments.

6.5. Energy consumption comparison

Securing the IoT has a cost in terms of added energy usage. We measure the energy overhead using 802.15.4 link-layer security with AES-CCM and IPsec with AES-CTR + AES-XCBC-MAC. Measurements are carried out on a Tmote Sky using Powertrace, Contiki's integrated power profiler [38]. To provide a fair comparison across different experiments, we consider a constant voltage of 3 V, and we report energy rather than mote lifetime—lifetime cannot be estimated accurately because the way battery discharge varies with the environmental settings and across different instances of a same hardware [39].

Powertrace provides an estimation of the energy consumption using power state tracking. This technique allows comparing different experiments in a fair way that is difficult to achieve with hardware energy measurement, because of the properties of battery [39].

For the experiments, we start counting of CPU ticks on the destination node (communication endpoint in the wireless network) from when the first fragment of the incoming message is received and decryption is started. We stop counting when the link-layer processing of the last fragment of the response message is finished. We ignore network processing times, the time the node spends on waiting for an incoming fragment or for the completion of a fragment transmission. In total, we include in the measurements the link-layer processing, 6LoWPAN processing, μIP stack handling, and application-layer processing. These experiments are run with and without hardware support.

Figure 10 shows the energy consumption of a client node using only link-layer security, IPsec using either AH or ESP, and with no security. The results show that the energy consumption with IPsec is clearly increased compared with using no security or when just using link-layer security. Still we argue that the overhead is acceptable in cases where E2E security is needed. Moreover, the overhead is small if compared with the energy consumption of the radio chip when it is turned on. In a radio duty-cycled network with one percent listening time, in less than 2 s of listening, the radio consumes as much energy as the maximum measured security overhead, around 1.2 mJ for ESP software processing of a 512-byte message.

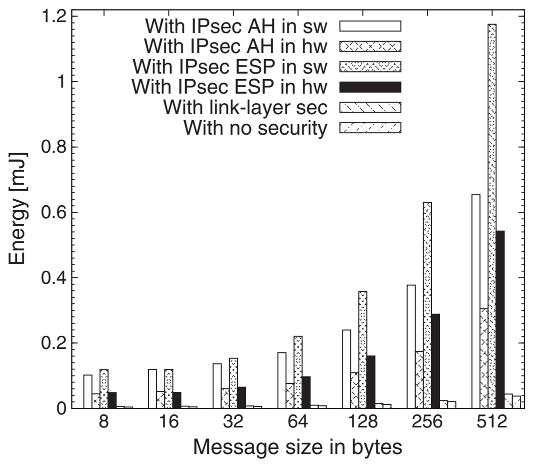


Figure 10. The node energy consumption is lower without IPsec and higher for Encapsulating Security Payload (ESP) than for authentication header (AH).

Whenever supported by the hardware, the overhead can be reduced by employing hardware-aided cryptographic processing. As can be seen in Figure 10, hardware-aided encryption reduces the energy consumption by 50%.

Although the IPsec overhead seems large for a destination node, we get a different picture if we focus on a forwarding node. With IPsec, only end nodes have to perform security operations. With link-layer security, every node in the path has to process encryption/decryption and/or authentication because it needs to access (protected) IP header fields in order to perform routing. As a result, IPsec has no overhead for forwarding nodes, besides a small added amount of 6LoWPAN header processing. For link-layer security on the other hand, the energy consumption grows with every node on the path between two end nodes. Unlike link-layer security, IPsec can thereby be used together with nodes that do not support any security at all. The measurements for nodes acting as forwarders in the multihop network are consistent with the micro-benchmarks, with a security processing time overhead of 0.3 ms for forwarding a 512-byte message. This does not include the radio transmission times—added link-layer security headers can cause a message to be fragmented in more packets, in which case, the system total energy overhead will increase noticeably. This is further discussed in the following section.

6.6. Overall network performance

We evaluate the system response time for different security solutions and services with different network diameters and different payload sizes. The response time in this case is the time an IP datagram takes to travel from an IP-connected Linux machine to a smart object plus the travel time of the response back to the sender. We let the size of the IP datagrams vary between 16 and 512 bytes. The number of hops ranges from one to four hops. We conduct every experiment 10 times and display the mean values using error bars for the standard deviation. We measure response time with 802.15.4 link-layer security enabled, with IPsec, and with no security. There are different modes of operation in both IPsec and link-layer security, which correspond to different security services. We present our results in accordance with the security services needs.

6.6.1. Performance impact of integrity

We first focus on the case where data integrity is required. IEEE 802.15.4 provides integrity services with AES-CBC-MAC with MIC sizes of 4, 8, and 16 bytes. IPsec provides integrity services with AH and optionally with ESP with a MIC size of 12 bytes.

Figure 11 shows the response time in relation to the network diameter for link-layer AES-CBC-MAC with minimum and maximum MIC sizes, IPsec AH with XCBC-MAC-SHA1-96, and without security. For small data sizes, the overhead of link-layer security is negligible. When carrying 512 bytes over four hops, pure software AH involves an overhead of 26%, which is reduced to 11% with the help of hardware AES. A point worth mentioning is that for a given data size, the overhead of IPsec is constant over the number of hops. For intermediate nodes, the cost of forwarding the data with and without IPsec is the same; the overhead is only incurred for computation on the end nodes. With large data sizes (512 bytes), the overhead of link-layer security increases with the number of hops, because of the per-fragment header overhead. As a result, with two hops or more, AH is faster than link-layer security.

Figure 12 shows the response time, depending on the IP payload size, of link-layer AES-CBC-MAC with minimum and maximum MIC sizes, IPsec AH overhead with XCBC-MAC-SHA1-96, and with no security. Consistently with the micro-benchmarks in Figure 9, the overhead of IPsec grows linearly with datagram sizes. In the four-hop case, we observe that hardware-aided AH is faster than link-layer security for data sizes of 256 bytes and more.

6.6.2. Performance impact of integrity and confidentiality

We now focus on the case where both integrity and confidentiality are required. These services are supported by 802.15.4 through the AES-CCM mode. IPsec ESP provides a combined integrity and confidentiality service—in this experiment, we used AES-CTR together with AES-XCBC-MAC.

Figures 13 and 14 show the response times obtained with different IPsec, link-layer security, and with no security. For small data sizes, the overhead of link-layer security is very small. Over four hops and with a data size of 512 bytes, ESP incurs an overhead of 46%, lowered to 19% with hardware-aided AES. As the number of hops grows or the data size increases, ESP becomes faster than link-layer security. This result is opposed to the common belief that link-layer security is the fastest solution in all cases.

6.7. Summary

Our evaluation shows that IPsec fits in a tiny node (e.g., Tmote Sky) with room still available for applications. Our cryptographic algorithms analysis shows that our implementation of AES-CTR and AES-XBC-MAC-96 is the fastest and most energy-efficient. We showed that link-layer security is more efficient for the end node in our experiment, whereas IPsec is more efficient for forwarding nodes. In absolute terms, the energy overheads we measured are not significant when compared with the energy consumption of a node in steady-state operation.

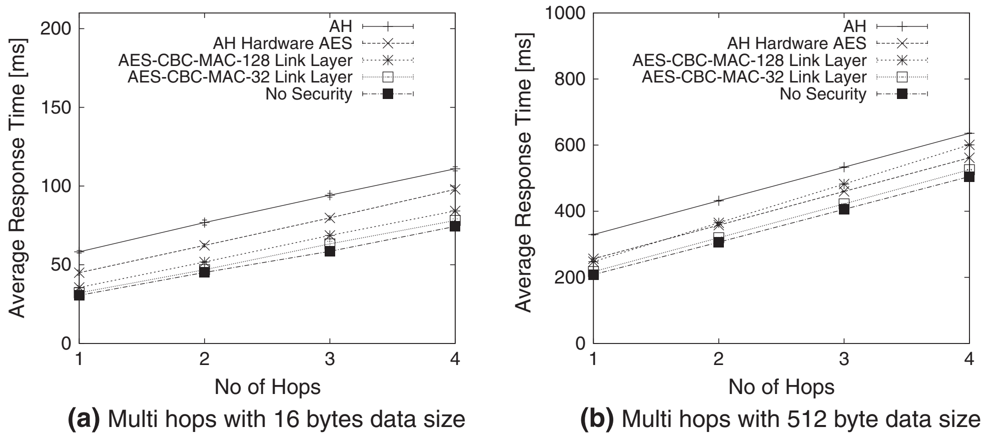
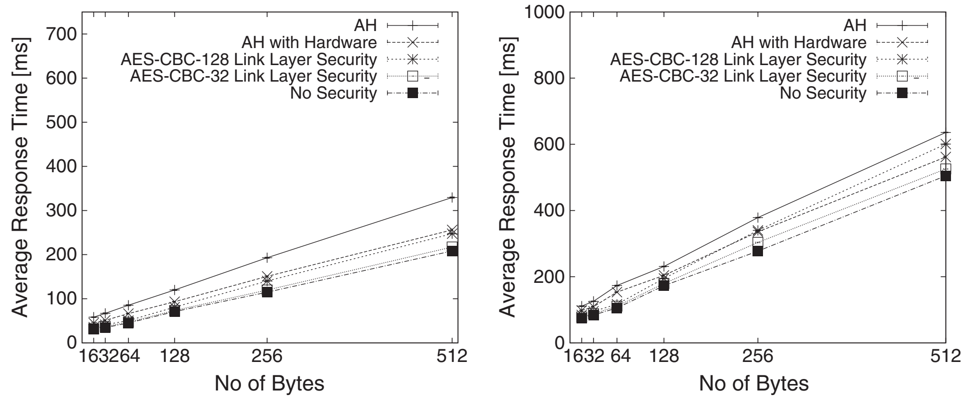
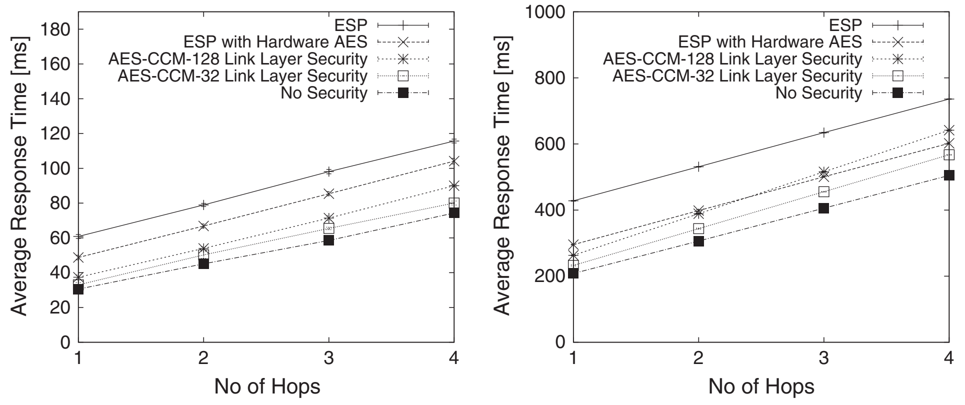


Figure 11. Response time versus number of hops when *integrity* is enabled. The 802.15.4 link-layer security is better for small data sizes, whereas IPsec gets better for large data sizes across multiple hops. The overhead of IPsec is constant across a single-hop and a multihop network, whereas link-layer security overhead increases with the number of hops.



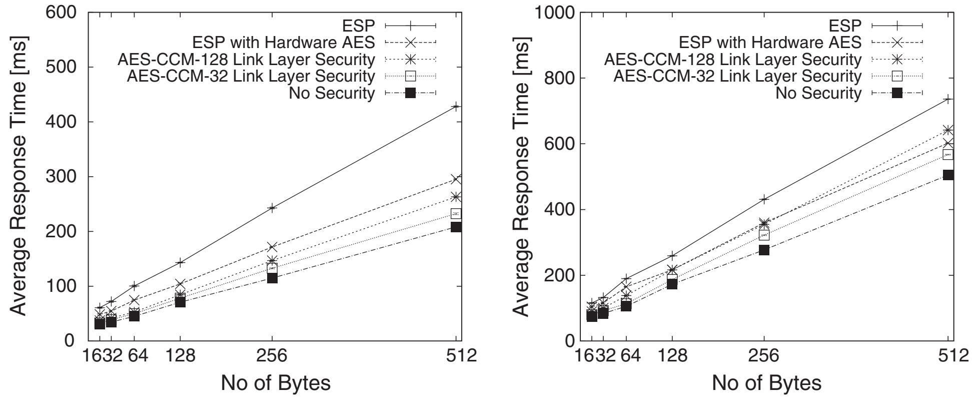
(a) Single hop with different data sizes **(b)** Four hops with different data sizes

Figure 12. Response time versus datagram size when *integrity* is enabled. The 802.15.4 link-layer security is better for single hop, whereas IPsec gets better for multihop networks. The overhead of IPsec is constant across a single hop and a multihop for different data sizes, whereas link-layer security overhead increases with the increasing data sizes.



(a) Multi hops with 16 bytes data size **(b)** Multi hops with 512 byte data size

Figure 13. Response time versus number of hops when *integrity* and *confidentiality* is enabled. The 802.15.4 link-layer security is better for small data sizes while IPsec gets better for large data sizes across multiple hops.



(a) Single hop with different data sizes **(b)** Four hops with different data sizes

Figure 14. Response time versus datagram size when *integrity* and *confidentiality* is enabled. The 802.15.4 link-layer Security is better for single hop, whereas IPsec gets better for multihop networks. The overhead of IPsec is constant across a single hop and a multihop network for different data sizes while link-layer security overhead increases with the increasing data sizes.

The system-wide response time shows a counter-intuitive result: in spite of its higher software complexity, IPsec provides a better scalability than link-layer security. By raising the level at which the security is guaranteed, it substantially reduces the data overhead on fragmented traffic. Thus, for large IP packets and/or a large number of hops, IPsec is faster than link-layer security. Our results also show that IPsec can be substantially accelerated with the help of the cryptographic co-processor included in modern radio chips such as the CC2420.

7. CONCLUSION

The future IoT will be an all-IP network. As it will be the foundation of many services, our daily life will depend on its availability and reliable operation. It is therefore important to find mechanisms providing security in the IoT. As the existing IEEE 802.15.4 link-layer security does not provide the required E2E security, alternative or complementary mechanisms must be found. In this paper, we have shown that IPsec implemented through 6LoWPAN extensions is a feasible option for providing E2E security in the IoT. This paper has presented a thorough evaluation of the proposed IPsec solution and has compared its performance with currently employed IEEE 802.15.4 link-layer security.

We plan to extend our IPsec solution with the IKE mechanism and to investigate the full system in larger deployments to evaluate scalability and usability of the approach. We also plan to research the 6LoWPAN compression mechanisms for TLS/DTLS and implement and evaluate it.

ACKNOWLEDGEMENTS

This work has been supported by the VINNOVA. This work has been partly performed within the SICS Center for Networked Systems funded by VINNOVA, SSF, KKS, ABB, Ericsson, Saab SDS, TeliaSonera, T2Data, Vendolocus and Peerialism. The Contiki IPsec/6LoWPAN extensions and the Contiki 802.15.4 link-layer security extensions described in this paper are available on request. Please e-mail `shahid@sics.se` to request a copy.

REFERENCES

1. Vasseur J, Dunkels A. *Interconnecting smart objects with IP—the next Internet*. Morgan Kaufmann: Waltham, Massachusetts, 2010.
2. IEEE std. 802.15.4—2003: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs). IEEE, 2003.
3. Deloche G, Kushalnagar N, Hui J, Culler D. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, 2007.
4. Kent S, Seo K. Security Architecture for the Internet Protocol. RFC 4301, 2005.
5. Dunkels A, Grönvall B, Voigt T. Contiki—a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of 1st IEEE Workshop on Embedded Networked Sensors (EmNetS'04)*, Tampa, USA, 2004.
6. Liu A, Ning P. TinyECC: a configurable library for elliptic curve cryptography in wireless sensor networks. In *Proceedings of 7th International Conference on Information Processing in Sensor Networks (IPSN'08)*, Washington, DC, USA, 2008.
7. Szczechowiak P, Oliveira L, Scott M, Collier M, Dahab R. NanoECC: testing the limits of elliptic curve cryptography in sensor networks. In *Proceedings of 5th European Conference on Wireless Sensor Networks (EWSN'08)*, Bologna, Italy, 2008.
8. Wood A, Stankovic J. Poster abstract: AMSecure—secure link-layer communication in TinyOS for IEEE 802.15.4-based wireless sensor networks. In *Proceedings of 4th ACM Conference on Networked Embedded Sensor Systems (SenSys'06)*, Boulder, USA, 2006.
9. Hu W, Corke P, Shih W, Overs L. secfleck: a public key technology platform for wireless sensor networks. In *Proceedings of 6th European Conference on Wireless Sensor Networks (EWSN'09)*, Cork, Ireland, 2009.
10. Liu D, Ning P. Establishing pairwise keys in distributed sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS)*, New York, NY, USA, 2003.
11. Chung A, Roedig U. DHB-KEY: an efficient key distribution scheme for wireless sensor networks. In *Proceedings of 4th IEEE International Workshop on Wireless and Sensor Networks Security (WSNS'08)*, Atlanta, USA, 2008.
12. ArchRock Corporation. Phynet n4x series, 2008.
13. Moskowitz R. HIP Diet EXchange (DEX). draft-moskowitz-hip-rg-dex-05, 2011.
14. Roman R, Alcaraz C, Lopez J, Sklavos N. Key management systems for sensor networks in the context of the Internet of Things. *Computers and Electrical Engineering* 2011; **37**(2): 147–159.
15. Dierks T, Allen C. The TLS protocol version 1.0. RFC 2246, 1999.
16. Hong S, Kim D, Ha M, *et al*. SNAIL: an IP-based wireless sensor network approach to the Internet of Things. *Wireless Communications, IEEE* 2010; **17**(6): 34–42.
17. Fouladgar S, Mainaud B, Masmoudi K, Afifi H. Tiny 3-TLS: a trust delegation protocol for wireless sensor networks. In *Proceedings of 3rd European Workshop on Security and Privacy in Ad-hoc and Sensor Networks (ESAS'03)*, Hamburg, Germany, 2006.

18. Granjal J, Silva R, Monteiro E, Sa Silva J, Boavida F. Why is IPsec a viable option for wireless sensor networks. In *Proceedings of 4th IEEE International Workshop on Wireless and Sensor Networks Security (WSNS'08)*, Atlanta, USA, 2008.
19. Riaz R, Kim K-H, Ahmed HF. Security analysis survey and framework design for IP connected LoWPANs. In *Proceedings of 9th International Symposium on Autonomous Decentralized Systems (ISADS'09)*, Athens, Greece, 2009.
20. Roman R, Lopez J. Integrating wireless sensor networks and the internet: a security analysis. *Internet Research* 2009; **19**(2): 246–259.
21. Alcaraz C, Najera P, Lopez J, Roman R. Wireless sensor networks and the Internet of Things: do we need a complete integration? In *Proceedings of 1st International Workshop on the Security of the Internet of Things (SecIoT'10)*, Tokyo, Japan, 2010.
22. Granjal J, Monteiro E, Sá Silva J. Enabling network-layer security on IPv6 wireless sensor networks. In *Proceedings of IEEE Global Communications Conference (GLOBECOM'10)*, Miami, USA, 2010.
23. Raza S, Chung SDA, Yazar D, Voigt T, Roedig U. Securing communication in 6LoWPAN with compressed IPsec. In *Proceedings of 7th International Conference on Distributed Computing in Sensor Systems (DCOSS'11)*, Barcelona, Spain, 2011.
24. Deering S, Hinden R. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, 1998.
25. Hui J, Thubert P. Compression Format for IPv6 Datagrams in Low Power and Lossy Networks. draft-ietf-6lowpan-hc-15, 2011.
26. Sastry N, Wagner D. Security considerations for IEEE 802.15.4 networks. In *Proceedings of the 3rd ACM Workshop on Wireless Security*, WiSe'04, New York, NY, USA, 2004; 32–42. ACM.
27. Kent S. IP Authentication Header. RFC 4302, 2005.
28. Kent S. IP Encapsulating Security Payload. RFC 4303, 2005.
29. Manral V. Cryptographic algorithm implementation requirements for encapsulating security payload (ESP) and authentication header (AH). RFC 4835, 2007.
30. Sakane S, Kamada K, Thomas M, Vilhuber J. Kerberized Internet Negotiation of Keys (KINK). RFC 4430, 2006.
31. Kaufman C, Hoffman P, Nir Y, Eronen P. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 5996 (Proposed Standard), 2010. Updated by RFC 5998.
32. Richardson M. A Method for Storing IPsec Keying Material in DNS. RFC 4025 (Proposed Standard), 2005.
33. Polastre J, Szewczyk R, Culler D. Telos: enabling ultra-low power wireless research. In *Proceedings of 4th International Conference on Information Processing in Sensor Networks (IPSN'05)*, Los Angeles, USA, 2005.
34. Shamus Software. Multiprecision Integer and Rational Arithmetic C/C++ Library.
35. Tsiftes N, Dunkels A. A database in every sensor. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Seattle, WA, USA, 2011.
36. Eriksson J, Dunkels A, Finne N, Österlind F, Voigt T. MSPsim—an extensible simulator for MSP430-equipped sensor boards. In *Proceedings of 4th European Conference on Wireless Sensor Networks (EWSN'07)*, Delft, The Netherlands, 2007. Demo Session.
37. Hoffman P. Cryptographic Suites for IPsec. RFC 4308, 2005.
38. Dunkels A, Eriksson J, Finne N, Tsiftes N. Powertrace: Network-level Power Profiling for Low-power Wireless Networks. Technical Report T2011:05, Swedish Institute of Computer Science, 2011.
39. Park C, Lahiri K, Raghunathan A. Battery discharge characteristics of wireless sensor nodes: an experimental analysis. In *Proceedings of the IEEE Conf. on Sensor and Ad-hoc Communications and Networks (SECON)*, Santa Clara, California, USA, September 2005.